ADOBE® AIR® Uygulamaları Oluşturma

Yasal bildirimler

Yasal bildirimler için bkz. http://help.adobe.com/tr_TR/legalnotices/index.html.

İçindekiler

Bölüm 1: Adobe AIR hakkında

Adobe® AIR®, masaüstünde ve mobil aygıtlarda zengin Internet uygulamaları (RIA'lar) oluşturmak ve dağıtmak için var olan web geliştirme becerilerinizi geliştirmenize olanak veren işletim sistemleri arası, çok ekranlı bir çalışma zamanıdır. Adobe® Flex ve Adobe® Flash® (SWF tabanlı) uygulamalarını kullanarak ActionScript 3.0 ile masaüstü, televizyon ve mobil AIR uygulamaları oluşturulabilir. Masaüstü AIR uygulamaları ayrıca HTML, JavaScript® ve Ajax (HTML tabanlı) ile de oluşturulabilir.

Adobe AIR'yi kullanmaya başlama hakkında daha fazla bilgiyi Adobe AIR Geliştirici Bağlantısı'nda (http://www.adobe.com/devnet/air/) bulabilirsiniz.

Kullanırken rahat ettiğiniz araçlardan ve yaklaşımlardan faydalanmak için AIR, tanıdığınız ortamlarda çalışmanıza olanak tanır. Flash, Flex, HTML, JavaScript ve Ajax'ı destekleyerek, ihtiyaçlarınızı karşılayan en iyi deneyime sahip olabilirsiniz.

Örneğin aşağıdaki teknolojilerin biri veya bir kombinasyonu kullanılarak uygulamalar geliştirilebilir:

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

Kullanıcılar AIR uygulamalarıyla etkileşime, yerel uygulamalarıyla geçtikleri gibi geçerler. Kullanıcının bilgisayarına veya aygıtına çalışma zamanı yüklenir, ardından AIR uygulamaları yüklenir ve diğer masaüstü uygulamaları gibi çalıştırılır. (iOS'de, ayrı bir AIR çalışma zamanı yüklenmez; her iOS AIR uygulaması bağımsız bir uygulamadır.)

Çalışma zamanı, uygulamaların konuşlandırılması için tutarlı bir işletim sistemleri arası platform ve çerçeve sağlar ve böylece masaüstleri arasında tutarlı işlevsellik ve etkileşim sağlayarak tarayıcılar arası testi ortadan kaldırır. Belirli bir işletim sistemi için geliştirme yapmak yerine, çalışma zamanını hedef aldığınızda şu gibi faydalardan yararlanırsınız:

- AIR için geliştirilen uygulamalar, size ek bir iş yükü getirmeden birden çok işletim sistemi arasında çalışır. Çalışma zamanı, AIR tarafından desteklenen tüm işletim sistemlerinde tutarlı ve tahmin edilebilir sunumlar ve etkileşimler sağlar.
- Uygulamalar, varolan web teknolojileri ve tasarım modellerinden faydalanmanıza izin vererek daha hızlı oluşturulabilir. Geleneksel masaüstü geliştirme teknolojilerini veya yerel kodun karmaşıklığını öğrenmeden web tabanlı uygulamaları masaüstüne genişletebilirsiniz.
- Uygulama geliştirme, C ve C++ gibi düşük düzeyli diller kullanmaktan daha kolaydır. Her işletim sistemi için farklı olan, karmaşık, düşük düzey API'leri öğrenmeniz gerekmez.

AIR için uygulama geliştirirken, zengin bir çerçeve ve API kümesi geliştirebilirsiniz:

- Çalışma zamanı ve AIR çerçevesi tarafından sağlanan AIR'e özgü API'ler
- SWF dosyalarında ve Flex çerçevesinde kullanılan ActionScript API'leri (bunun yanında diğer ActionScript tabanlı kütüphaneler ve çerçeveler)
- HTML, CSS ve JavaScript
- Çoğu Ajax çerçevesi
- Yerel kodda programlanan platforma özgü işlevselliğe erişmenize izin veren ActionScript API'leri sağlayan Adobe AIR'ye yönelik yerel uzantılar. Yerel uzantılar ayrıca eski yerel koda ve daha yüksek performans sağlayan yerel koda erişim sağlayabilir.

AIR, uygulamaların oluşturulma, konuşlandırılma ve deneyim edilme şekillerini ciddi bir şekilde değiştirir. Daha fazla yaratıcı kontrol elde edersiniz ve Flash, Flex, HTML ve Ajax tabanlı uygulamalarınızı masaüstüne, mobil aygıtlara ve televizyonlara genişletebilirsiniz.

Her yeni AIR güncellemesine nelerin dahil edildiğiyle ilgili bilgi için Adobe AIR Sürüm Notları'na bakın (http://www.adobe.com/go/learn_air_relnotes_tr).

Bölüm 2: Adobe AIR yüklemesi

Adobe® AIR® çalışma zamanı, AIR uygulamalarını çalıştırmanıza olanak verir. Çalışma zamanını şu şekillerde yükleyebilirsiniz:

- Çalışma zamanını ayrı yükleyerek (bir AIR uygulaması da yüklemeden)
- Bir AIR uygulamasını ilk kez web sayfası yükleme "işareti" ile yükleyerek (çalışma zamanını yüklemeniz de istenir)
- Hem uygulamanızı hem de çalışma zamanını yükleyen özel bir yükleyici oluşturarak. AIR çalışma zamanını bu şekilde dağıtmak için Adobe'den onay almanız gerekir. Adobe çalışma zamanı lisanslama sayfasında onay isteyebilirsiniz. Adobe'nin böyle bir yükleyici oluşturmak için araçlar sağlamadığını unutmayın. Bununla birlikte, çok sayıda üçüncü taraf yükleyici araç takımı da kullanılabilir.
- AIR'yi sabit çalışma zamanı olarak paketleyen AIR uygulamasını yükleyerek. Sabit çalışma zamanı yalnızca paketleme uygulaması tarafından kullanılır. Diğer AIR uygulamalarını çalıştırmak için kullanılmaz. Çalışma zamanının paketlenmesi Mac ve Windows'ta bir seçenektir. iOS'ta tüm uygulamalar paketlenmiş bir çalışma zamanını içerir. AIR 3.7 sürümünden itibaren tüm Android uygulamaları varsayılan olarak bir paketlenmiş çalışma zamanı içerir (ayrı bir çalışma zamanı kullanma seçeneğine sahip olmanıza rağmen).
- AIR SDK, Adobe® Flash® Builder™ veya Adobe Flex® SDK gibi (AIR komut satırı geliştirme araçlarını içeren) bir AIR geliştirme ortamı kurarak SDK'de bulunan çalışma zamanı yalnızca uygulamalarda hata ayıklanırken kullanılır — yüklenen AIR uygulamalarını çalıştırmak için kullanılmaz.

AIR'i yüklemek ve AIR uygulamalarını çalıştırmak için sistem gereksinimleri burada ayrıntılı biçimde açıklanmıştır: Adobe AIR: Sistem gereksinimleri(http://www.adobe.com/tr/products/air/systemreqs/).

Çalışma zamanı yükleyicisi ve AIR uygulama yükleyicisi yüklendiğinde, güncellendiğinde veya AIR uygulamalarını veya AIR'in kendisini kaldırdığında günlük dosyaları oluşturur. Herhangi bir yükleme sorununun nedenini belirlemenize yardımcı olması için bu günlüklere başvurabilirsiniz. Bkz. Yükleme günlükleri.

Adobe AIR'yi yükleme

Kullanıcı, çalışma zamanını yüklemek veya güncellemek için bilgisayara ilişkin yönetici ayrıcalıklarına sahip olmalıdır.

Çalışma zamanını Windows kurulu bir bilgisayara yükleme

1. Çalışma zamanı yükleme dosyasını http://get.adobe.com/air/ adresinden indirin.
2. Çalışma zamanı yükleme dosyasını çift tıklatın.
3. Yüklemeyi tamamlamak için yükleme penceresindeki istemleri izleyin.

Çalışma zamanını Mac kurulu bir bilgisayara yükleme

1. Çalışma zamanı yükleme dosyasını http://get.adobe.com/air/ adresinden indirin.
2. Çalışma zamanı yükleme dosyasını çift tıklatın.
3. Yüklemeyi tamamlamak için yükleme penceresindeki istemleri izleyin.
4. Yükleyici bir Kimlik Denetimi penceresi görüntülerse, Mac OS kullanıcı adınızı ve parolanızı girin.

Bir Linux bilgisayarına çalışma zamanını yükleyin

Not: *Şu anda, AIR 2.7 ve üstü Linux'ta desteklenmemektedir. Linux'a dağıtılan tüm AIR uygulamalarının, AIR 2.6 SDK uygulamasını kullanmaya devam etmesi gerekir.*

İkili yükleyiciyi kullanma:

1. http://kb2.adobe.com/cps/853/cpsid_85304.html adresinden yükleme ikili dosyasını bulun ve indirin.
2. Dosya izinlerini, yükleyici uygulamanın yürütülebileceği biçimde ayarlayın. Bir komut satırından dosya izinlerini şu şekilde ayarlayabilirsiniz:

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

   Bazı Linux sürümleri, dosya izinlerini bir bağlam menüsü aracılığıyla açılan Özellikler iletişim kutusunda ayarlamanıza olanak verir.
3. Yükleyiciyi komut satırından veya çalışma zamanı yükleme dosyasını çift tıklatarak çalıştırın.
4. Yüklemeyi tamamlamak için yükleme penceresindeki istemleri izleyin.

Adobe AIR yerel bir paket olarak yüklenir. Diğer bir deyişle rpm temelli bir dağıtımda rpm olarak, bir Debian dağıtımında deb olarak. AIR şu anda başka bir paket biçimini desteklememektedir.

Paket yükleyicileri kullanma:

1. http://kb2.adobe.com/cps/853/cpsid_85304.html adresinden AIR paket dosyasını bulun. Sisteminizin desteklediği paket biçimine bağlı olarak rpm veya Debian paketini indirin.
2. Gerekirse paketi yüklemek için AIR paket dosyasını çift tıklatın.

   Komut satırından da yükleyebilirsiniz:

   a. Bir Debian sisteminde:

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

   b. Rpm temelli bir sistemde:

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

      Veya mevcut bir sürümü güncelliyorsanız (AIR 1.5.3 veya üstü):

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

AIR 2 ve AIR uygulamalarını yüklemek için bilgisayarınızda yönetici ayrıcalıklarına sahip olmanız gerekmektedir.

Adobe AIR aşağıdaki konuma yüklenir: /opt/Adobe AIR/Versions/1.0

AIR, "application/vnd.adobe.air-application-installer-package+zip" mime türünü kaydeder, yani .air dosyaları bu mime türündedir ve bu nedenle AIR çalışma zamanıyla kaydedilir.

Çalışma zamanını Android aygıtına yükleme

Android Market'tan AIR çalışma zamanının en son sürümünü yükleyebilirsiniz.

AIR çalışma zamanının geliştirme sürümlerini bir web sayfasındaki bağlantıdan veya ADT `-installRuntime` komutunu kullanarak yükleyebilirsiniz. Bir defada yalnızca bir AIR çalışma zamanı yüklenebilir; hem sürüm hem de geliştirme sürümü yüklü olamaz.

Daha fazla bilgi için bkz. “ADT installRuntime komutu” sayfa 175.

Çalışma zamanını iOS aygıtına yükleme

Gerekli AIR çalışma zamanı kodu iPhone, iTouch ve iPad aygıtları için oluşturulmuş her uygulamayla birlikte paketlenir. Ayrıca bir çalışma zamanı bileşeni yüklemezsiniz.

Daha fazla Yardım konusu

"AIR for iOS" sayfa 68

Adobe AIR'i kaldırma

Çalışma zamanını bir kez yükledikten sonra, aşağıdaki yordamları kullanarak kaldırabilirsiniz.

Çalışma zamanını Windows kurulu bir bilgisayardan kaldırma

1. Windows Başlat menüsünde, Ayarlar > Denetim Masası öğelerini seçin.
2. Programlar, Programlar ve Özellikler veya Program Ekle veya Kaldır denetim masasını açın (kullandığınız Windows sürümüne bağlı olarak).
3. Çalışma zamanını kaldırmak için "Adobe AIR"i seçin.
4. Değiştir/Kaldır düğmesini tıklatın.

Çalışma zamanını Mac kurulu bir bilgisayardan kaldırma

- /Applications/Utilities klasöründe bulunan "Adobe AIR Uninstaller"ı çift tıklatın.

Çalışma zamanını Linux kurulu bir bilgisayardan kaldırma

Şunlardan birini yapın:

- Uygulamalar menüsünden "Adobe AIR Uninstaller" komutunu seçin.
- `-uninstall` seçeneğini içeren AIR yükleyici ikilisini çalıştırın
- AIR paketlerini (`adobeair` ve `adobecerts`) paket yöneticinizle kaldırın.

Çalışma zamanını Android aygıtından kaldırma

1. Aygıtta Ayarlar uygulamasını açın.
2. Uygulamalar > Uygulamaları Yönet öğesinin altıdaki Adobe AIR girişine dokunun.
3. Kaldır düğmesine dokunun.

Ayrıca ADT `-uninstallRuntime` komutunu da kullanabilirsiniz. Daha fazla bilgi için bkz. "ADT uninstallRuntime komutu" sayfa 176.

Paketlenmiş çalışma zamanını kaldırma

Paketlenmiş sabit bir çalışma zamanını kaldırmak için çalışma zamanının birlikte kurulduğu uygulamayı kaldırmalısınız. Sabit çalışma zamanlarının yalnızca yükleme uygulamasını çalıştırmak için kullanıldığını unutmayın.

AIR örnek uygulamalarını yükleme ve çalıştırma

Kullanıcı, bir AIR uygulamasını yüklemek veya güncellemek için bilgisayara yönelik yönetici ayrıcalıklarına sahip olmalıdır.

AIR özellikleri gösteren bazı örnek uygulamalar mevcuttur. Aşağıdaki talimatları kullanarak onlara erişebilir ve onları yükleyebilirsiniz:

1. AIR örnek uygulamalarını indirin ve çalıştırın. Kaynak kodun yanı sıra derlenen uygulamalar da kullanılabilir.

2 Örnek bir uygulamayı indirmek ve çalıştırmak için, örnek uygulamanın Şimdi Yükle düğmesini tıklatın. Uygulamayı yüklemeniz ve çalıştırmanız istenir.

3 Örnek uygulamaları indirmeyi ve sonra çalıştırmayı seçerseniz, indirme bağlarını seçin. AIR uygulamalarını istediğiniz zaman şu şekillerde çalıştırabilirsiniz:

- Windows'ta masaüstündeki uygulama simgesini çift tıklatarak veya Windows Başlat menüsünden seçerek.
- Mac OS'de varsayılan olarak kullanıcı dizininizin Uygulamalar klasöründe yüklü olan uygulama simgesini çift tıklatarak (örneğin Macintosh'ta HD/Users/JoeUser/Applications/).

Not: *Bu talimatların güncellemeleri için, şu adreste bulunan AIR sürüm notlarını kontrol edin: http://www.adobe.com/go/learn_air_relnotes_tr.*

Adobe AIR güncellemeleri

Adobe, belirli aralıklarla yeni özellikler ekleyerek ve küçük sorunları çözerek Adobe AIR'yi günceller. Otomatik Bildirim ve Güncelleme özelliği, Adobe'nin, güncellenmiş bir Adobe AIR sürümü bulunduğunda kullanıcılara otomatik olarak bildirimde bulunmasını sağlar.

Adobe AIR güncellemeleri Adobe AIR'nin doğru çalıştığından emin olmayı sağlar ve genellikle güvenlik açısından önemli değişiklikler içerir. Adobe, kullanıcıların, Adobe AIR'yi her yeni sürüm bulunduğunda, özellikle güvenlik güncellemesi içeriyorsa, son sürümüne güncellemelerini önerir.

Varsayılan olarak, Adobe AIR başlatıldığında çalışma zamanı, herhangi bir güncellemenin bulunup bulunmadığını kontrol eder. Bu kontrolü, son güncelleme kontrolünden itibaren iki haftadan fazla süre geçmişse yapar. Güncelleme mevcutsa AIR bu güncellemeyi arka planda indirir.

Kullanıcılar, AIR SettingsManager uygulamasını kullanarak otomatik güncelleme özelliğini devre dışı bırakabilir. AIR SettingsManager uygulaması şu adresten indirilebilir: http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air.

Adobe AIR için normal indirme işlemi, http://airinstall.adobe.com adresine bağlanıp işletim sistemi sürümü ve dil gibi indirme ortamı hakkındaki temel bilgileri göndermeyi içerir. Bu bilgiler, her indirmede bir kez gönderilir ve Adobe'nin indirmenin başarılı olduğunu doğrulamasını sağlar. Hiçbir kişisel tanım bilgisi toplanmaz veya gönderilmez.

Sabit çalışma zamanlarını güncelleme

Uygulamanızı sabit bir çalışma zamanı paketiyle dağıtırsanız sabit çalışma zamanı otomatik olarak güncellenmez. Kullanıcılarınızın güvenliği için Adobe tarafından yayınlanan güncellemeleri izlemeli ve ilgili bir güvenlik değişimi yayınlandığında uygulamanızı yeni çalışma zamanı sürümüyle güncellemelisiniz.

Bölüm 3: AIR API'leriyle Çalışma

Adobe® AIR®, Adobe® Flash® Player'da mevcut olmayan işlevler içerir.

ActionScript 3.0 Geliştiricileri
Adobe AIR API'leri aşağıdaki iki kitapta açıklanmıştır:

- ActionScript 3.0 Geliştirici Kılavuzu
- Adobe Flash Platform için ActionScript 3.0 Başvurusu

HTML Geliştiricileri
HTML temelli AIR uygulamaları oluşturuyorsanız, JavaScript'te AIRAliases.js dosyası (bkz. JavaScript'ten AIR API sınıflarına erişme) üzerinden erişebildiğiniz API'ler aşağıdaki iki kitapta açıklanmıştır:

- Adobe AIR için HTML Geliştirici Kılavuzu
- HTML Geliştiricileri için Adobe AIR API Başvurusu

AIR'ye özgü ActionScript 3.0 sınıfları

Aşağıdaki tablo Adobe AIR uygulamasına özgü çalışma zamanı sınıflarını içerir. Tarayıcıda Adobe® Flash® Player'da çalışan SWF içeriği için erişilebilir değildir.

HTML Geliştiricileri
JavaScript'te AIRAliases.js dosyası üzerinden erişebileceğiniz sınıflar HTML Geliştiricileri için Adobe AIR API Başvurusu dosyasında listelenmiştir.

Sınıf	ActionScript 3.0 Paketi	Eklendiği AIR sürümü
ARecord	flash.net.dns	2.0
AAAARecord	flash.net.dns	2.0
ApplicationUpdater	air.update	1.5
ApplicationUpdaterUI	air.update	1.5
AudioPlaybackMode	flash.media	3.0
AutoCapitalize	flash.text	3.0
BrowserInvokeEvent	flash.events	1.0
CameraPosition	flash.media	3.0
CameraRoll	flash.media	2.0
CameraRollBrowseOptions	flash.media	3.0
CameraUI	flash.media	2.5
CertificateStatus	flash.security	2.0

Sınıf	ActionScript 3.0 Paketi	Eklendiği AIR sürümü
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 ve öncesi; 3.7 sürümünden itibaren bırakılmıştır
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 ve öncesi; 3.7 sürümünden itibaren bırakılmıştır
GameInputHand	flash.ui	3.6 ve öncesi; 3.7 sürümünden itibaren bırakılmıştır
Geolocation	flash.sensors	2.0
GeolocationEvent	flash.events	2.0
HTMLHistoryItem	flash.html	1.0
HTMLHost	flash.html	1.0
HTMLLoader	flash.html	1.0
HTMLPDFCapability	flash.html	1.0
HTMLSWFCapabiltiy	flash.html	2.0

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

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

Sınıf	ActionScript 3.0 Paketi	Eklendiği AIR sürümü
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

AIR'ye özgü işlevselliğe sahip Flash Player sınıfları

Aşağıdaki sınıflar tarayıcıda çalışan SWF içeriği tarafından kullanılabilir, ancak AIR ek özellikler veya yöntemler sağlar:

Paket	Sınıf	Özellik, yöntem veya olay	Eklendiği AIR sürümü
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 olayı	2.0
		orientationChanging olayı	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	Sınıf	Özellik, yöntem veya olay	Eklendiği AIR sürümü
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	Sınıf	Özellik, yöntem veya olay	Eklendiği AIR sürümü
flash.net	FileReference	`extension`	1.0
		`httpResponseStatus` olayı	1.0
		`uploadUnencoded()`	1.0
	NetStream	`drmAuthenticate` olayı	1.0
		`onDRMContentData` olayı	1.5
		`preloadEmbeddedData()`	1.5
		`resetDRMVouchers()`	1.0
		`setDRMAuthenticationCredentials()`	1.0
	URLRequest	authenticate	1.0
		cacheResponse	1.0
		followRedirects	1.0
		idleTimeout	2.0
		manageCookies	1.0
		useCache	1.0
		userAgent	1.0
	URLStream	`httpResponseStatus` olayı	1.0
flash.printing	PrintJob	`active`	2.0
		`copies`	2.0
		`firstPage`	2.0
		`isColor`	2.0
		`jobName`	2.0
		`lastPage`	2.0
		`maxPixelsPerInch`	2.0
		`paperArea`	2.0
		`printableArea`	2.0
		`printer`	2.0
		`printers`	2.0
		`selectPaperSize()`	2.0
		`showPageSetupDialog()`	2.0
		`start2()`	2.0
		`supportsPageSetupDialog`	2.0
		`terminate()`	2.0
	PrintJobOptions	`pixelsPerInch`	2.0
		`printMethod`	2.0

Paket	Sınıf	Özellik, yöntem veya olay	Eklendiği AIR sürümü
flash.system	Capabilities	`languages`	1.1
	LoaderContext	`allowLoadBytesCodeExecution`	1.0
	Security	`APPLICATION`	1.0
flash.ui	KeyLocation	D_PAD	2.5

Bu yeni özelliklerin ve yöntemlerin çoğu yalnızca AIR uygulaması güvenlik sanal alanındaki içerik tarafından kullanılabilir. Ancak URLRequest sınıflarındaki yeni üyeler, diğer sanal alanlarda çalışan içerik tarafından da kullanılabilir.

`ByteArray.compress()` ve `ByteArray.uncompress()` yöntemlerinin her biri yeni bir `algorithm` parametresi içerir, böylece deflate ve zlib sıkıştırmaları arasında seçim yapabilirsiniz. Bu parametre yalnızca AIR uygulamasında çalışan içerik tarafından kullanılabilir.

AIR uygulamasına özgü Flex bileşenleri

Aşağıdaki Adobe® Flex™ MX bileşenleri Adobe AIR için içerik geliştirilirken kullanılabilir:

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

Bunlara ek olarak, Flex 4'e şu AIR bileşenleri dahil edilmiştir:

- Window
- WindowedApplication

AIR Flex bileşenleri hakkında daha fazla bilgi için bkz. Flex AIR bileşenlerini kullanma.

Bölüm 4: AIR geliştirme için Adobe Flash Platform araçları

Aşağıdaki Adobe Flash Platform geliştirme araçlarıyla AIR uygulamaları geliştirebilirsiniz.

ActionScript 3.0 (Flash and Flex) geliştiricileri için:

- Adobe Flash Professional (bkz. AIR için yayınlama)
- Adobe Flex 3.x ve 4.x SDK'leri (bkz. "Flex SDK'yi kurma" sayfa 18 ve "AIR Geliştirici Aracı (ADT)" sayfa 163)
- Adobe Flash Builder (bkz. Flash Builder ile AIR Uygulamaları Geliştirme)

HTML ve Ajax geliştiricileri için:

- Adobe AIR SDK (bkz. "AIR SDK'yi yükleme" sayfa 16 ve "AIR Geliştirici Aracı (ADT)" sayfa 163)
- Adobe Dreamweaver CS3, CS4, CS5 (bkz. Dreamweaver için AIR Uzantısı)

AIR SDK'yi yükleme

Adobe AIR SDK uygulamaları başlatmak ve paketlemek için kullandığınız şu komut satırı araçlarını içerir:

AIR Hata Ayıklama Başlatıcısı (ADL) AIR uygulamalarını önce yüklemek zorunda kalmadan çalıştırmanıza olanak tanır. Bkz. "AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157.

AIR Geliştirme Aracı (ADT) AIR uygulamalarını dağıtılabilir yükleme paketleri haline getirir. Bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163.

AIR komut satırı araçları bilgisayarınızda Java yüklenmesini gerektirir. Java sanal makinesini JRE veya JDK'den (1.5 veya sonraki sürümü) kullanabilirsiniz. Java JRE ve Java JDK http://java.sun.com/ adresinde mevcuttur.

ADT aracını çalıştırmak için en az 2 GB bilgisayar belleği gereklidir.

Not: *Son kullanıcıların AIR uygulamaları çalıştırması için Java gerekmez.*

AIR SDK ile AIR uygulaması oluşturmaya hızlı bir genel bakış için bkz. "AIR SDK ile ilk HTML tabanlı AIR uygulamanızı oluşturma" sayfa 31.

AIR SDK'yi indirme ve yükleme

AIR SDK'yi aşağıdaki talimatları kullanarak indirip kurabilirsiniz:

AIR SDK'yi Windows'ta yükleme

- AIR SDK yükleme dosyasını indirin.
- AIR SDK standart bir dosya arşivi olarak dağıtılır. AIR'yi yüklemek için, SDK'nin içeriğini bilgisayarınızdaki bir klasöre çıkarın (örneğin: C:\Program Files\Adobe\AIRSDK veya C:\AIRSDK).
- ADL ve ADT araçları AIR SDK'deki bin klasöründe bulunur; yolu bu klasöre ve PATH ortam değişkenine ekleyin.

AIR SDK'yi Mac OS X'te yükleme

- AIR SDK yükleme dosyasını indirin.

- AIR SDK standart bir dosya arşivi olarak dağıtılır. AIR'yi yüklemek için, SDK'nin içeriğini bilgisayarınızdaki bir klasöre çıkarın (örneğin: /Users/<kullanıcı adı>/Applications/AIRSDK).
- ADL ve ADT araçları AIR SDK'deki bin klasöründe bulunur; yolu bu klasöre ve PATH ortam değişkenine ekleyin.

AIR SDK'yi Linux'ta yükleme

- SDK tbz2 biçiminde mevcuttur.
- SDK'yi yüklemek için, SDK'yi çıkarmak istediğiniz klasörü oluşturun, sonra şu komutu kullanın: tar -jxvf <path to AIR-SDK.tbz2>

AIR SDK araçlarını kullanmaya başlamayla ilgili bilgi için Komut satırı araçlarını kullanarak bir AIR uygulaması oluşturma bölümüne bakın.

AIR SDK'nin içeriği

Aşağıdaki tabloda AIR SDK'de bulunan dosyaların amacı açıklanmaktadır:

SDK klasörü	Dosyalar/araçlar açıklaması
bin	AIR Hata Ayıklama Başlatıcısı (ADL) bir AIR uygulamasını önce paketleyip yüklemeden çalıştırmanıza olanak verir. Bu aracın kullanımıyla ilgili bilgi için bkz. "AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157. AIR Geliştirici Aracı (ADT), uygulamanızı dağıtım için bir AIR dosyası olarak paketler. Bu aracı kullanmayla ilgili bilgi için bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163.
frameworks	Libs dizini AIR uygulamalarında kullanmak için kod kütüphaneleri içerir. Projects dizini derlenen SWF ve SWC kütüphaneleri için kod içerir.
include	Dahil edilen dizin yerel uzantılar yazmaya yönelik C-dili başlık dosyasını içerir.
install	install dizini Android cihazları için Windows USB sürücülerini içerir. (Bunlar Android SDK'de Google tarafından sağlanan sürücülerdir.)
lib	AIR SDK araçları için destek kodu içerir.
runtimes	Masaüstü ve mobil cihazlar için AIR çalışma zamanları. Masaüstü çalışma zamanı ADL tarafından AIR uygulamalarınızın paketlenmeden veya yüklenmeden başlatılması için kullanılır. Android için AIR çalışma zamanları (APK paketleri), geliştirme ve test için Android aygıtlarında veya taklitçilerde yüklenebilir. Aygıtlar ve taklitçiler için ayrı APK paketleri kullanılır. (Genel Android için AIR çalışma zamanı Android Market'ta mevcuttur.)
samples	Bu klasör kesintisiz yükleme özelliğinin örneği olan bir uygulama tanımlayıcı dosyası (badge.swf) ve varsayılan AIR uygulama simgelerini içerir.
templates	descriptor-template.xml - Uygulama açıklayıcı dosyasının her AIR uygulaması için gereken bir şablonu. Uygulama tanımlayıcı dosyasının ayrıntılı açıklaması için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201. AIR'nin her yayın sürümüne yönelik uygulama tanımlayıcısının XML yapısı için şema dosyaları da bu klasörde bulunur.

Flex SDK'yi kurma

Adobe® Flex™ ile Adobe® AIR® uygulamaları oluşturmak için, aşağıdaki seçenekler vardır:

- Adobe AIR projeleri oluşturmak, AIR uygulamalarınıza test ve hata ayıklama yapmak ve onları paketlemek için entegre araçlar sağlayan Adobe® Flash® Builder™ uygulamasını indirip yükleyebilirsiniz. Bkz. "Flash Builder'da ilk masaüstü Flex AIR uygulamanızı oluşturma" sayfa 19.
- Adobe® Flex™ SDK'yi indirip tercih ettiğiniz metin düzenleyicisi ve komut satırı araçlarıyla Flex AIR uygulamaları geliştirebilirsiniz.

Flex SDK ile AIR uygulaması oluşturmaya hızlı bir genel bakış için bkz. "Flex SDK ile ilk masaüstü AIR uygulamanızı oluşturma" sayfa 35.

Flex SDK'yi yükle

Komut satırı araçlarıyla AIR uygulamaları oluşturma işlemi, bilgisayarınızda Java'nın yüklü olmasını gerektirir. Java sanal makinesini JRE veya JDK'den (1.5 veya sonraki sürümü) kullanabilirsiniz. Java JRE ve JDK http://java.sun.com/ adresinde mevcuttur.

Not: *Son kullanıcıların AIR uygulamaları çalıştırması için Java gerekmez.*

Flex SDK, size AIR API'sini ve AIR uygulamalarınızı paketlemek, derlemek ve onların hatalarını ayıklamak için kullandığınız komut satırı araçlarını sağlar.

1. Önceden yapmadıysanız, Flex SDK'yi http://opensource.adobe.com/wiki/display/flexsdk/Downloads adresinden indirin.
2. SDK'nin içeriğini bir klasöre yerleştirin (Örneğin, Flex SDK).
3. AIR SDK'nin içeriğini Flex SDK'nin içindeki dosyaların üzerine kopyalayın.

   Not: *Mac bilgisayarlarda, tek tek dosyaları tüm dizinde değil SDK klasörlerinde kopyaladığınızdan veya değiştirdiğinizden emin olun. Varsayılan olarak, Mac'teki bir dizini aynı adda bir dizine kopyalamak hedef dizindeki mevcut dosyaları kaldırır; iki dizinin içeriğini birleştirmez. AIR SDK ile Flex SDK öğesini birleştirmek için bir terminal penceresinde* `ditto` *komutunu kullanabilirsiniz:* `ditto air_sdk_folder flex_sdk_folder`
4. Komut satırı AIR yardımcı programları bin klasöründe bulunur.

Harici SDK'leri ayarlama

Android ve iOS için uygulama geliştirmek, platform oluşturuculardan ön hazırlık dosyaları, SDK'ler veya diğer geliştirme araçlarını indirmenizi gerektirir.

Android SDK'yi indirme ve yüklemeyle ilgili bilgi için bkz. Android Developers: Installing the SDK (Android Geliştiricileri: SDK'yi yükleme). AIR 2.6'dan itibaren Android SDK'yi indirmeniz gerekmez. AIR SDK artık APK paketlerini yüklemek ve başlatmak için gerekli olan temel bileşenleri içerir. Yine de Android SDK, yazılım taklitçileri oluşturma ve çalıştırma ve aygıt ekran görüntüleri alma dahil çeşitli geliştirme görevleri için çok yararlı olabilir.

iOS geliştirmeleri için harici SDK gerekli değildir. Ancak, özel sertifikalar ve ön hazırlık profilleri gereklidir. Daha fazla bilgi için bkz. Apple'dan geliştirici dosyaları edinme.

Bölüm 5: İlk AIR uygulamanızı oluşturma

Flash Builder'da ilk masaüstü Flex AIR uygulamanızı oluşturma

Adobe® AIR® uygulamasının nasıl çalıştığıyla ilgili hızlı ve etkin bir açıklama için, Adobe® Flash® Builder kullanarak basit bir SWF dosya tabanlı AIR "Hello World" uygulaması oluşturmak ve paketlemek amacıyla bu talimatları izleyin.

Zaten yapmadıysanız, Flash Builder indirip yükleyin. Ayrıca şu konumda bulunan Adobe AIR'nin en son sürümünü indirin ve yükleyin: www.adobe.com/go/air_tr.

Bir AIR projesi oluşturun

Flash Builder, AIR uygulamaları geliştirmek ve paketlemek için ihtiyacınız olan araçları içerir.

Flash Builder veya Flex Builder'da AIR uygulamalarını diğer Flex tabanlı uygulama projelerini oluşturduğunuz şekilde, yeni bir proje tanımlayarak oluşturursunuz.

1. Flash Builder'ı açın.
2. Dosya Seçin > Yeni > Flex Projesi.
3. Proje adını AIRHelloWorld olarak girin.
4. Flex'te AIR uygulamaları uygulama türü varsayılır. İki seçeneğiniz var:
   - Adobe® Flash® Player'da çalışan bir web uygulaması
   - Adobe AIR'de çalışan bir masaüstü uygulaması

   Uygulama türü olarak Masaüstü'nü seçin.
5. Projeyi oluşturmak için Bitir'i tıklatın.

AIR projeleri başta iki dosyadan oluşur: ana MXML dosyası ve (uygulama tanımlayıcısı dosyası olarak bilinen) bir uygulama XML dosyası. Sonraki dosya uygulama özelliklerini belirtir.

Daha fazla bilgi için bkz. Flash Builder ile AIR uygulamaları geliştirme.

AIR uygulama kodunu yazın

"Hello World" uygulama kodunu yazmak için, düzenleyicide açık olan uygulama MXML dosyasını (AIRHelloWorld.mxml) düzenlersiniz. (Dosya açık değilse, dosyayı açmak için Proje Gezgini'ni kullanın.)

Masaüstündeki Flex AIR uygulamaları MXML WindowedApplication etiketine dahildir. MXML WindowedApplication etiketi başlık çubuğu ve kapat düğmesi gibi temel pencere kontrollerini içeren basit bir pencere oluşturur.

1. WindowedApplication bileşenine bir `ba lık` niteliği ekleyin ve onu `"Hello World"` değerine atayın:

```
<?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 Uygulamaya bir Etiket bileşeni ekleyin (onu WindowedApplication etiketinin içine yerleştirin). Etiket bileşeninin `text` özelliğini `"Hello AIR"` değerine ayarlayın ve mizanpaj sınırlamalarını ortaya hizalı tutmak için burada gösterildiği gibi ayarlayın:

```
<?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 Onu izleyen stil bloğunu WindowedApplication etiketini açtıktan hemen sonra ve yeni girdiğiniz etiket bileşen etiketinden önce ekleyin.

```
<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>
```

Bu stil ayarları uygulamanın tamamına uygulanır ve pencere arka planını çok az saydam gri olacak şekilde oluşturur.

Uygulama kodu artık aşağıdaki gibi görünür:

```
<?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>
```

Daha sonra, uygulamanın saydam olmasına izin vermek için uygulama tanımlayıcısında bazı ayarları değiştireceksiniz:

1 Flex Gezgini bölmesinde, uygulama tanımlayıcısı dosyasını projenin kaynak dizininde bulun. Projenizi AIRHelloWorld olarak adlandırdıysanız, bu dosyanın adı AIRHelloWorld-app.xml olur.

2 Uygulama açıklayıcısı dosyasını Flash Builder'da düzenlemek için çift tıklatın.

3 XML kodunda, (`initialWindow` özelliğinin) `systemChrome` ve `transparent` özelliklerine ilişkin yorumlanan satırları bulun. Yorumları kaldırın. (`"<! --"` ve `"-->"` yorum ayırıcıları.)

4 `systemChrome` özelliğinin metin değerini aşağıdaki gibi `none` olarak ayarlayın:

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

5 `transparent` özelliğinin metin değerini aşağıdaki gibi `true` olarak ayarlayın:

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

6 Dosyayı kaydedin.

AIR uygulamasını test edin

Yazdığınız uygulama kodunu test etmek için onu hata ayıklama modunda çalıştırın.

1 Ana araç çubuğundaki Hata Ayıkla düğmesini tıklatın.

Ayrıca Çalıştır > Hata Ayıkla > AIRHelloWorld komutunu da seçebilirsiniz.

Sonuçta elde edilen AIR uygulaması aşağıdaki gibi görünmelidir:

2 Etiket denetiminin `horizontalCenter` ve `verticalCenter` özellikleri kullanılarak metin pencerenin merkezine yerleştirilir. Herhangi başka bir masaüstü uygulamasında yapacağınız şekilde pencereyi taşıyın veya yeniden boyutlandırın.

Not: *Uygulama derlemezse, koda yanlışlıkla girdiğiniz herhangi bir sözdizimi veya yazım hatasını onarın. Hatalar ve uyarılar Flash Builder'ın Sorunlar görünümünde görüntülenir.*

AIR uygulamanızda paketleme, imzalama ve çalıştırma

Artık "Hello World" uygulamasını dağıtım için bir AIR dosyasına paketlemeye hazırsınız. Bir AIR dosyası projenin bin klasöründe bulunan dosyaların tümü olan uygulama dosyalarını içeren bir arşiv dosyasıdır. Bu basit örnekte, bu dosyalar SWF ve uygulama XML dosyalarıdır. AIR paketini daha sonra onu uygulamayı yüklemek için kullanan kullanıcılara dağıtırsınız. Bu işlemde gerekli bir adım onu dijital olarak imzalamaktır.

1 Uygulamada derleme hataları olmadığından ve uygulamanın beklendiği şekilde çalıştığından emin olun.

2 Proje > Sürüm Yapısını Dışa Aktar'ı seçin.

3 AIRHelloWorld projesinin ve AIRHelloWorld.mxml uygulamasının proje ve uygulama için listelenip listelenmediğini kontrol edin.

4 İmzalanmış AIR paketi seçeneği olarak Dışa Aktar'ı seçin. Ardından İleri'yi tıklatın.

5 Mevcut bir dijital sertifikanız varsa, onu bulup seçmek için Gözat'ı tıklatın.

6 Yeni bir kendinden imzalı dijital sertifika oluşturmanız gerekiyorsa, Oluştur'u seçin.

7 Gerekli bilgiyi girin ve Tamam'ı tıklatın.

8 AIRHelloWorld.air adlı AIR paketini oluşturmak için Bitir'i tıklatın.

Artık uygulamayı AIR dosyasını çift tıklatarak Flash Builder'daki Proje Gezgini'nden veya dosya sisteminden yükleyip çalıştırabilirsiniz.

Flash Professional kullanarak ilk masaüstü AIR uygulamanızı oluşturma

Adobe® AIR® uygulamasının nasıl çalıştığını hızlı ve uygulamalı olarak görmek için bu başlık altındaki talimatları takip edin ve Adobe® Flash® Professional uygulamasını kullanarak basit bir "Hello World" AIR uygulaması oluşturun ve paketleyin.

Henüz yapmadıysanız şu konumda bulunan Adobe AIR'i indirin ve yükleyin: www.adobe.com/go/air_tr.

Flash içinde Hello World uygulamasını oluşturma

Flash içinde bir Adobe AIR uygulaması oluşturma, sıradan bir FLA dosyası oluşturmaya benzer. Aşağıdaki adımlar size Flash Professional kullanarak nasıl basit bir Hello World uygulaması oluşturulacağını gösterir.

Hello World uygulamasını oluşturmak için

1 Flash uygulamasını başlatın.

2 Karşılama Ekranı'nda Adobe AIR yayınlama ayarları ile boş bir FLA dosyası oluşturmak için AIR'yi tıklatın.

3 Araçlar panelinde Metin aracını seçin ve Sahne Alanı'nın ortasında statik bir metin alanı (varsayılan) oluşturun. 15 - 20 karakter alacak şekilde genişletin.

4 Metin alanına "Hello World" metnini yazın.

5 Dosyayı bir ad vererek (örneğin HelloAIR) kaydedin.

Uygulamayı test edin

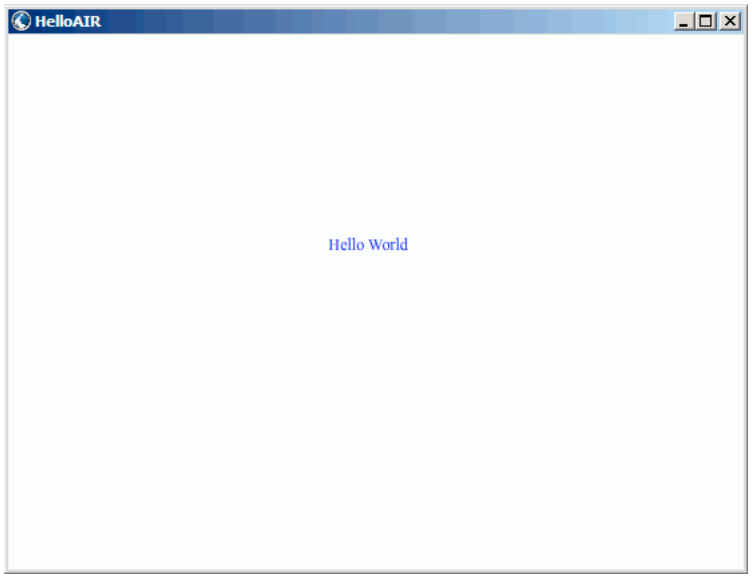
1 Uygulamayı Adobe AIR'de test etmek için Ctrl + Enter tuşlarına basın veya Kontrol Et -> Filmi Test Et -> Test Et öğesini seçin.

2 Film Hatalarını Ayıkla özelliğini kullanmak için ilk olarak uygulamaya ActionScript kodu ekleyin. Aşağıdaki gibi bir izleme ifadesi ekleyerek bunu hızlı bir şekilde deneyebilirsiniz:

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

3 Uygulamayı Film Hatalarını Ayıkla özelliği ile çalıştırmak için Ctrl + Shift + Enter tuşlarına basın veya Hata Ayıkla -> Film Hatalarını Ayıkla -> Hata Ayıkla öğesini seçin.

Hello World uygulaması aşağıdaki resimde gösterildiği gibi görünür:



Uygulamayı paketleme

1. Dosya > Yayınla seçeneğini belirleyin.
2. Adobe AIR paketini mevcut bir dijital sertifikayla imzalayın veya aşağıdaki adımları izleyerek kendi imzalı sertifikanızı oluşturun:
   - **a** Sertifika alanının yanındaki Yeni düğmesini tıklatın.
   - **b** Yayıncı adı, Birim, Kurum adı, E-posta, Ülke, Şifre ve Şifreyi Doğrulayın bilgilerini tamamlayın.
   - **c** Sertifika türünü belirleyin. Sertifika Türü seçeneği güvenlik düzeyini belirlemenizi sağlar: 1024-RSA, 1024-bit anahtar (daha az güvenli) kullanırken 2048-RSA, 2048-bit anahtar kullanır (daha güvenli).
   - **d** Farklı kaydet girişini tamamlayarak veya bir klasör konumuna göz atmak için Gözat... düğmesini tıklatarak bilgiyi bir sertifika dosyası içine kaydedin. (Örneğin *C:/Temp/mycert.pfx*) Kaydetme işlemini tamamladığınızda Tamam düğmesini tıklatın.
   - **e** Flash Dijital İmza İletişim Kutusuna geri döner. Oluşturduğunuz kendinden imzalı sertifikanın yolu ve dosya adı Sertifika metin kutusunda görüntülenir. Görüntülenmiyorsa, yol ve dosya adını girin veya Gözat düğmesini tıklatarak dosyayı bulup seçin.
   - **f** B adımında atadığınız şifrenin aynısını Dijital İmza iletişim kutusunun Şifre metin alanına girin. Adobe AIR uygulamalarınızı imzalama konusunda daha fazla bilgi için bkz. “AIR dosyasını dijital olarak imzalama” sayfa 186.
3. Uygulama ve yükleyici dosyasını oluşturmak için Yayınla düğmesini tıklatın. (Flash CS4 ve CS5'te Tamam düğmesini tıklatın.) AIR dosyasını oluşturmadan önce SWF dosyasını ve application.xml dosyalarını oluşturmak için Filmi Test Et veya Film Hatalarını Ayıkla eylemlerini gerçekleştirmeniz gerekir.
4. Uygulamayı yüklemek için, uygulamanızı kaydettiğiniz klasörün içinde bulunan AIR dosyasını (*application*.air) çift tıklatın.
5. Uygulama Yüklemesi iletişim kutusunda Yükle düğmesini tıklatın.
6. Yükleme Tercihlerini ve Konum ayarlarını inceledikten sonra "Yüklemeden sonra uygulamayı başlat" onay kutusunun işaretli olduğundan emin olun. Ardından Devam düğmesini tıklatın.

7 Yükleme Tamamlandı mesajı görüntülendiğinde Son düğmesini tıklatın.

Flash Professional'da Android için ilk AIR uygulamanızı oluşturma

Android için AIR uygulamaları oluşturmak üzere Adobe Labs bağlantısından Android için Flash Professional CS5 uzantısını indirmeniz gerekir.

Ayrıca şurada açıklandığı gibi Android web sitesinden Android SDK'yi indirip yüklemeniz gerekir: Android Developers: Installing the SDK (Android Geliştiricileri: SDK'yi Yükleme).

Proje oluşturma

1 Flash Professional CS5'i açın

2 Yeni bir AIR for Android projesi oluşturun.

Flash Professional giriş ekranı AIR for Android uygulaması oluşturmak için bir bağlantı içerir. Ayrıca Dosya > Yeni öğesini seçip ardından AIR for Android şablonunu seçebilirsiniz.

3 Belgeyi HelloWorld.fla olarak kaydedin

Kodu yazma

Bu eğitim gerçekten kod yazmayla ilgili olmadığından sahne alanında "Hello, World!" yazmak için Metin aracını kullanın.

Uygulama özelliklerini ayarlama

1 Dosya > AIR Android Ayarları'nı seçin.

2 Genel sekmesinde aşağıdaki ayarları yapın:

- Çıkış Dosyası: HelloWorld.apk
- Uygulama adı: HelloWorld
- Uygulama Kimliği: HelloWorld
- Sürüm numarası: 0.0.1
- En boy oranı: Dikey

3 Dağıtma sekmesinde aşağıdaki ayarları yapın:

- Sertifika: Geçerli bir AIR kod imzalama sertifikasını işaret edin. Yeni bir sertifika oluşturmak için Oluştur düğmesini tıklatabilirsiniz. (Android Marketplace aracılığıyla dağıtılan Android uygulamaları en az 2033 yılına kadar geçerli olan sertifikalara sahip olmalıdır.) Şifre alanına sertifika şifresini girin.
- Android dağıtım türü: Hata Ayıkla
- Yayınladıktan sonra: İki seçeneği de belirleyin
- Android SDK'nin araçlar alt dizininde ADB aracının yolunu girin.

4 Tamam'ı tıklatarak Android ayarları iletişim kutusunu kapatın.

Uygulamanın geliştirmenin bu aşamasında simgelere veya izinlere ihtiyacı yoktur. Android'e yönelik birçok AIR uygulaması korumalı özelliklere erişebilmek için bazı izinler gerektirir. Kullanıcılar uygulamanız çok fazla izin isterse uygulamayı reddedebileceğinden, yalnızca uygulamanızın gerçekten gerektirdiği izinleri ayarlamalısınız.

5 Dosyayı kaydedin.

Android aygıtında uygulamayı Paketleme ve Yükleme

1 Aygıtınızda USB hata ayıklamanın etkin olduğundan emin olun. USB hata ayıklamayı Uygulamalar > Geliştirme altındaki Ayarlar uygulamasından açabilirsiniz.

2 Bir USB kablosuyla aygıtınızı bilgisayara bağlayın.

3 Henüz yapmadıysanız, Android Market'a giderek ve Adobe AIR uygulamasını indirerek AIR çalışma zamanını yükleyin. ("ADT installRuntime komutu" sayfa 175 kullanarak AIR'yi yerel olarak da yükleyebilirsiniz. Android aygıtlar ve taklitçilerde kullanım için Android paketleri AIR SDK'ye dahildir.)

4 Dosya > Yayınla seçeneğini belirleyin.

Flash Professional APK dosyasını oluşturur, uygulamayı bağlı Android aygıtına yükler ve başlatır.

iOS için ilk AIR uygulamanızı oluşturma

AIR 2.6 veya üstü, iOS 4.2 veya üstü

Yalnızca Adobe araçlarını kullanarak bir iOS uygulamasının temel özelliklerini kodlayabilir, oluşturabilir ve test edebilirsiniz. Ancak, bir aygıta iOS uygulaması yüklemek ve uygulamayı dağıtmak için Apple iOS Developer programına (ücretli bir hizmettir) katılmanız gerekir. iOS Developer programına katıldığınızda bir aygıtta test etmek ve ardından dağıtmak üzere uygulama yüklemek için gerekli olan aşağıdaki öğeleri ve dosyaları Apple'dan alabileceğiniz iOS Provisioning Portal sitesine erişebilirsiniz. Bu öğeler ve dosyalar şunları içerir:

- Geliştirme ve dağıtım sertifikaları
- Uygulama Kimlikleri
- Geliştirme ve dağıtım ön hazırlık dosyaları

Uygulama içeriği oluşturma

"Hello world!" metnini görüntüleyen bir SWF dosyası oluşturun Bu görevi Flash Professional, Flash Builder veya başka bir IDE kullanarak gerçekleştirebilirsiniz. Bu örnek metin düzenleyicisi ve Flex SDK'ye dahil olan komut satırı SWF derleyicisini kullanır.

1 Uygulama dosyalarınızı saklamak için uygun bir konumda dizin oluşturun. *HelloWorld.as* adlı bir dosya oluşturun ve dosyayı sık kullandığınız kod düzenleyicide düzenleyin.

2 Aşağıdaki kodu ekleyin:

```
package{

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

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

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

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

3 amxmlc derleyicisini kullanarak sınıfı derleyin:

```
amxmlc HelloWorld.as
```

Aynı klasörde *HelloWorld.swf* adlı bir SWF dosyası oluşturulur.

Not: *Bu örnek, ortam yol değişkeninizi amxmlc'yi barındıran dizini dahil edecek şekilde ayarladığınızı varsayar. Yolu ayarlamayla ilgili bilgi için bkz. "Path ortam değişkenleri" sayfa 300. Alternatif olarak, amxmlc ve bu örnekte kullanılan diğer komut satırı araçlarının tam yolunu yazabilirsiniz.*

Uygulama için simge resmi ve başlangıç ekranı resmi oluşturma

Tüm iOS uygulamaları iTunes uygulamasının kullanıcı arabiriminde ve aygıt ekranında görünen simgelere sahiptir.

1 Proje dizininde bir dizin oluşturarak simgeler adını verin.

2 Simgeler dizininde üç adet PNG dosyası oluşturun. Bunları Icon_29.png, Icon_57.png ve Icon_512.png olarak adlandırın.

3 Uygulamanız için uygun resmi oluşturmak üzere PNG dosyalarını düzenleyin. Dosyalar 29x29 piksel, 57x57 piksel ve 512x512 piksel olmalıdır. Bu test için resim olarak düz renkli kareler kullanabilirsiniz.

Not: *Apple Uygulama Deposuna bir uygulama gönderirken 512 piksellik dosyanın JPG (PNG değil) versiyonunu kullanırsınız. Bir uygulamanın geliştirme sürümlerini test ederken ise PNG versiyonunu kullanırsınız.*

Tüm iPhone uygulamaları yüklenirken bir açılış resmi görüntüler. Açılış resmini bir PNG dosyasında tanımlarsınız:

1 Ana geliştirme dizininde Default.png adlı bir PNG dosyası oluşturun. (Bu dosyayı simgeler altdizinine *koymayın*. Dosyayı baş harfi büyük olacak şekilde Default.png olarak adlandırdığınızdan emin olun.

2 Dosyayı genişliği 320 piksel, yüksekliği 480 piksel olacak şekilde düzenleyin. Şimdilik içerik düz beyaz bir dikdörtgen olabilir. (Bunu daha sonra değiştirirsiniz.)

Bu grafiklerle ilgili ayrıntılı bilgi için bkz. "Uygulama simgeleri" sayfa 85.

Uygulama tanımlayıcı dosyasını oluşturun

Uygulamanın temel özelliklerini belirten bir uygulama tanımlayıcı dosyası oluşturun. Bu görevi Flash Builder veya metin düzenleyicisi gibi bir IDE kullanarak tamamlayabilirsiniz.

1 HelloWorld.as dosyasını içeren proje klasöründe, *HelloWorld-app.xml* adlı bir XML dosyası oluşturun. Bu dosyayı sık kullandığınız XML düzenleyicisinde düzenleyin.

2 Aşağıdaki XML'i ekleyin:

```
<?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>
```

Kolaylık olması açısından, bu örnek yalnızca kullanılabilir özelliklerden birkaçını ayarlar.

Not: *AIR 2 veya öncesini kullanıyorsanız* `<versionNumber>` *öğesi yerine* `<version>` *öğesini kullanmanız gerekir.*

3 Uygulama kimliğini iOS Provisioning Portal'da belirtilen uygulama kimliği ile eşleşecek şekilde değiştirin. (Rastgele çekirdek paketi bölümünü kimliğin başlangıcına dahil etmeyin.

4 Uygulamayı ADL kullanarak test edin:

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

ADL masaüstünüzde şu metni görüntüleyen bir pencere açmalıdır: *Hello World!* Açmıyorsa, hata olup olmadığını görmek için kaynak kodunu ve uygulama tanımlayıcısını kontrol edin.

IPA dosyasını derleyin

Artık ADT kullanarak IPA yükleyici dosyasını derleyebilirsiniz. Apple geliştirici sertifikasına ve P12 dosya biçimde özel anahtara ve Apple geliştirme ön hazırlık profiline sahip olmanız gerekir.

ADT yardımcı programını, anahtar deposunu, depo şifresini ve ön hazırlık profili değerlerini kendi değerlerinizle değiştirerek şu seçeneklerle çalıştırın:

```
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
```

(Tek bir komut satırı kullanın; bu örnekteki satır sonları okumayı kolaylaştırmak için verilmiştir.)

ADT, proje dizininde *HelloWorld.ipa* iOS uygulama yükleyici dosyasını oluşturur. IPA dosyasını derlemek birkaç dakika sürebilir.

Uygulamayı bir aygıta yükleme

Test amacıyla iOS uygulamasını yüklemek için:

1. iTunes uygulamasını açın.
2. Bunu zaten yaptıysanız, bu uygulamanın ön hazırlık profilini iTunes'a ekleyin. iTunes'ta File (Dosya) > Add To Library (Kütüphaneye Ekle) seçimini yapın. Ardından ön hazırlık profilini (dosya türü olarak mobileprovision öğesine sahip olan) seçin.

   Şimdilik, uygulamayı geliştirici aygıtınızda test etmek için, geliştirme ön hazırlık profilini kullanın.

   Daha sonra uygulamayı iTunes Mağazasına dağıtırken, dağıtım profilini kullanın. Uygulamayı geçici olarak (iTunes Mağazasından geçirmeden birden fazla aygıta) dağıtmak için geçici ön hazırlık profilini kullanın.

   Ön hazırlık profilleriyle ilgili daha fazla bilgi için bkz. “iOS kurulumu” sayfa 64.
3. Bazı iTunes sürümleri uygulamanın aynı versiyonu zaten yüklüyse uygulamayı yenisiyle değiştirmez. Bu durumda, uygulamayı aygıttan ve iTunes içerisindeki uygulamalar listesinden silin.
4. Uygulamaya ait IPA dosyasını çift tıklatın. iTunes'da uygulama listesinde görüntülenmelidir.
5. Aygıtı bilgisayarın USB bağlantı noktasına bağlayın.
6. iTunes'ta aygıtın Uygulama sekmesini kontrol edin ve uygulamanın yüklenecek uygulamalar listesinde seçili olduğundan emin olun.
7. iTunes uygulamasının sol taraftaki listesinden aygıtı seçin. Ardından Sync (Senkr) düğmesine basın. Senkr tamamlandığında, Hello World uygulaması iPhone aygıtınızda görünür.

Yeni sürüm yüklü değilse, aygıtınızdan ve iTunes'taki uygulamalar listesinden silin ve ardından bu işlemi yineleyin. O anda yüklü olan sürüm aynı uygulama kimliği ve sürümüne sahipse bu durum söz konusu olabilir.

Başlangıç ekranı grafiğini düzenleme

Uygulamayı derlemenizden önce bir Default.png dosyayı oluşturdunuz (bkz. “Uygulama için simge resmi ve başlangıç ekranı resmi oluşturma” sayfa 26). Bu PNG dosyası, uygulama yüklenirken başlangıç görüntüsü olarak görev yapar. Uygulamayı iPhone aygıtınızda test ederken başlangıçtaki bu boş ekranı fark etmiş olabilirsiniz.

Bu resmi, uygulamanın (“Hello World!”) başlangıç ekranıyla eşleşecek şekilde değiştirmelisiniz:

1. Uygulamayı aygıtınızda açın. İlk “Hello World” metni göründüğünde, Home (Ana Sayfa) düğmesini (ekranın altında) basılı tutun. Home (Ana Sayfa) düğmesini basılı tutarken Power/Sleep (Güç/Uyku) düğmesine basın (iPhone menüsünün en üstünde). Bu bir ekran görüntüsü alır ve bu görüntüyü Film Rulosu'na gönderir.
2. iPhoto veya başka bir fotoğraf aktarma uygulamasından fotoğraf aktararak görüntüyü geliştirme bilgisayarınıza aktarın. (Mac OS'ta Image Capture (Resim Yakalama) uygulamasını da kullanabilirsiniz.)

   Ayrıca fotoğrafı e-posta ile geliştirme bilgisayarınıza gönderebilirsiniz:

   - Fotoğraflar uygulamasını açın.
   - Kamera Kaydını açın.
   - Yakaladığınız ekran görüntüsü resmini açın.
   - Resmi tıklatın ve ardından sol alt köşedeki “ileri” (ok) düğmesini tıklatın. Sonra Email Photo (Fotoğrafı E-posta ile Gönder) seçeneğini tıklatarak resmi kendinize gönderin.

3 Default.png dosyasını (geliştirme dizininde) yakalanan ekran resminin bir PNG sürümüyle değiştirin.

4 Uygulamayı yeniden derleyin (bkz. "IPA dosyasını derleyin" sayfa 27) ve aygıtınıza yeniden yükleyin.

Artık uygulama yüklenirken yeni başlangıç ekranını kullanır.

Not: *Doğru boyutlarda olduğu sürece (320'ye 480 piksel), Default.png dosyası için dilediğiniz resmi oluşturabilirsiniz. Ancak, çoğunlukla ideal olan Default.png resmini uygulamanın ilk haliyle eşleştirmektir.*

Dreamweaver ile ilk HTML tabanlı AIR uygulamanızı oluşturma

Adobe® AIR®'in nasıl çalıştığıyla ilgili hızlı ve etkin bir örnek olarak, Dreamweaver® için Adobe® AIR® Uzantısı'nı kullanarak basit bir HTML tabanlı AIR "Hello World" uygulaması oluşturup paketlemek için bu talimatları kullanın.

Henüz yapmadıysanız şu konumda bulunan Adobe AIR'yi indirin ve yükleyin: www.adobe.com/go/air_tr.

Dreamweaver için Adobe AIR Uzantısı yükleme talimatları için bkz. Dreamweaver için Adobe AIR Uzantısını yükleme.

Sistem gereksinimleri de dahil olmak üzere uzantıya genel bir bakış için bkz. Dreamweaver için AIR Uzantısı.

Not: *HTML tabanlı AIR uygulamaları yalnızca masaüstü ve extendedDesktop profilleri için geliştirilebilir. Mobil profili desteklenemez.*

Uygulama dosyalarını hazırlayın

Adobe AIR uygulamanızın bir başlangıç sayfası olması ve onunla ilgili tüm ilgili sayfaların bir Dreamweaver sitesinde tanımlı olması gerekir:

1 Dreamweaver'ı başlatın ve tanımlı bir siteniz olduğundan emin olun.

2 Dosya > Yeni'yi seçerek yeni bir HTML sayfası açın, Sayfa Türü sütununda HTML'yi seçip Mizanpaj sütununda Yok'u seçin ve Oluştur'u tıklatın.

3 Yeni sayfada, **Hello World!** yazın.

Bu örnek son derece basittir, ancak isterseniz metni istediğiniz şekilde biçimlendirebilir, sayfaya daha fazla içerik ekleyebilir, başka sayfaları bu başlangıç sayfasına bağlayabilir ve bunun gibi bir çok işlem uygulayabilirsiniz.

4 Sayfayı hello_world.html olarak kaydedin (Dosya > Kaydet) Dosyayı bir Dreamweaver sitesinde kaydettiğinizden emin olun.

Dreamweaver siteleriyle ilgili daha fazla bilgi için, bkz. Dreamweaver Yardımı

Adobe AIR uygulamasını oluşturun

1 Dreamweaver Belge penceresinde hello_world.html sayfasının açık olduğundan emin olun. (Oluşturmayla ilgili talimatlar için önceki bölümü inceleyin.)

2 Site Seç > AIR Uygulaması Ayarları.

AIR Uygulaması ve Ayarları iletişim kutusundaki en gerekli ayarlar sizin için otomatik olarak doldurulur. Ancak uygulamanızın ilk içeriğini (veya başlangıç sayfasını) seçmeniz gerekir.

3 İlk İçerik seçeneğinin yanındaki Gözat düğmesini tıklatın, hello_world.html sayfanıza gidin ve onu seçin.

4 Dijital imza seçeneğinin yanındaki Ayarla düğmesini tıklatın.

Bir dijital imza, bir uygulamanın kodunun yazılım yazarı tarafından oluşturulduğu zamandan beri değiştirilmediğinden veya bozulmadığından emin olmayı sağlar ve tüm Adobe AIR uygulamalarında gerekir.

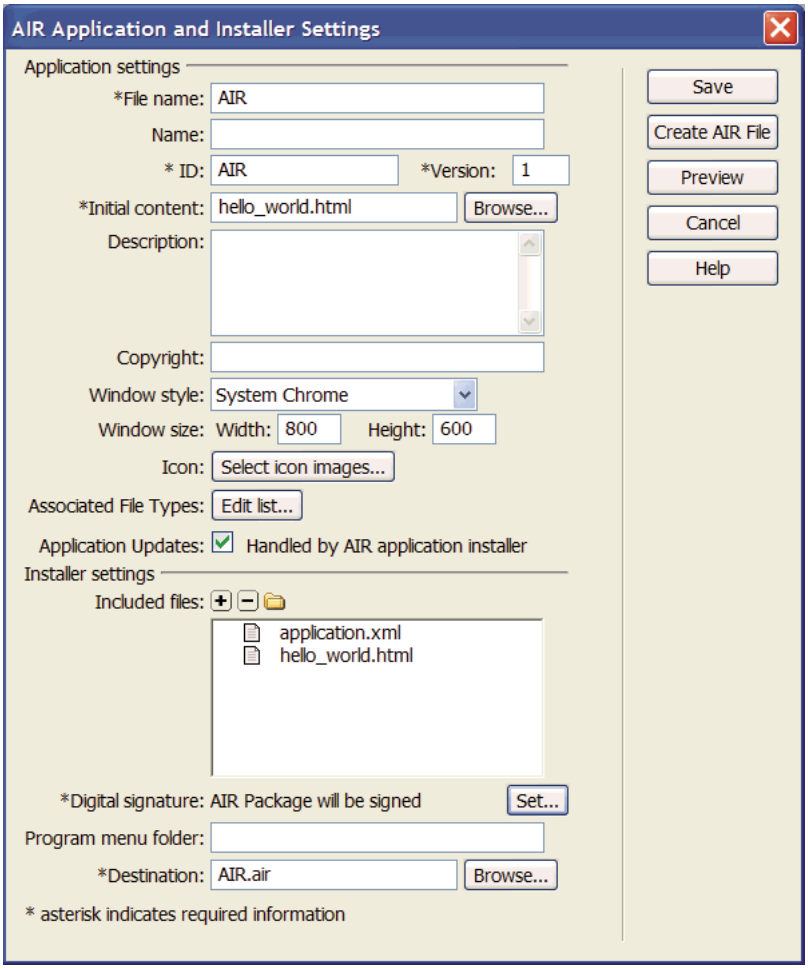
5 Dijital İmza iletişim kutusunda, AIR paketini bir dijital sertifika ile İmzala'yı seçin ve Oluştur düğmesini tıklatın. (Bir dijital imzaya zaten erişiminiz varsa, bunun yerine onu seçmek için Gözat düğmesini tıklatabilirsiniz.)

6 Kendinden İmzalı Dijital Sertifika iletişim kutusunda gereken alanları doldurun. Adınızı girmeniz, bir şifre girmeniz, onu onaylamanız ve dijital sertifika dosyası için bir ad girmeniz gerekecek. Dreamweaver dijital sertifikayı site kökünüze kaydeder.

7 Dijital İmza iletişim kutusuna dönmek için Tamam'ı tıklatın.

8 Dijital İmza iletişim kutusunda, dijital sertifikanız için belirlediğiniz şifreyi girin ve Tamam'ı tıklatın.

AIR Uygulamanız ve Yükleyici Ayarları iletişim kutunuz tamamlandığında böyle görünebilir:



Tüm iletişim kutusu seçenekleri ve onları düzenleme konusunda daha fazla açıklama için bkz. Dreamweaver'da bir AIR uygulaması oluşturma.

9 AIR Dosyasını Oluştur düğmesini tıklatın.

Dreamweaver Adobe AIR uygulama dosyasını oluşturur ve onu sizin site kök klasörünüze kaydeder. Dreamweaver ayrıca bir application.xml dosyası da oluşturur ve onu aynı yere kaydeder. Bu dosya bir açıklama olarak işlev görür, uygulamanın çeşitli özelliklerini tanımlar.

Uygulamayı masaüstüne yükleyin

Artık uygulama dosyasını oluşturduğunuza göre, onu herhangi bir masaüstünde yükleyebilirsiniz.

1 Adobe AIR dosyasını Dreamweaver sitenizin dışına ve masaüstünüze veya başka bir masaüstüne taşıyın.

Bu adım isteğe bağlıdır. Aslında tercih ederseniz bilgisayarınızdaki yeni uygulamayı doğrudan Dreamweaver site dizininden yükleyebilirsiniz.

2 Uygulamayı yüklemek için uygulama çalıştırılabilir dosyasını (.air dosyası) çift tıklatın.

Adobe AIR uygulamasının önizlemesini yapın.

Herhangi bir zaman AIR uygulamalarının parçası olabilecek sayfaların önizlemelerini yapabilirsiniz. Yani, yüklendiğinde nasıl görüneceğini bilmediğiniz uygulamaları paketlemeniz gerekmez.

1 Dreamweaver Belge penceresinde hello_world.html sayfanızın açık olduğundan emin olun.

2 Belge araç çubuğunda, Tarayıcıda Önizleme/Hata Ayıklama düğmesini tıklatın ve sonra AIR'de Önizleme'yi seçin.

Ayrıca (Windows'ta) Ctrl+Shift+F12 veya (Macintosh'ta) Cmd+Shift+F12 tuşlarına basabilirsiniz.

Bu sayfanın önizlemesini yaparken, esasında bir kullanıcının uygulamayı bir masaüstünde yükledikten sonra uygulamanın başlangıç sayfası olarak göreceği öğeyi görürsünüz.

AIR SDK ile ilk HTML tabanlı AIR uygulamanızı oluşturma

Adobe® AIR®'in nasıl çalıştığıyla ilgili hızlı ve etkin bir örnek olarak, basit bir HTML tabanlı AIR "Hello World" uygulaması oluşturup paketlemek için bu talimatları kullanın.

Başlamak için, çalışma zamanını yüklemiş ve AIR SDK'yi kurmuş olmanız gerekir. Bu eğitimde *AIR Hata Ayıklama Başlatıcısı* (ADL) ve *AIR Geliştirme Aracı*'nı (ADT) kullanırsınız. ADL ve ADT komut satırı yardımcı programlardır ve AIR SDK'nin `bin` dizininde bulunabilir (bkz. "AIR SDK'yi yükleme" sayfa 16). Bu eğitim komut satırından program çalıştırmayı ve işletim sisteminiz için gerekli yol ortam değişkenlerinin nasıl kurulacağını bildiğinizi varsayar.

Not: *Adobe® Dreamweaver® kullanıcısıysanız, "Dreamweaver ile ilk HTML tabanlı AIR uygulamanızı oluşturma" sayfa 29 bölümünü okuyun.*

Not: *HTML tabanlı AIR uygulamaları yalnızca masaüstü ve extendedDesktop profilleri için geliştirilebilir. Mobil profili desteklenemez.*

Proje dosyalarını oluştur

Her HTML tabanlı AIR projesi şu iki dosyayı içermelidir: uygulama meta verisini belirleyen bir uygulama tanımlayıcı dosyası ve bir üst düzey HTML sayfası. Bu gerekli dosyalara ek olarak, bu proje AIR API sınıflarına ilişkin uygun takma ad değişkenlerini tanımlayan bir `AIRAliases.js` adlı bir JavaScript kod dosyası içerir.

1 Proje dosyalarını içerecek bir `HelloWorld` dizini oluşturun.

2 `HelloWorld-app.xml` adlı bir XML dosyası oluşturun.

3 `HelloWorld.html` adlı bir HTML dosyası oluşturun.

4 `AIRAliases.js` dosyasını AIR SDK'nin çerçeveler klasöründen proje dizinine kopyalayın.

AIR uygulama tanımlayıcı dosyasını oluşturma

AIR uygulamanızı oluşturmaya başlarken, aşağıdaki yapıda bir XML uygulama tanımlayıcı dosyası oluşturun.

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

1 Düzenlenecek `HelloWorld-app.xml` dosyasını açın.

2 AIR ad alanı niteliğini dahil ederek kök `<application>` öğesini ekleyin.

<application xmlns="http://ns.adobe.com/air/application/2.7"> Ad alanının son bölmesi olan "2.7", uygulama için gerekli çalışma zamanının sürümünü belirtir.

3 `<id>` öğesini ekleyin:

<id>examples.html.HelloWorld</id> Uygulama kimliği uygulamanızı benzersiz şekilde (AIR'nin uygulama paketini imzalamak için kullanılan sertifikadan türettiği) yayıncı kimliğiyle birlikte tanımlar. Uygulama kimliği; yükleme, özel uygulama dosya sistemi deposu dizinine erişim, özel şifrelenmiş depoya erişim ve uygulamalar arası iletişim için kullanılır.

4 `<versionNumber>` öğesini ekleyin:

<versionNumber>0.1</versionNumber> Kullanıcıların uygulamanızın hangi sürümünü yüklediklerini belirlemelerine yardımcı olur.

Not: *AIR 2 veya öncesini kullanıyorsanız* `<versionNumber>` *öğesi yerine* `<version>` *öğesini kullanmanız gerekir.*

5 `<filename>` öğesini ekleyin:

<filename>HelloWorld</filename> İşletim sisteminde uygulama çalıştırılabilir dosyası, yükleme dizini ve diğer uygulama başvuruları için kullanılan ad.

6 İlk uygulama pencerenizin özelliklerini belirlemek için aşağıdaki alt öğeleri içeren `<initialWindow>` öğesini ekleyin.

<content>HelloWorld.html</content> AIR'in yükleyeceği kök HTML dosyasını tanımlar.

<visible>true</visible> Pencereyi hemen görünür hale getirir.

<width>400</width> Pencere genişliğini ayarlar (piksel biriminde).

<height>200</height> Pencere yüksekliğini ayarlar.

7 Dosyayı kaydedin. Tamamlanan uygulama tanımlayıcısı dosyasının aşağıdaki gibi görünmesi gerekir:

```
<?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>
```

Bu örnek olabilecek uygulama özelliklerinin yalnızca birkaçını ayarlar. Pencere kromu, pencere boyutu, saydamlık, varsayılan yükleme dizini, ilişkilendirilmiş dosya türleri ve uygulama simgeleri gibi öğeleri belirlemenize olanak veren tam bir uygulama özellikleri kümesi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201.

Uygulama HTML sayfasını oluşturun

Artık AIR uygulamanız için ana dosya olarak işlev görecek basit bir HTML sayfası oluşturmanız gerekir.

1 Düzenlenecek `HelloWorld.html` dosyasını açın. Aşağıdaki HTML kodunu ekleyin:

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

2 HTML'nin `<head>` bölmesinde, `AIRAliases.js` dosyasını içe aktarın:

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

AIR HTML window nesnesinde `runtime` adlı bir özellik tanımlar. Çalışma zamanı özelliği sınıfın tam nitelikli paket adını kullanarak yerleşik AIR sınıfına erişim sağlar. Örneğin, bir AIR File nesnesi oluşturmak için JavaScript'e aşağıdaki cümleyi ekleyebilirsiniz:

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

`AIRAliases.js` dosyası en kullanışlı AIR API'lerine ilişkin pratik adları tanımlar. `AIRAliases.js` dosyasını kullanarak File sınıfının başvurusunu aşağıdaki gibi olacak şekilde kısaltabilirsiniz:

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

3 AIRAliases komut dosyası etiketinin altında, `onLoad` olayını işlemek için bir JavaScript işlevi içeren başka bir komut dosyası etiketi ekleyin:

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

`appLoad()` işlevi sadece `air.trace()` işlevini çağırır. ADL kullanarak uygulamayı çalıştırdığınızda, izleme mesajı komut konsoluna yazdırılır. İzleme ifadeleri hata ayıklamak için çok kullanışlı olabilir.

4 Dosyayı kaydedin.

`HelloWorld.html` dosyanızın aşağıdaki gibi görünmesi gerekir:

```
<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>
```

Uygulamayı test edin

Uygulamayı komut satırından çalıştırmak ve test etmek için, AIR Hata Ayıklama Başlatıcısı (ADL) yardımcı programı kullanın. FDB programı, AIR SDK'nin `bin` dizininde bulunabilir. Zaten AIR SDK'yi yüklemediyseniz "AIR SDK'yi yükleme" sayfa 16 bölümüne bakın.

1. Bir komut konsolu veya bir kabuk açın. Bu proje için oluşturduğunuz dizini değiştirin.
2. Aşağıdaki komutu çalıştırın:

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

   Bir AIR penceresi uygulamanızı görüntüleyerek açılır. Konsol penceresi ayrıca `air.trace()` çağrısının sonucunda meydana gelen mesajı görüntüler.

   Daha fazla bilgi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201.

AIR yükleme dosyasını oluşturma

Uygulamanız başarıyla çalıştığında, uygulamayı bir AIR yükleme dosyasına paketlemek için ADT yardımcı programı kullanabilirsiniz. Bir AIR yükleme dosyası kullanıcılarınıza dağıtabileceğiniz tüm uygulama dosyalarını içeren bir arşiv dosyasıdır. Bir paketlenmiş AIR dosyası yüklemeden önce Adobe AIR yüklemeniz gerekir.

Uygulama güvenliğini kesinleştirmek için, tüm AIR yükleme dosyaları dijital olarak imzalanmalıdır. Geliştirme nedenlerinden dolayı, ADT veya başka bir sertifika oluşturma aracıyla temel, kendinden imzalı bir sertifika oluşturabilirsiniz. Ayrıca VeriSign veya Thawte gibi bir ticari sertifika yetkilisinden ticari bir kod imzalama sertifikası da alabilirsiniz. Kullanıcılar bir kendinden imzalı AIR dosyası yüklediklerinde, yayıncı yükleme işlemi sırasında "bilinmeyen" olarak görüntülenir. Bu bir kendinden imzalı sertifikanın yalnızca AIR dosyasının oluşturulduğundan beri değiştirilmediğini garantilemesinden kaynaklanır. Herhangi birinin bir sahte AIR dosyasını kendisi imzalayıp onu sizin uygulamanızmış gibi sunmasını engelleyecek herhangi bir durum yoktur. Genel olarak yayınlanan AIR dosyaları için, doğrulanabilir ve ticari bir sertifika tavsiye edilir. AIR güvenlik sorunlarına genel bir bakış için bkz. AIR güvenliği (ActionScript geliştiricileri için) veya AIR güvenliği (HTML geliştiricileri için).

Bir kendinden imzalı sertifika ve bir anahtar çifti oluşturun.

❖ Komut isteminde, aşağıdaki komutu girin (ADT çalıştırılabilir dosyası AIR SDK'nin `bin` dizinindedir):

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

ADT bir sertifika ve ilişkili bir özel anahtar içeren *sampleCert.pfx* adlı bir anahtar deposu oluşturur.

Bu örnek bir sertifika için ayarlanabilecek minimum sayıda niteliği kullanır: Anahtar türü *1024-RSA* veya *2048-RSA* olmalıdır (bkz. "AIR uygulamalarını imzalama" sayfa 186).

AIR yükleme dosyasını oluşturma

❖ Komut istemcisinde, aşağıdaki komutu girin (tek satır halinde):

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

Sizden anahtar deposu dosya şifresi istenecek.

HelloWorld.air argümanı ADT'nin oluşturduğu bir AIR dosyasıdır. HelloWorld-app.xml uygulama tanımlayıcı dosyasıdır. Sonraki argümanlar uygulamanız tarafından kullanılan dosyalardır. Bu örnek yalnızca iki dosya kullanır, ancak herhangi sayıda dosya ve dizin dahil edebilirsiniz. ADT ana içerik dosyası HelloWorld.html öğesinin pakette olduğunu doğrular, ancak AIRAliases.js dosyasını koymayı unutursanız uygulamanız çalışmaz.

AIR paketi oluşturulduktan sonra, paket dosyasını çift tıklatarak uygulamayı yükleyip çalıştırabilirsiniz. Ayrıca AIR dosya adını kabukta veya komut penceresinde bir komut olarak da yazabilirsiniz.

Sonraki Adımlar

AIR'de, HTML ve JavaScript kodu genelde tipik bir web tarayıcısında davranacağı davranır. (Aslında, AIR Safari web tarayıcısıyla aynı WebKit oluşturma motorunu kullanır.) Ancak, AIR'de HTML uygulamaları geliştirirken anlamanız gereken bazı önemli farklar vardır. Bu farklar ve diğer önemli konularla ilgili daha fazla bilgi için bkz. HTML ve JavaScript'i Programlama.

Flex SDK ile ilk masaüstü AIR uygulamanızı oluşturma

Adobe® AIR® uygulamasının nasıl çalıştığıyla ilgili hızlı ve etkin bir örnek olarak, Flex SDK'yi kullanarak basit bir SWF tabanlı AIR "Hello World" uygulaması oluşturmak için bu talimatları kullanın. Bu eğitim Flex SDK (Flex SDK AIR SDK'yi içerir) ile sağlanan komut satırı araçlarını kullanarak bir AIR uygulamasının nasıl derleneceğini, test edileceğini ve paketleneceğini gösterir.

Başlamak için, çalışma zamanını yüklemiş ve Adobe® Flex™ uygulamasını kurmuş olmanız gerekir. Bu eğitim *AMXMLC* derleyicisini, *AIR Hata Ayıklama Başlatıcısı*'nı (ADL) ve *AIR Geliştiricisi Aracı*'nı (ADT) kullanır. Bu programlar Flex SDK'nin `bin` dizininde bulunur (bkz. "Flex SDK'yi kurma" sayfa 18).

AIR uygulama tanımlayıcı dosyasını oluşturma

Bu bölüm bir XML dosyası olan uygulama açıklayıcısının aşağıdaki yapı ile nasıl oluşturulacağını açıklar:

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

1 `HelloWorld-app.xml` adlı bir XML dosyası oluşturup bu dosyayı proje dizinine kaydedin.

2 AIR ad alanı niteliğini dahil ederek `<application>` öğesini ekleyin.

<application xmlns="http://ns.adobe.com/air/application/2.7"> Ad alanının son bölmesi olan "2.7," uygulama için gerekli çalışma zamanının sürümünü belirtir.

3 `<id>` öğesini ekleyin:

<id>samples.flex.HelloWorld</id> Uygulama kimliği uygulamanızı benzersiz şekilde (AIR'nin uygulama paketini imzalamak için kullanılan sertifikadan türettiği) yayıncı kimliğiyle birlikte tanımlar. Tavsiye edilen form, `"com.company.AppName"` gibi nokta sınırlı ve ters DNS stili olan bir dizedir. Uygulama kimliği; yükleme, özel uygulama dosya sistemi deposu dizinine erişim, özel şifrelenmiş depoya erişim ve uygulamalar arası iletişim için kullanılır.

4 `<versionNumber>` öğesini ekleyin:

<versionNumber>1.0</versionNumber> Kullanıcıların uygulamanızın hangi sürümünü yüklediklerini belirlemelerine yardımcı olur.

***Not:** AIR 2 veya öncesini kullanıyorsanız `<versionNumber>` öğesi yerine `<version>` öğesini kullanmanız gerekir.*

5 `<filename>` öğesini ekleyin:

<filename>HelloWorld</filename> İşletim sistemindeki başvurularda uygulama çalıştırılabilir dosyası, yükleme dizini ve benzerleri için kullanılan ad.

6 İlk uygulama pencerenizin özelliklerini belirlemek için aşağıdaki alt öğeleri içeren `<initialWindow>` öğesini ekleyin:

<content>HelloWorld.swf</content> AIR'nin yüklemesi için kök SWF dosyasını tanımlar.

<visible>true</visible> Pencereyi hemen görünür hale getirir.

<width>400</width> Pencere genişliğini ayarlar (piksel biriminde).

<height>200</height> Pencere yüksekliğini ayarlar.

7 Dosyayı kaydedin. Tamamlanmış uygulama açıklayıcısı dosyanız aşağıdaki gibi görünmelidir:

```
<?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>
```

Bu örnek olabilecek uygulama özelliklerinin yalnızca birkaçını ayarlar. Pencere kromu, pencere boyutu, saydamlık, varsayılan yükleme dizini, ilişkilendirilmiş dosya türleri ve uygulama simgeleri gibi öğeleri belirlemenize olanak veren tam bir uygulama özellikleri kümesi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201

Uygulama kodunu yazma

Not: *SWF tabanlı AIR uygulamaları MXML veya Adobe® ActionScript® 3.0 ile tanımlanan bir ana sınıf kullanabilir. Bu örnek, ana sınıfını tanımlamak için bir MXML dosyası kullanır. Ana ActionScript sınıfı olan bir AIR uygulaması oluşturma süreci buna benzerdir. SWF dosyasına bir MXML dosyası derlemek yerine, ActionScript sınıf dosyasını derlersiniz. ActionScript kullanırken, ana sınıf flash.display.Sprite öğesini genişletmelidir.*

Tüm Flex tabanlı uygulamalar gibi, Flex çerçevesiyle oluşturulan AIR uygulamaları bir ana MXML dosyası içerir. Masaüstü AIR uygulamaları, kök öğe olarak Application bileşeni yerine *WindowedApplication* bileşenini kullanır. WindowedApplication bileşeni, uygulamanızın ve ilk penceresinin kontrolü için özellikler, yöntemler ve olaylar sağlar. AIR'nin birden fazla pencereyi desteklemediği platformlar ve profiller için Application bileşenini kullanmaya devam edin. Mobil Flex uygulamalarında View veya TabbedViewNavigatorApplication bileşenlerini de kullanabilirsiniz.

Aşağıdaki yordam Hello World uygulamasını oluşturur:

1. Bir metin düzenleyicisi kullanarak `HelloWorld.mxml` adlı bir dosya oluşturun ve aşağıdaki MXML kodunu ekleyin:

   ```
   <?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. Sonra, uygulamaya bir Etiket bileşeni ekleyin (bunu WindowedApplication etiketinin içine yerleştirin).
3. Etiket bileşeninin `text` özelliğini `"Hello AIR"` olarak ayarlayın.
4. Mizanpaj sınırlamalarını her zaman ortada tutulacak şekilde ayarlayın.

   Aşağıdaki örnek şimdiye kadar oluşan kodu gösterir:

   ```
   <?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>
   ```

Uygulamayı derleme

Uygulamayı çalıştırıp hatalarını ayıklamadan önce, MXML kodunu amxmlc derleyicisini kullanarak bir SWF dosyasına derleyin. amxmlc derleyicisi, Flex SDK'nin `bin` dizininde bulunabilir. İsterseniz, bilgisayarınızın yol ortamını Flex SDK bin dizinini içerecek şekilde ayarlayabilirsiniz. Yolun ayarlanması komut satırındaki hizmet programlarının daha kolay çalıştırılmasını sağlar.

1 AIR uygulamanızın proje klasörüne gitmek için bir komut kabuğu veya terminal açın.

2 Aşağıdaki komutu girin:

```
amxmlc HelloWorld.mxml
```

`amxmlc` öğesinin çalıştırılması uygulamanın derlenmiş kodunu içeren `HelloWorld.swf` dosyasını oluşturur.

Not: *Uygulama derlenmezse, sözdizimi veya yazım hatalarını onarın. Hatalar ve uyarılar amxmlc derleyicisini çalıştırmak için kullanılan konsol penceresinde görüntülenir.*

Daha fazla bilgi için, bkz. "MXML ve ActionScript dosyalarını AIR için derleme" sayfa 154.

Uygulamayı test edin

Uygulamayı komut satırından çalıştırmak ve test etmek için, uygulamayı uygulama açıklayıcısı dosyasını kullanarak başlatmak amacıyla AIR Hata Ayıklama Başlatıcısı'nı (ADL) kullanın. (ADL, Flex SDK'nin bin dizininde bulunabilir.)

❖ Komut istemcisinde aşağıdaki komutu girin:

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

Sonuç olarak ortaya çıkan AIR uygulaması şu şekildeki gibi görünür:

Etiket denetiminin horizontalCenter ve verticalCenter özellikleri kullanılarak metin, pencerenin merkezine yerleştirilir. Herhangi başka bir masaüstü uygulamasında yapacağınız şekilde pencereyi taşıyın veya yeniden boyutlandırın.

Daha fazla bilgi için bkz. "AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157.

AIR yükleme dosyasını oluşturma

Uygulamanız başarıyla çalıştığında, uygulamayı bir AIR yükleme dosyasına paketlemek için ADT yardımcı programı kullanabilirsiniz. Bir AIR yükleme dosyası kullanıcılarınıza dağıtabileceğiniz tüm uygulama dosyalarını içeren bir arşiv dosyasıdır. Bir paketlenmiş AIR dosyası yüklemeden önce Adobe AIR yüklemeniz gerekir.

Uygulama güvenliğini kesinleştirmek için, tüm AIR yükleme dosyaları dijital olarak imzalanmalıdır. Geliştirme nedenlerinden dolayı, ADT veya başka bir sertifika oluşturma aracıyla temel, kendinden imzalı bir sertifika oluşturabilirsiniz. Ayrıca, ticari bir sertifika yetkilisinden ticari bir kod imzalama sertifikası da satın alabilirsiniz. Kullanıcılar bir kendinden imzalı AIR dosyası yüklediklerinde, yayıncı yükleme işlemi sırasında "bilinmeyen" olarak görüntülenir. Bu bir kendinden imzalı sertifikanın yalnızca AIR dosyasının oluşturulduğundan beri değiştirilmediğini garantilemesinden kaynaklanır. Herhangi birinin bir sahte AIR dosyasını kendisi imzalayıp onu sizin uygulamanızmış gibi sunmasını engelleyecek herhangi bir durum yoktur. Genel olarak yayınlanan AIR dosyaları için, doğrulanabilir ve ticari bir sertifika tavsiye edilir. AIR güvenlik sorunlarına genel bir bakış için bkz. AIR güvenliği (ActionScript geliştiricileri için) veya AIR güvenliği (HTML geliştiricileri için).

Bir kendinden imzalı sertifika ve bir anahtar çifti oluşturun.

❖ Komut isteminde, aşağıdaki komutu girin (ADT çalıştırılabilir dosyası Flex SDK'nin `bin` dizininde bulunabilir):

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

Bu örnek bir sertifika için ayarlanabilecek minimum sayıda niteliği kullanır: Anahtar türü *1024-RSA* veya *2048-RSA* olmalıdır (bkz. “AIR uygulamalarını imzalama” sayfa 186).

AIR paketini oluşturma

❖ Komut istemcisinde, aşağıdaki komutu girin (tek satır halinde):

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

Sizden anahtar deposu dosya şifresi istenecek. Şifreyi girin ve Enter tuşuna basın. Şifre karakterleri güvenlik nedeniyle görüntülenmez.

HelloWorld.air argümanı ADT'nin oluşturduğu bir AIR dosyasıdır. HelloWorld-app.xml uygulama tanımlayıcı dosyasıdır. Sonraki argümanlar uygulamanız tarafından kullanılan dosyalardır. Bu örnek yalnızca üç dosya kullanır, ancak siz herhangi sayıda dosya ve dizin dahil edebilirsiniz.

AIR paketi oluşturulduktan sonra, paket dosyasını çift tıklatarak uygulamayı yükleyip çalıştırabilirsiniz. Ayrıca AIR dosya adını kabukta veya komut penceresinde bir komut olarak da yazabilirsiniz.

Daha fazla bilgi için bkz. “Bir masaüstü AIR yükleme dosyasını paketleme” sayfa 51.

Flex SDK ile Android için ilk AIR uygulamanızı oluşturma

Başlamak için AIR ve Flex SDK'lerini yüklemiş ve ayarlamış olmanız gerekir. Bu eğitim Flex SDK'den *AMXMLC* derleyicisini ve *AIR Hata Ayıklama Başlatıcısı*'nı (ADL) ve AIR SDK'den *AIR Geliştirici Aracı*'nı (ADT) kullanır. Bkz. “Flex SDK’yi kurma” sayfa 18.

Ayrıca şurada açıklandığı gibi Android web sitesinden Android SDK'yi indirip yüklemeniz gerekir: Android Developers: Installing the SDK (Android Geliştiricileri: SDK'yi Yükleme).

Not: *iPhone geliştirmesiyle ilgili bilgi için bkz. Flash Professional CS5 ile bir Hello World iPhone uygulaması oluşturma.*

AIR uygulama tanımlayıcı dosyasını oluşturma

Bu bölüm bir XML dosyası olan uygulama açıklayıcısının aşağıdaki yapı ile nasıl oluşturulacağını açıklar:

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

1 `HelloWorld-app.xml` adlı bir XML dosyası oluşturup bu dosyayı proje dizinine kaydedin.

2 AIR ad alanı niteliğini dahil ederek `<application>` öğesini ekleyin.

<application xmlns="http://ns.adobe.com/air/application/2.7"> Ad alanının son bölmesi olan "2.7," uygulama için gerekli çalışma zamanının sürümünü belirtir.

3 `<id>` öğesini ekleyin:

<id>samples.android.HelloWorld</id> Uygulama kimliği uygulamanızı benzersiz şekilde yayıncı kimliğiyle (AIR'nin uygulama paketini imzalamak için kullanılan sertifikadan türettiği) birlikte tanımlar. Tavsiye edilen form, `"com.company.AppName"` gibi nokta sınırlı ve ters DNS stili olan bir dizedir.

4 `<versionNumber>` öğesini ekleyin:

<versionNumber>0.0.1</versionNumber> Kullanıcıların uygulamanızın hangi sürümünü yüklediklerini belirlemelerine yardımcı olur.

5 `<filename>` öğesini ekleyin:

<filename>HelloWorld</filename> İşletim sistemindeki başvurularda uygulama çalıştırılabilir dosyası, yükleme dizini ve benzerleri için kullanılan ad.

6 İlk uygulama pencerenizin özelliklerini belirlemek için aşağıdaki alt öğeleri içeren `<initialWindow>` öğesini ekleyin:

<content>HelloWorld.html</content> AIR'nin yükleyeceği kök HTML dosyasını tanımlar.

7 `<supportedProfiles>` öğesini ekleyin.

<supportedProfiles>mobileDevice</supportedProfiles> Uygulamanın yalnızca mobil profilinde çalıştığını belirtir.

8 Dosyayı kaydedin. Tamamlanmış uygulama açıklayıcısı dosyanız aşağıdaki gibi görünmelidir:

```
<?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>
```

Bu örnek olabilecek uygulama özelliklerinin yalnızca birkaçını ayarlar. Uygulama tanımlayıcı dosyasında kullanabileceğiniz başka ayarlar da vardır. Örneğin, tam ekranlı bir uygulama oluşturmak için initialWindow öğesine <fullScreen>true</fullScreen> öğesini ekleyebilirsiniz. Android'de uzaktan hata ayıklamayı ve erişimi kontrol edilen özellikleri etkinleştirmek için ayrıca uygulama tanımlayıcısına Android izinleri eklemeniz gerekir. Bu basit uygulama için izinlere gerek yoktur. Bu nedenle şu anda eklemeniz gerekmez.

Daha fazla bilgi için bkz. “Mobil uygulama özelliklerini ayarlama” sayfa 68.

Uygulama kodunu yazma

HelloWorld.as adlı bir dosya oluşturun ve bir metin düzenleyicisi kullanarak aşağıdaki kodu ekleyin:

```
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 );
        }
    }
}
```

Uygulamayı derleme

Uygulamayı çalıştırıp hatalarını ayıklamadan önce, MXML kodunu amxmlc derleyicisini kullanarak bir SWF dosyasına derleyin. amxmlc derleyicisi, Flex SDK'nin `bin` dizininde bulunabilir. İsterseniz, bilgisayarınızın yol ortamını Flex SDK bin dizinini içerecek şekilde ayarlayabilirsiniz. Yolun ayarlanması komut satırındaki hizmet programlarının daha kolay çalıştırılmasını sağlar.

1. AIR uygulamanızın proje klasörüne gitmek için bir komut kabuğu veya terminal açın.
2. Aşağıdaki komutu girin:

   ```
   amxmlc HelloWorld.as
   ```

`amxmlc` öğesinin çalıştırılması uygulamanın derlenmiş kodunu içeren `HelloWorld.swf` dosyasını oluşturur.

Not: *Uygulama derlenmezse, sözdizimi veya yazım hatalarını onarın. Hatalar ve uyarılar amxmlc derleyicisini çalıştırmak için kullanılan konsol penceresinde görüntülenir.*

Daha fazla bilgi için, bkz. “MXML ve ActionScript dosyalarını AIR için derleme” sayfa 154.

Uygulamayı test edin

Uygulamayı komut satırından çalıştırmak ve test etmek için, uygulamayı uygulama açıklayıcısı dosyasını kullanarak başlatmak amacıyla AIR Hata Ayıklama Başlatıcısı'nı (ADL) kullanın. (ADL, AIR ve Flex SDK'lerinin bin dizininde bulunabilir.)

- Komut istemcisinde aşağıdaki komutu girin:

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

  Daha fazla bilgi için bkz. “ADL kullanarak aygıt benzetimi” sayfa 98.

APK paket dosyası oluşturma

Uygulamanız başarıyla çalıştığında, uygulamayı bir APK paket dosyasına paketlemek için ADT yardımcı programı kullanabilirsiniz. Bir APK paket dosyası, kullanıcılarınıza dağıtabileceğiniz yerel Android uygulama dosyası biçimidir.

Tüm Android uygulamaları imzalı olmalıdır. AIR dosyalarının aksine, Android uygulamalarını kendinden imzalı bir sertifikayla imzalamak gerekir. Android işletim sistemi uygulama geliştiricisinin kimliğini oluşturmayı denemez. Android paketlerini imzalamak için ADT tarafından oluşturulmuş bir sertifika kullanabilirsiniz. Android market'a gönderilen uygulamalar için kullanılan sertifikalar en az 25 yıllık bir geçerlilik süresine sahip olmalıdır.

Bir kendinden imzalı sertifika ve bir anahtar çifti oluşturun.

❖ Komut isteminde, aşağıdaki komutu girin (ADT çalıştırılabilir dosyası Flex SDK'nin `bin` dizininde bulunabilir):

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

Bu örnek bir sertifika için ayarlanabilecek minimum sayıda niteliği kullanır: Anahtar türü *1024-RSA* veya *2048-RSA* olmalıdır (bkz. “ADT certificate komutu” sayfa 172).

AIR paketini oluşturma

❖ Komut istemcisinde, aşağıdaki komutu girin (tek satır halinde):

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

Sizden anahtar deposu dosya şifresi istenecek. Şifreyi girin ve Enter tuşuna basın.

Daha fazla bilgi için bkz. “Mobil AIR uygulamasını paketleme” sayfa 91.

AIR çalışma zamanını yükleme

Android Market'tan aygıtınıza AIR çalışma zamanının en son sürümünü yükleyebilirsiniz. Ayrıca SDK'nizde bulunan çalışma zamanını da bir aygıta veya Android taklitçisine yükleyebilirsiniz.

❖ Komut istemcisinde, aşağıdaki komutu girin (tek satır halinde):

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

Android SDK dizininizde `-platformsdk` bayrağını ayarlayın (araçlar klasörünün üst öğesini belirtin).

ADT, SDK içinde bulunan Runtime.apk dosyasını yükler.

Daha fazla bilgi için bkz. “AIR çalışma zamanını ve geliştirme uygulamalarını yükleme” sayfa 106.

AIR uygulamasını yükleme

❖ Komut istemcisinde, aşağıdaki komutu girin (tek satır halinde):

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

Android SDK dizininizde `-platformsdk` bayrağını ayarlayın (araçlar klasörünün üst öğesini belirtin).

Daha fazla bilgi için bkz. “AIR çalışma zamanını ve geliştirme uygulamalarını yükleme” sayfa 106.

Aygıtın veya taklitçinin ekranındaki uygulama simgesine dokunarak uygulamanızı başlatabilirsiniz.

Bölüm 6: Masaüstü için AIR uygulamaları geliştirme

Masaüstü bir AIR uygulaması geliştirmek için iş akışı

Bir AIR uygulaması geliştirmek için temel iş akışı birçok geleneksel geliştirme modeliyle aynıdır: kodla, derle, test et ve döngünün sonuna doğru bir yükleyici dosyasına paketle.

Uygulama kodunu Flash, Flex ve ActionScript kullanarak yazabilir ve Flash Professional, Flash Builder veya mxmlc ve compc komut satırı derleyicilerini kullanarak derleyebilirsiniz. Ayrıca, HTML ve JavaScript kullanarak uygulama kodunu yazabilir ve derleme adımını atlayabilirsiniz.

Masaüstü AIR uygulamalarını, bir uygulamayı önce paketlenmesini veya yüklenmesini gerektirmeden çalıştıran ADL aracı ile test edebilirsiniz. Flash Professional, Flash Builder, Dreamweaver ve Aptana IDE Flash hata ayıklayıcısıyla entegre olur. Ayrıca komut satırından ADL'yi kullanırken manuel olarak FDB hata ayıklama aracını da başlatabilirsiniz. ADL'nin kendisi hataları görüntüler ve ifade çıktısını izler.

Tüm AIR uygulamaları bir yükleme dosyasına paketlenmelidir. Platformlar arası AIR dosya biçimi şu koşulların olmaması durumunda önerilir:

- NativeProcess sınıfı gibi platforma bağlı API'lere erişim sağlamanız gerekir.
- Uygulamanız yerel uzantıları kullanır.

Bu gibi durumlarda bir AIR uygulamasını platforma özgü yerel yükleyici dosyası olarak paketleyebilirsiniz.

SWF tabanlı uygulamalar

1. MXML veya ActionScript kodunu yazın.
2. Simge bitmap dosyaları gibi gerekli varlıkları oluşturun.
3. Uygulama tanımlayıcısını oluşturun.
4. ActionScript kodunu derleyin.
5. Uygulamayı test edin.
6. *air* hedefini kullanarak paketleyin ve bir AIR dosyası olarak imzalayın.

Html tabanlı uygulamalar

1. HTML ve JavaScript kodunu yazın.
2. Simge bitmap dosyaları gibi gerekli varlıkları oluşturun.
3. Uygulama tanımlayıcısını oluşturun.
4. Uygulamayı test edin.
5. *air* hedefini kullanarak paketleyin ve bir AIR dosyası olarak imzalayın.

AIR uygulamaları için yerel yükleyiciler oluşturma

1. Kodu yazın (ActionScript veya HTML ve JavaScript).

2 Simge bitmap dosyaları gibi gerekli varlıkları oluşturun.

3 *extendedDesktop* profilini belirterek uygulama tanımlayıcısını oluşturun.

4 Herhangi bir ActionScript kodunu derleyin.

5 Uygulamayı test edin.

6 *native* hedefini kullanarak her hedef platformundaki uygulamayı paketleyin.

Not: *Bir hedef platformu için yerel yükleyici o platformda oluşturulmalıdır. Örneğin, Mac üzerinde bir Windows yükleyici oluşturamazsınız. Aynı bilgisayar donanımında birden çok platform çalıştırmak için VMWare gibi sanal bir makine kullanabilirsiniz.*

Sabit çalışma zamanı paketi ile AIR uygulamaları oluşturma

1 Kodu yazın (ActionScript veya HTML ve JavaScript).

2 Simge bitmap dosyaları gibi gerekli varlıkları oluşturun.

3 *extendedDesktop* profilini belirterek uygulama tanımlayıcısını oluşturun.

4 Herhangi bir ActionScript kodunu derleyin.

5 Uygulamayı test edin.

6 *bundle* hedefini kullanarak her hedefteki uygulamayı paketleyin.

7 Paket dosyalarını kullanarak bir yükleme programı oluşturun. (AIR SDK, bu tür bir yükleyici oluşturmaya yönelik araçlar sağlamaz, ancak çok sayıda üçüncü taraf araç takımı kullanılabilir durumdadır.)

Not: *Bir hedef platformu için paket o platformda oluşturulmalıdır. Örneğin, Mac üzerinde bir Windows paketi oluşturamazsınız. Aynı bilgisayar donanımında birden çok platform çalıştırmak için VMWare gibi sanal bir makine kullanabilirsiniz.*

Masaüstü uygulama özelliklerini ayarlama

Uygulama tanımlayıcı dosyasında temel uygulama özelliklerini ayarlayın. Bu bölüm, masaüstü AIR uygulamaları ile ilgili özellikleri kapsar. Uygulama tanımlayıcı dosyasının öğeleri "AIR uygulama tanımlayıcı dosyaları" sayfa 201 bölümünde tam olarak açıklanmıştır.

Gerekli AIR çalışma zamanı sürümü

Uygulama tanımlayıcı dosyasının ad alanını kullanarak uygulamanızın gerektirdiği AIR çalışma zamanı sürümünü belirtin.

`application` öğesinde atanan ad alanı büyük bir bölümünde uygulamanızın hangi özellikleri kullandığını belirler. Örneğin, uygulamanız AIR 1.5 ad alanını kullanıyorsa ve kullanıcı AIR 3.0'ı yüklemişse uygulamanız AIR 1.5 davranışını görür (davranış AIR 3.0'da değişmiş olsa bile). Yalnızca ad alanını değiştirdiğinizde ve bir güncelleme yayınladığınızda uygulamanız yeni davranışa ve özelliklere erişir. Güvenlik ve WebKit değişiklikleri bu ilkedeki birincil istisnalardır.

Ad alanını kök `application` öğesinin xmlns niteliğini kullanarak belirtin:

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

Daha fazla Yardım konusu

"application" sayfa 206

Uygulama kimliği

Yayınladığınız her uygulama için birkaç ayar benzersiz olmalıdır. Benzersiz ayarlar, kimliği, adı ve dosya adını içerir.

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

Daha fazla Yardım konusu

"id" sayfa 222

"filename" sayfa 217

"name" sayfa 230

Uygulama sürümü

AIR 2.5'ten önceki AIR sürümlerinde uygulamayı `version` öğesinde belirtin. Herhangi bir dizeyi kullanabilirsiniz. AIR çalışma zamanı dizeyi yorumlamaz; "2.0", "1.0" sürümünden daha yüksek bir sürüm olarak algılanmaz.

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

AIR 2.5 ve üstünde, uygulama sürümünü `versionNumber` öğesinde belirtin. `version` öğesi artık kullanılamaz. `versionNumber` öğesi için bir değer belirtirken, noktalarla ayrılmış üç numaraya kadar sayı içerebilen, şunun gibi bir sıralama kullanmalısınız: "0.1.2". Sürüm numarasının her bir parçası en fazla üç basamak içerebilir. (Başka bir deyişle, "999.999.999" izin verilen en büyük sürüm sayısıdır.) Sayıda tüm parçaları içermeniz gerekmez; "1" ve "1.0" örnekleri de geçerli sayılardır.

Ayrıca `versionLabel` öğesini kullanan sürüm için de bir etiket belirtebilirsiniz. Bir sürüm etiketi eklediğinizde, AIR uygulaması yükleyicisi iletişim kutuları gibi yerlerde sürüm numarasının yerine görüntülenir.

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

Daha fazla Yardım konusu

"version" sayfa 238

"versionLabel" sayfa 239

"versionNumber" sayfa 239

Ana pencere özellikleri

AIR masaüstünde bir uygulama başlattığında, bir pencere oluşturur ve bu pencereye ana SWF dosyasını veya HTML sayfası yükler. AIR, bu uygulama başlangıç penceresinin ilk görünümünü ve davranışını denetlemek için `initialWindow` öğesinin alt öğelerini kullanır.

- **content** — `initalWindow` öğesinin `content` alt öğesindeki ana uygulama SWF dosyası. Masaüstü profilinde aygıtları hedeflediğinizde SWF veya HTML dosyası kullanabilirsiniz.

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

Dosyayı AIR paketine dahil etmeniz gerekir (ADT veya IDE'nizi kullanarak) Uygulama tanımlayıcısında ada başvurmak dosyanın otomatik olarak pakete dahil olmasını sağlamaz.

- **depthAndStencil** — Derinlik veya şablon arabelleğini kullanmayı belirtir. Genelde 3B içeriği ile çalışırken bu arabellekleri kullanırsınız.

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

- **height** — İlk pencerenin yüksekliği.
- **maximizable** — Pencerenin ekranı kaplaması için sistem kromunun gösterilip gösterilmediği.
- **maxSize** — İzin verilen maksimum boyut.
- **minimizable** — Pencereyi simge durumuna getirmek için sistem kromunun gösterilip gösterilmediği.
- **minSize** — İzin verilen minimum boyut.
- **renderMode** — AIR 3 veya üst sürümünde masaüstü uygulamaları için oluşturma modu *auto*, *cpu*, *direct* veya *gpu* olarak ayarlanabilir. AIR uygulamasının daha önceki sürümlerinde bu ayar masaüstü platformlarında yoksayılır. renderMode ayarı çalışma zamanında değiştirilemez.
  - auto — aslında cpu modu ile aynıdır.
  - cpu — görüntüleme nesneleri yazılımda görüntüleme belleğinde oluşturulur ve buraya kopyalanır. StageVideo, yalnızca bir pencerenin tam ekran modunda olması durumunda kullanılabilir. Stage3D, yazılım oluşturucusunu kullanır.
  - direct — görüntüleme nesneleri çalışma zamanı yazılımı tarafından oluşturulur, ancak oluşturulan çerçevenin görüntü belleğine kopyalanması (doldurarak oluşturma işlemi) donanımsal olarak hızlandırılır. StageVideo kullanılabilir. Diğer türlüsü mümkünse Stage3D donanım hızlandırmayı kullanır. Pencere saydamlığı true olarak ayarlandığında pencere yazılım oluşturma ve doldurarak oluşturma işlemine "geri döner".

    Not: *Mobil platformlar için AIR ile Flash içeriğinin GPU hızlandırmasını desteklemek üzere Adobe, renderMode="gpu" yerine renderMode="direct" (yani Stage3D) kullanmanızı önerir. Adobe resmi olarak şu Stage3D tabanlı çerçeveleri desteklemekte ve önermektedir: Starling (2D) ve Away3D (3D). Stage3D ve Starling/Away3D hakkında daha fazla ayrıntı için bkz. http://gaming.adobe.com/getstarted/.*

  - gpu — varsa donanım hızlandırma kullanılır.
- **requestedDisplayResolution** — Uygulamanız, yüksek çözünürlüklü ekranlara sahip MacBook Pro bilgisayarlarda *standart* çözünürlük modunu mu yoksa *yüksek* çözünürlük modunu mu kullanmalıdır? Diğer tüm platformlarda değer yoksayılır. Değer *standart* ise, her bir sahne alanı pikseli ekranda dört piksel olarak oluşturulur. Değer *yüksek* ise, her bir sahne alanı pikseli ekranda tek bir fiziksel piksele karşılık gelir. Belirtilen değer tüm uygulama pencerelerinde kullanılır. AIR 3.6 ve üst sürümlerinde, `requestedDisplayResolution` öğesi masaüstü AIR uygulamaları (`intialWindow` öğesinin alt öğesi olarak) için kullanılabilir.
- **resizable** — Pencereyi yeniden boyutlandırmak için sistem kromunun gösterilip gösterilmediği.
- **systemChrome** — Standart işletim sistemi pencere görünümünün kullanılıp kullanılmadığı. Bir pencerenin systemChrome ayarı çalışma zamanında değiştirilemez.
- **title** — Pencerenin başlığı.
- **transparent** — Pencerenin arka planla alfa karışımlı olup olmadığı. Saydamlık açıksa pencere sistem kromunu kullanamaz. Bir pencerenin saydamlık ayarı çalışma zamanında değiştirilemez.

- **visible** — Pencerenin oluşturulur oluşturulmaz görünür olup olmadığı. Varsayılan olarak, uygulamanızın kendini görünür yapmadan önce içeriğini çizmesine zaman tanımak için pencere ilk başta görünür olmaz.
- **width** — Pencerenin genişliği.
- **x** — Pencerenin yatay konumu.
- **y** — Pencerenin dikey konumu.

Daha fazla Yardım konusu

"content" sayfa 211

"depthAndStencil" sayfa 213

"height" sayfa 221

"maximizable" sayfa 229

"maxSize" sayfa 229

"minimizable" sayfa 230

"minimizable" sayfa 230

"minSize" sayfa 230

"renderMode" sayfa 232

"requestedDisplayResolution" sayfa 233

"resizable" sayfa 234

"systemChrome" sayfa 237

"title" sayfa 238

"transparent" sayfa 238

"visible" sayfa 240

"width" sayfa 240

"x" sayfa 240

"y" sayfa 241

Masaüstü özellikleri

Aşağıdaki öğeler, masaüstü yüklemesini ve güncelleme özelliklerini kontrol eder.

- customUpdateUI — Uygulamayı güncellemek için kendi iletişim kutularınızı sağlamanıza izin verir. Varsayılan `false` olarak ayarlanırsa, standart AIR iletişim kutuları kullanılır.
- fileTypes — Uygulamanızın varsayılan açılış uygulaması olarak kaydetmek isteyeceği dosya türlerini belirtir. Bir dosya türü için başka bir uygulama zaten varsayılan açıcı olarak ayarlanmışsa, AIR mevcut kaydı geçersiz kılmaz. Ancak, uygulamanız NativeApplication nesnesinin `setAsDefaultApplication()` yöntemini kullanarak çalışma zamanında kaydı geçersiz kılabilir. Varolan dosya türü ilişkilerini geçersiz kılmadan önce kullanıcının iznini istemek uygun bir davranıştır.

  ***Not:** Bir uygulamayı sabit çalışma zamanı paketi olarak paketlediğinizde (`-bundle` hedefini kullanarak) dosya türü kaydı yoksayılır. Verilen bir dosya türünü kaydetmek için kaydı uygulayan bir yükleyici programı oluşturmalısınız.*

- installFolder — Uygulamanın yüklü olduğu standart uygulama yükleme klasörüne göre bir yol belirtir. Bu ayarı ortak bir klasörde birden fazla uygulamayı gruplamanın yanı sıra özel bir klasör adı sağlamak için de kullanabilirsiniz.
- programMenuFolder — Windows Tüm Programlar menüsü için menü hiyerarşisini belirtir. Bu ayarı birden fazla uygulamayı ortak bir menüde gruplamak için kullanabilirsiniz. Hiçbir menü klasörü belirtilmemişse, uygulama kısayolu doğrudan ana menüye eklenir.

Daha fazla Yardım konusu

"customUpdateUI" sayfa 212

"fileTypes" sayfa 219

"installFolder" sayfa 226

"programMenuFolder" sayfa 232

Desteklenen profiller

Uygulamanız yalnızca masaüstü için uygunsa, profili desteklenen profiller listesinden hariç tutarak başka profil kullanan aygıtlarda yüklenmesini engelleyebilirsiniz. Uygulamanız NativeProcess sınıfını veya yerel uzantıları kullanıyorsa `extendedDesktop` profilini desteklemeniz gerekir.

`supportedProfile` öğesini uygulama tanımlayıcısının dışında bırakırsanız, uygulamanızın tanımlanan tüm profilleri desteklediği varsayılır. Uygulamanızı belirli bir profil listesiyle sınırlandırmak için profilleri boşlukla ayırarak listeleyin:

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

`desktop` ve `extendedDesktop` profillerinde desteklenen ActionScript sınıflarının bir listesi için bkz. "Farklı profillerin yetenekleri" sayfa 243.

Daha fazla Yardım konusu

"supportedProfiles" sayfa 236

Gerekli yerel uzantılar

`extendedDesktop` profilini destekleyen uygulamalar yerel uzantıları kullanabilir.

AIR uygulamasının uygulama tanımlayıcısında kullandığı tüm yerel uzantıları bildirin. Aşağıdaki örnek, gerekli iki yerel uzantıyı belirtmeye yönelik sözdizimini gösterir:

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

`extensionID` öğesi uzantı tanımlayıcı dosyasındaki `id` öğesiyle aynı değere sahiptir. Uzantı tanımlayıcı dosyası extension.xml adlı bir XML dosyasıdır. Bu sözdizimi yerel uzantı geliştiricisinden aldığınız ANE dosyasında paketlenir.

Uygulama simgeleri

Masaüstünde, uygulama tanımlayıcısında belirtilen simgeler uygulama dosyası, kısayol ve program menüsü simgeleri olarak kullanılır. Uygulama simgesi, 16x16, 32x32, 48x48 ve 128x128 piksel PNG görüntüleri kümesi olarak sağlanmalıdır. Uygulama tanımlayıcı dosyasının simge öğesinde simge dosyalarının yolunu belirtin:

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

Belirli bir boyutta bir simge sağlamazsanız, bir sonraki en büyük boyuttaki simge kullanılır ve sığması için ölçeklenir. Hiçbir simge belirtmezseniz, varsayılan sistem simgesi kullanılır.

Daha fazla Yardım konusu

"icon" sayfa 222

"imageNxN" sayfa 223

Yoksayılan ayarlar

Masaüstündeki uygulamalar mobil profil özellikleri için geçerli olan uygulama ayarlarını yok sayar. Yoksayılan ayarlar şunlardır:

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

Bir masaüstü AIR uygulamasında hata ayıklama

Flash Builder, Flash Professional veya Dreamweaver gibi bir IDE ile uygulama geliştiriyorsanız, hata ayıklama araçları normalde yerleşiktir. Uygulamanızı hata ayıklama modunda açarak hata ayıklayabilirsiniz. Doğrudan hata ayıklamayı destekleyen bir IDE kullanmıyorsanız, uygulamanızda hata ayıklamaya yardımcı olması için AIR Hata Ayıklama Başlatıcısı'nı (ADL) ve Flash Hata Ayıklayıcı'yı (FDB) kullanabilirsiniz.

Daha fazla Yardım konusu

De Monsters: Monster Debugger (Büyük Hata Ayıklayıcı)

"AIR HTML Introspector ile hata ayıklama" sayfa 278

Bir uygulamayı ADL ile çalıştırma

Bir AIR uygulamasını, ADL'yi kullanarak paketlemeden ve yüklemeden kullanabilirsiniz. Uygulama tanımlayıcı dosyasını ADL'ye aşağıdaki örnekte gösterildiği gibi bir parametre olarak iletin (önce uygulamada ActionScript kodunun derlenmesi gerekir):

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

ADL izleme ifadelerini, çalışma zamanı istisnalarını ve HTML ayıklama hatalarını terminal penceresine yazdırır. Gelen bir bağlantı için bir FDB işlemi bekliyorsa, ADL hata ayıklayıcısına bağlanır.

Yerel uzantıları kullanan bir AIR uygulamasında hata ayıklamak için ADL öğesini de kullanabilirsiniz. Örneğin:

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

Daha fazla Yardım konusu

"AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157

İzleme ifadelerini yazdırma

İzleme ifadelerini ADL'yi çalıştırmak için kullanılan konsola yazdırmak için, izleme ifadelerini `trace()` işleviyle kodunuza ekleyin.

Not: *`trace()` ifadeleriniz konsolda görüntülenmiyorsa, mm.cfg dosyasında `ErrorReportingEnable` veya `TraceOutputFileEnable` öğesini belirtmediğinizden emin olun. Bu dosyanın platforma özgü konumuyla ilgili daha fazla bilgi için bkz. mm.cfg dosyasını düzenleme.*

ActionScript örneği:

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

JavaScript örneği:

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

JavaScript uygulamasında, uygulamanızdan hata ayıklama mesajları görüntülemek için `alert()` ve `confirm()` işlevlerini kullanabilirsiniz. Buna ek olarak, yakalanmayan JavaScript istisnalarının yanı sıra, sözdizimi hataları için satır numaraları da konsola yazdırılır.

Not: *JavaScript örneğinde gösterilen air önekini kullanmak için AIRAliases.js dosyasını sayfanın içine aktarmanız gerekir. Dosya, AIR SDK'nin frameworks dizininin içindedir.*

Flash Hata Ayıklayıcı'ya (FDB) Bağlanma

Flash Hata Ayıklayıcı'yla AIR uygulamalarının hatalarını ayıklamak için bir FDB oturumu başlatın ve daha sonra ADL kullanarak uygulamayı çalıştırın.

Not: *SWF tabanlı AIR uygulamalarında, ActionScript kaynak dosyalarının `-debug` bayrağıyla derlenmesi gerekir. (Flash Professional'da, Yayınlama Ayarları iletişim kutusundaki hata ayıklamaya İzin Ver seçeneğini işaretleyin.)*

1. FDB'yi başlatın. FDB programı, Flex SDK'nin `bin` dizininde bulunabilir.

   Konsol, FDB istemini görüntüler: `<fdb>`

2. `run` komutunu çalıştırın: `<fdb>run [Enter]`

3. Farklı bir komut veya kabuk konsolunda, uygulamanızın hata ayıklama sürümünü başlatın:

```
adl myApp.xml
```

4 FDB komutlarını kullanarak kesme noktalarını istenilen şekilde oluşturun.

5 Yazın: `continue [Enter]`

AIR uygulaması SWF tabanlıysa, hata ayıklayıcı yalnızca ActionScript kodu yürütmesini kontrol eder. AIR uygulaması HTML tabanlıysa, hata ayıklayıcı yalnızca JavaScript kodu yürütmesini kontrol eder.

Hata ayıklayıcıya bağlanmadan ADL'yi çalıştırmak için `-nodebug` seçeneğini dahil edin:

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

FDB komutları hakkında temel bilgi için `help` komutunu yürütün:

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

FDB komutları hakkındaki ayrıntılar için Flex belgelerindeki Komut satırı hata ayıklayıcı komutlarını kullanma bağına bakın.

Bir masaüstü AIR yükleme dosyasını paketleme

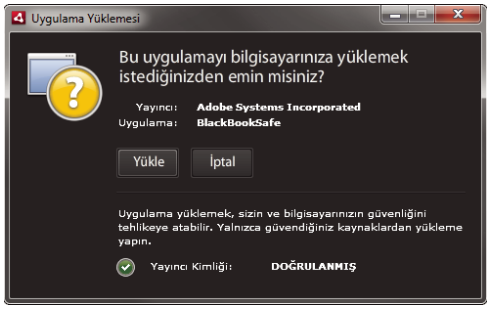
Her AIR uygulaması, en az bir uygulama tanımlayıcı dosyasına ve ana SWF veya HTML dosyasına sahip olmalıdır. Uygulamayla birlikte yüklenmesi gereken diğer varlıklar da AIR dosyasında paketlenmelidir.

Bu makalede SDK bile birlikte gelen komut satırı araçları kullanılarak AIR uygulamasını paketleme anlatılır. Adobe geliştirme araçlarından birini kullanarak bir uygulamayı paketlemeyle ilgili bilgi için aşağıdakilere bakın:

- Adobe® Flex® Builder™, bkz. Packaging AIR applications with Flex Builder (Flash Builder ile AIR uygulamalarını paketleme).
- Adobe® Flash® Builder™, bkz. Flex Builder ile AIR uygulamalarını paketleme.
- Adobe® Flash® Professional, bkz. Adobe AIR için yayınlama.
- Adobe® Dreamweaver®, bkz. Dreamweaver'da AIR uygulaması oluşturma.

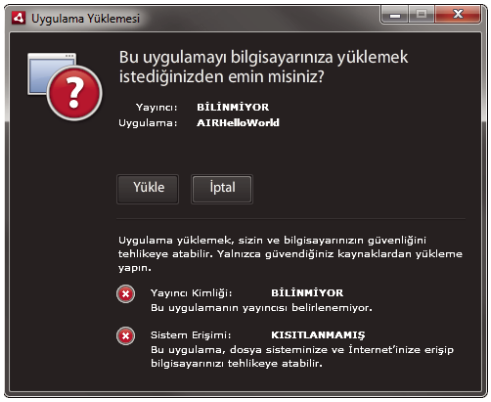
Tüm AIR yükleyici dosyaları dijital sertifika kullanılarak imzalanmalıdır. AIR yükleyici, imzayı, uygulama dosyanızda siz imzaladıktan sonra değişiklik yapılmadığını doğrulamak için kullanır. Bir sertifika yetkilisinden kod imzalayıcı sertifika veya kendinden imzalı bir sertifika kullanabilirsiniz.

Güvenilen bir sertifika yetkilisi tarafından yayımlanan bir sertifika kullandığınızda, uygulamanızın kullanıcılarına yayıncı kimliğinize ilişkin bir güvence sağlarsınız. Yükleme iletişim kutusu, kimliğinizin sertifika yetkisiyle doğrulandığı gerçeğini yansıtır:



Güvenilir bir sertifikayla imzalanan uygulama için yükleme onayı iletişim kutusu

Kendinden imzalı bir sertifika kullandığınızda, kullanıcılar imzalayan olarak kimliğinizi doğrulayamaz. Ayrıca, kendinden imzalı bir sertifika paketin değişmediği güvencesini bozar. (Bunun sebebi, yasal bir yükleme dosyasının kullanıcıya ulaşmadan önce sahte bir imza ile değiştirilebilmesidir) Yükleme iletişim kutusu yayıncının kimliğinin doğrulanamadığını yansıtır. Kullanıcılar uygulamanızı yüklerken daha büyük bir güvenlik riski alır.



Kendinden imzalı bir uygulama için yükleme onayı iletişim kutusu

ADT `-package` komutunu kullanarak bir AIR dosyasını tek adımda paketleyebilir ve imzalayabilirsiniz. Ayrıca `-prepare` komutuyla ara, imzalanmamış bir paket oluşturabilirsiniz ve ara paketi ayrı bir adımda `-sign` komutuyla imzalayabilirsiniz.

Not: *1.5 ve üstü Java sürümleri, PKCS12 sertifika dosyalarını korumak için kullanılan şifrelerde yüksek ASCII karakterlerini kabul etmez. Kod imzalama sertifikanızı oluşturduğunuzda veya dışa aktardığınızda, şifrede yalnızca normal ASCII karakterleri kullanın.*

Yükleme paketini imzalarken, ADT, zamanı doğrulamak için otomatik olarak bir zaman damgası yetkili sunucusuna başvurur. Zaman damgası bilgileri AIR dosyasında bulunmaktadır. Doğrulanmış bir zaman damgası içeren AIR dosyası, ileride herhangi bir zamanda yüklenebilir. ADT zaman damgası sunucusuna ulaşamazsa, paketleme iptal edilir. Zaman damgalama seçeneğini geçersiz kılabilirsiniz, ancak zaman damgası olmadan, yükleme dosyasını imzalamak için kullanılan sertifikanın süresi dolduktan sonra artık AIR uygulaması yüklenemez.

Mevcut bir AIR uygulamasını güncellemek için bir paket hazırlıyorsanız, paket orijinal uygulamayla aynı sertifika ile imzalı olmalıdır. Orijinal sertifika yenilendiyse, sertifikanın süresi son 180 gün içinde dolduysa veya yeni bir sertifikaya geçmek istiyorsanız, bir geçiş imzası uygulayabilirsiniz. Bir geçiş imzası uygulama AIR dosyasının hem yeni hem de eski sertifikayla imzalanmasını içerir. "ADT migrate komutu" sayfa 171 bölümünde açıklandığı gibi geçiş imzasını uygulamak için `-migrate` komutunu kullanın.

Önemli: *Orijinal sertifikanın süresi dolduktan sonra bir geçiş imzası uygulamak için 180 günlük katı bir yetkisiz kullanım süresi vardır. Bir geçiş imzası olmadan, mevcut kullanıcıların yeni sürümünüzü yüklemeden önce mevcut olan uygulamayı kaldırmaları gerekir. Yetkisiz kullanım süresi yalnızca uygulama açıklayıcısı ad alanında AIR'nin 1.5.3 sürümü veya daha sonrasını belirten uygulamalar için geçerlidir. AIR çalışma zamanının daha eski sürümlerini hedeflerken bir yetkisiz kullanım süresi yoktur.*

AIR 1.1'den önce, geçiş imzaları desteklenmiyordu. Geçiş imzası uygulamak için uygulamayı 1.1 veya sonraki sürümlü bir SDK ile paketlemeniz gerekir.

AIR dosyaları kullanılarak dağıtılan uygulamalar masaüstü profil uygulamaları olarak tanınır. Uygulama açıklayıcı dosyası masaüstü profilini desteklemiyorsa, ADT'yi bir AIR uygulaması için yerel bir yükleyiciyi paketlemek için kullanamazsınız. Bu profili uygulama tanımlayıcı dosyasındaki `supportedProfiles` öğesini kullanarak kısıtlayabilirsiniz. Bkz. "Aygıt profilleri" sayfa 242 ve "supportedProfiles" sayfa 236.

Not: *Uygulama tanımlayıcı dosyasındaki ayarlar, AIR uygulamasının kimliğini ve varsayılan yükleme yolunu belirler.* *Bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201.*

Yayıncı Kimlikleri

AIR 1.5.3'te olduğu gibi, yayıncı kimlikleri kabul edilmez. Yeni uygulamaların (ilk defa AIR 1.5.3 veya sonrasıyla yayınlanan) bir yayıncı kimliğine ihtiyacı yoktur ve yayıncı kimliği belirtmemelidir.

AIR'nin daha önceki sürümleriyle yayınlanan uygulamaları güncellerken, uygulama açıklayıcı dosyasında orijinal yayıncı kimliğini belirtmeniz gerekir. Yoksa, uygulamanızın yüklü olan sürümü ve güncel sürümü farklı uygulamalar olarak kabul edilir. Farklı bir kimlik kullanırsanız veya publisherID etiketini çıkarırsanız, kullanıcının yeni sürümü yüklemeden önce önceki sürümü kaldırması gerekir.

Orijinal yayıncı kimliğini belirlemek için, META-INF/AIR alt dizininde orijinal uygulamanın yüklü olduğu `publisherid` dosyasını bulun. Bu dosyanın içindeki dize yayıncı kimliğidir. Uygulama açıklayıcınız, yayıncı kimliğinin elle belirtmek için uygulama açıklayıcısı dosyasının ad alanı bildiriminde AIR 1.5.3 çalışma zamanını (veya sonrasını) belirtmelidir.

AIR 1.5.3'ten önce yayınlanan uygulamalarda - veya uygulama açıklayıcı ad alanında AIR'nin daha eski bir sürümünü belirten ve AIR 1.5.3 SDK ile yayınlananlarda- imza sertifikasına dayanılarak bir yayıncı kimliği belirlenir. Bu kimlik, uygulama kimliğiyle beraber, uygulamanın kimliğinin belirlenmesi için kullanılır. Yayıncı kimliği mevcut olduğunda, aşağıdaki amaçlar için kullanılır:

- Bir AIR dosyasının yüklenecek yeni bir uygulamanın yerine bir güncelleme olduğunu doğrulama
- Şifrelenmiş yerel veri deposuna ilişkin şifreleme anahtarının bir parçası olarak
- Uygulama depolama dizinine ilişkin yolun bir parçası olarak
- Yerel bağlantılara ilişkin bağlantı dizesinin bir parçası olarak
- Bir uygulamanın çağrılması için AIR tarayıcı içi API'si ile kullanılan kimlik dizesinin bir parçası olarak
- OSID'nin bir parçası olarak (özel yükleme/kaldırma programları oluşturulurken kullanılır)

AIR 1.5.3'ten önce, uygulamanın yayıncı kimliği bir uygulama güncellemesini yeni veya yenilenmiş bir sertifika kullanarak geçiş imzasıyla imzalamanız durumunda değişebiliyordu. Bir yayıncı kimliği değiştiğinde, kimliğe bağlı herhangi bir AIR özelliğinin davranışı da değişir. Örneğin, mevcut şifrelenmiş yerel veri deposundaki verilere artık erişilemez ve uygulamaya yerel bağlantı oluşturan Flash veya AIR örnekleri, bağlantı dizesindeki yeni kimliği kullanmalıdır.

AIR 1.5.3 veya sonrasında, yayıncı kimliği imzalama sertifikasını temel almaz ve yalnızca publisherID etiketi uygulama açıklayıcısına dahilse atanır. Güncelleme AIR paketi için belirtilen yayıncı kimliği, geçerli yayıncı kimliğiyle eşleşmiyorsa uygulama güncellenemez.

ADT ile paketleme

Bir AIR uygulamasını paketlemek için AIR ADT komut satırı aracını kullanabilirsiniz. Paketlemeden önce tüm ActionScript, MXML ve uzantı kodları derlenmelidir. Ayrıca, bir kod imzalama sertifikanızın olması gerekir.

ADT komutları ve seçenekleriyle ilgili ayrıntılı bir başvuru için bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163.

AIR paketi oluşturma

Bir AIR paketi oluşturmak üzere sürüm yapıları için hedef türünü *air* olarak ayarlayarak ADT package komutunu kullanın.

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

Örnek, ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu varsayar. (Yardım için bkz. "Path ortam değişkenleri" sayfa 300.)

Komutu uygulama dosyalarını içeren dizinden çalıştırmalısınız. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf ve bir simgeler dizinidir.

Komutu gösterildiği şekilde çalıştırdığınızda ADT sizden anahtar deposu şifresini ister. (Yazdığınız şifre karakterleri her zaman görüntülenmez; yazmanız bittiğinde Enter tuşuna basın.)

AIRI dosyasından AIR paketi oluşturma

Yüklenebilir bir AIR paketi oluşturmak için bir AIRI dosyası oluşturabilirsiniz.

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

Masaüstü yerel yükleyicisini paketleme

AIR 2'den itibaren, AIR uygulamalarını dağıtmak üzere yerel uygulama yükleyicileri oluşturmak için ADT'yi kullanabilirsiniz. Örneğin, Windows'taki bir AIR uygulamasının dağıtımı için bir EXE yükleyici dosyası oluşturabilirsiniz. Mac OS'deki bir AIR uygulamasının dağıtımı için bir DMG yükleyici dosyası oluşturabilirsiniz. AIR 2.5 ve AIR 2.6'da, Linux'taki bir AIR uygulamasının dağıtımı için bir DEB veya RPM yükleyici dosyası oluşturabilirsiniz.

Yerel bir uygulama yükleyicisiyle yüklenen uygulamalar, genişletilmiş masaüstü profil uygulamaları olarak bilinir. Uygulama açıklayıcı dosyası genişletilmiş masaüstü profilini desteklemiyorsa, ADT'yi bir AIR uygulaması için yerel bir yükleyiciyi paketlemek için kullanamazsınız. Bu profili uygulama tanımlayıcı dosyasındaki `supportedProfiles` öğesini kullanarak kısıtlayabilirsiniz. Bkz. "Aygıt profilleri" sayfa 242 ve "supportedProfiles" sayfa 236.

AIR uygulamasının yerel yükleyici versiyonunu iki temel yolla oluşturabilirsiniz:

- Yerel yükleyiciyi uygulama tanımlayıcı dosyası ve başka kaynak dosyalarını temel alarak oluşturabilirsiniz. (Diğer kaynak dosyaları SWF dosyalarını, HTML dosyalarını ve diğer varlıkları içerebilir)
- Yerel yükleyiciyi bir AIR dosyasına veya AIRI dosyasına dayanarak oluşturabilirsiniz.

ADT'yi oluşturmak istediğiniz yerel yükleyici dosyasıyla aynı işletim sisteminde kullanmanız gerekir. Bundan dolayı, Windows'ta bir EXE dosyası oluşturmak için, Windows'ta ADT'yi çalıştırın. Mac OS'de bir DMG dosyası oluşturmak için, ADT'yi MAC OS'de çalıştırın. Linux için bir DEB veya RPG dosyası oluşturmak üzere ADT'yi Linux'taki AIR 2.6 SDK'den çalıştırın.

Bir AIR uygulamasını dağıtmak için bir yerel yükleyici oluşturduğunuzda, uygulama şu yeteneklere sahip olur:

- NativeProcess sınıfını kullanarak yerel işlemler başlatabilir ve onlarla etkileşebilir. Ayrıntılar için, aşağıdakilerden birini inceleyin:
  - AIR'de yerel işlemlerle iletişim kurma (ActionScript geliştiricileri için)
  - Communicating with native processes in AIR (AIR'de yerel işlemlerle iletişim kurma) (HTML geliştiricileri için)
- ADL yerel uzantıları kullanabilir.
- Dosya türüne bakılmaksızın, herhangi bir dosyayı onu açmak için tanımlanmış varsayılan sistem uygulamasıyla açmak için `File.openWithDefaultApplication()` yöntemini kullanabilir. (Bir yerel yükleyiciyle *yüklenmeyen*uygulamalar üzerinde kısıtlamalar vardır. Ayrıntılar için, dil başvurusundaki `File.openWithDefaultApplication()` girişine ilişkin girişi inceleyin.)

Ancak, yerel bir yükleyici olarak paketlendiğinde uygulama AIR dosya biçiminin avantajlarından bazılarını kaybeder. Tek bir dosya tüm masaüstü bilgisayarlara dağıtılamaz. Yerleşik güncelleme işlevi (ve güncelleme çerçevesi) çalışmaz.

Kullanıcı yerel yükleyici dosyasını çift tıklattığında, dosya AIR uygulamasını yükler. Adobe AIR'in istenen sürümü makinede zaten yüklü değilse, yükleyici onu ağdan indirir ve önce onu yükler. Adobe AIR'in (gerekiyorsa) doğru sürümünün alınacağı bir ağ bağlantısı yoksa, yükleme başarısız olur. Ayrıca, yükleme işlemi işletim sistemi Adobe AIR 2'de desteklenmiyorsa da başarısız olur.

Not: *Bir dosyanın yüklü uygulamanızda çalıştırılabilir olmasını istiyorsanız, uygulamanızı paketlediğinizde dosya sisteminde çalıştırılabilir durumda olduğundan emin olun. (Mac ve Linux'ta gerekirse, çalıştırılabilir bayrağını ayarlamak için chmod kullanabilirsiniz.)*

Uygulama kaynak dosyalarından bir yerel yükleyici oluşturma

Uygulamanın kaynak dosyalarından bir yerel yükleyici oluşturmak için, `-package` komutunu aşağıdaki sözdizimi ile kullanın (tek bir komut satırında):

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

Bu sözdizimi, bir AIR dosyasının paketlenmesi için olan sözdizimine benzer (bir yerel yükleyici olmadan). Ancak, bazı farklılıklar vardır:

- Komuta `-target native` seçeneğini eklersiniz. (`-target air` öğesini belirlerseniz, ADT yerel yükleyici dosyası yerine bir AIR dosyası oluşturur.)
- Hedef DMG veya EXE dosyasını `installer_file` olarak belirtirsiniz.
- İsteğe bağlı olarak, Windows'ta sözdizimi listesine `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` olarak belirtilen ikinci bir imzalama seçenekleri kümesi ekleyebilirsiniz. Windows'ta, AIR dosyasını imzalamaya ek olarak, Windows Yükleyici dosyasını da imzalayabilirsiniz. AIR dosyasını imzalamak için kullandığınız sertifika türü ve imzalama seçeneği sözdiziminin aynısını kullanın (bkz. "ADT kod imzalama seçenekleri" sayfa 177). AIR dosyasını ve yükleyici dosyasını imzalamak için aynı sertifikayı kullanabilirsiniz veya farklı sertifikalar belirleyebilirsiniz. Bir kullanıcı web'den imzalı bir Windows Yükleyici dosyası indirdiğinde, Windows dosyanın kaynağını o sertifikayı temel alarak tanımlar.

`-target` seçeneği dışındaki diğer ADT seçenekleriyle ilgili ayrıntı için bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163.

Aşağıdaki örnek bir DMG dosyası oluşturur (Mac OS için bir yerel yükleyici dosyası):

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

Aşağıdaki örnek bir EXE dosyası oluşturur (Windows için bir yerel yükleyici dosyası):

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

Aşağıdaki örnek bir EXE dosyası oluşturur ve onu imzalar:

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

Yerel uzantıları kullanan bir uygulama için yerel bir yükleyici oluşturma

Uygulamaya yönelik kaynak dosyalardan ve uygulamanın gerektirdiği yerel uzantı paketlerinden yerel bir yükleyici oluşturabilirsiniz. Aşağıdaki sözdizimiyle `-package` komutunu kullanın (tek bir komut satırında):

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

Bu sözdizimi, iki ek seçenekle birlikte yerel bir yükleyiciyi paketlemek için kullanılan sözdizimiyle aynıdır. Uygulamanın kullandığı ANE dosyalarını (yerel uzantıları) içeren dizini belirtmek için `-extdir` *`extension-directory`* seçeneğini kullanın. Birincil kod imzalama sertifikası, önceki sürümde kullanılan sertifikadan farklı olduğunda, bir uygulamaya yönelik güncellemeyi geçiş imzası ile imzalamak için isteğe bağlı `-migrate` bayrağını ve `MIGRATION_SIGNING_OPTIONS` parametrelerini kullanın. Daha fazla bilgi için bkz. “Bir AIR uygulamasının güncellenmiş sürümünü imzalama” sayfa 195.

ADT seçenekleri hakkında ayrıntılar için bkz. “AIR Geliştirici Aracı (ADT)” sayfa 163.

Aşağıdaki örnek yerel uzantıları kullanan bir uygulama için bir DMG dosyası (Mac OS için yerel bir yükleyici dosyası) oluşturur.

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

AIR dosyasından veya AIRI dosyasından bir yerel yükleyici oluşturmak.

ADT'yi bir AIR veya AIRI dosyasını temel alarak bir yerel yükleyici dosyası oluşturmak için kullanabilirsiniz. AIR dosyasını temel alarak bir yerel yükleyici oluşturmak için, ADT `-package` komutunu aşağıdaki sözdizimi (tek bir komut satırında) ile kullanın:

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

Bu sözdizimi AIR uygulamasının kaynak dosyalarını temel alarak yerel yükleyici oluşturmak için olan sözdizimine benzer. Ancak, bazı farklılıklar vardır:

- Kaynak olarak, uygulama tanımlayıcı dosyası ve AIR uygulamasına ilişkin diğer kaynak dosyalarının yerine bir AIR dosyası belirtirsiniz.
- AIR dosyası zaten imzalı olduğu için imzalama seçeneklerini belirtmeyin.

AIRI dosyasını temel alarak bir yerel yükleyici oluşturmak için, ADT `-package` komutunu aşağıdaki sözdizimi (tek bir komut satırında) ile kullanın:

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

Bu sözdizimi, yerel yükleyicinin bir AIR dosyası temel alınarak oluşturulması için olan sözdizimine benzer. Ancak, bazı farklılıklar vardır:

- Kaynak olarak bir AIRI dosyası belirtirsiniz.
- Hedef AIR uygulaması için imzalama seçenekleri belirtirsiniz.

Aşağıdaki örnek bir AIR dosyasını temel alarak bir DMG dosyası oluşturur (Mac OS için bir yerel yükleyici dosyası):

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

Aşağıdaki örnek bir AIR dosyasını temel alarak bir EXE dosyası oluşturur (Windows için bir yerel yükleyici dosyası):

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

Aşağıdaki örnek bir EXE dosyası oluşturup (bir AIR dosyasını temel alarak) onu imzalar:

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

Aşağıdaki örnek bir AIRI dosyasını temel alarak bir DMG dosyası oluşturur (Mac OS için bir yerel yükleyici dosyası):

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

Aşağıdaki örnek bir AIRI dosyasını temel alarak bir EXE dosyası oluşturur (Windows için bir yerel yükleyici dosyası):

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

Aşağıdaki örnek bir EXE dosyası (bir AIRI dosyasına bağlı olarak) oluşturur ve bunu hem AIR hem de yerel Windows imzası ile imzalar:

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

Masaüstü bilgisayarlar için yerel bir sabit çalışma zamanı paketleme

Yerel bir çalışma zamanı paketi çalışma zamanının ayrılmış sürümü ile birlikte uygulama kodunuzu içeren bir pakettir. Bu şekilde paketlenen bir uygulama, bir kullanıcının bilgisayarında herhangi bir yere yüklü paylaşılan çalışma zamanı yerine paketlenmiş çalışma zamanını kullanır.

Üretilen paket Windows üzerindeki uygulama dosyalarının ve Mac OS üzerindeki bir .app paketinin bağımsız bir klasörüdür. Bu işletim sistemi altında çalışırken hedef işletim sistemine yönelik paket üretmelisiniz. (VMWare gibi sanal bir makine tek bilgisayarda birden çok işletim sistemi çalıştırmak için kullanılabilir.)

Uygulama yüklemeye gerek kalmadan bu klasörden veya paketten çalıştırılabilir.

Avantajlar

- Bağımsız bir uygulama üretir
- Yükleme için İnternet erişimi gerekmez
- Uygulama, çalışma zamanı güncellemelerinden ayrılır
- Kurumlar belirli uygulama ve çalışma zamanı bileşimini onaylayabilir
- Geleneksel yazılım dağıtma modelini destekler
- Ayrı bir çalışma zamanı yeniden dağıtımı gerekmez
- NativeProcess API'sini kullanabilir
- Yerel uzantıları kullanabilir
- `File.openWithDefaultApplication()` işlevini kısıtlama olmadan kullanabilir
- Yükleme olmadan bir USB'den veya optik diskten çalışabilir

Dezavantajlar

- Önemli güvenlik düzeltmeleri, Adobe bir güvenlik yamasını yayınladığında kullanıcılar tarafından otomatik olarak kullanılabilir durumda olmaz
- .air dosya biçimini kullanamaz
- Gerekirse kendi yükleyicinizi oluşturmanız gerekir
- AIR güncelleme API'si ve çerçeve desteklenmez
- Bir web sayfasından AIR uygulaması yüklemeye ve başlatmaya yönelik AIR tarayıcı içi API'si desteklenmez
- Windows'ta dosya kaydı tarayıcınız tarafından ele alınmalıdır
- Uygulama diskinin kapladığı alan daha büyüktür

Windows'ta sabit bir çalışma zamanı paketi oluşturma

Windows için sabit bir çalışma zamanı paketi oluşturmak üzere Windows işletim sistemi altında çalışırken uygulamayı paketlemelisiniz. ADT *paket* hedefini kullanarak uygulamayı paketleyin:

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

Bu komut, paketi myApp adlı bir dizinde oluşturur. Dizin, çalışma zamanı dosyalarının yanı sıra uygulamanız için dosyaları içerir. Programı doğrudan klasörden çalıştırabilirsiniz. Ancak, program menüsü girişi, kayıt dosyası türleri veya URI şema işleyicileri oluşturmak için gerekli kayıt girişlerini ayarlayan bir yükleyici programı oluşturmalısınız. AIR SDK bu tür yükleyiciler oluşturmaya yönelik araçlar içermez, ancak hem ticari hem ücretsiz açık kaynaklı yükleyici araç takımları dahil birkaç üçüncü taraf seçeneği kullanılabilir durumdadır.

Komut satırındaki `-target bundle` girişinden sonra ikinci bir imzalama seçeneği ayarını belirterek Windows'taki yerel yürütülebilir dosyayı imzalayabilirsiniz. Bu imzalama seçenekleri yerel Windows imzasını uygularken kullanılacak özel anahtarı ve ilişkilendirilmiş sertifikayı tanımlar. (Bir AIR kod imzalama sertifikası normal şekilde kullanılabilir.) Yalnızca birincil çalıştırılabilir dosya imzalanır. Uygulamanızla paketlenmiş çalıştırılabilir tüm ek dosyalar bu işlemle imzalanmaz.

Dosya türü ilişkisi
Uygulamanızı, Windows'taki genel veya özel dosya türleriyle ilişkilendirmek için yükleyici programınız uygun kayıt girişlerini ayarlamalıdır. Dosya türlerinin, uygulama tanımlayıcı dosyasının dosya türleri öğesinde de listelenmesi gerekir.

Windows dosya türleri hakkında daha fazla bilgi için bkz. MSDN Kütüphanesi: Dosya Türleri ve Dosya İlişkileri

URI işleyici kaydı
Uygulamanızın verilen bir URI şemasını kullanarak URL'yi başlatmayı ele alması için yükleyiciniz gerekli kayıt girişlerini ayarlamalıdır.

Bir URI şemasını işlemek üzere bir uygulamayı kaydetme hakkında daha fazla bilgi için bkz. MSDN Kütüphanesi: Bir Uygulamayı Bir URL Protokolüne Kaydetme

Mac OS X üzerinde sabit bir çalışma zamanı paketi oluşturma

Mac OS X için sabit bir çalışma zamanı paketi oluşturmak üzere Macintosh işletim sistemi altında çalışırken uygulamayı paketlemelisiniz. ADT *paket* hedefini kullanarak uygulamayı paketleyin:

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

Bu komut myApp.app adlı uygulama paketini oluşturur. Paket, çalışma zamanı dosyalarının yanı sıra uygulamanız için dosyaları içerir. myApp.app simgesini çift tıklatarak uygulamayı çalıştırabilir ve bu uygulamayı Uygulamalar klasörü gibi uygun bir konuma sürükleyerek yükleyebilirsiniz. Ancak, dosya türlerini veya URI şema işleyicilerini kaydetmek için uygulama paketi içindeki özellik listesi dosyası düzenlenmelidir.

Dağıtım için bir disk görüntü dosyası (.dmg) oluşturabilirsiniz. Adobe AIR SDK, sabit bir çalışma zamanı paketi için bir dmg dosyası oluşturmaya yönelik araçlar sağlamaz.

Dosya türü ilişkisi
Uygulamanızı Mac OS X üzerinde genel veya özel dosya türleriyle ilişkilendirmek için CFBundleDocumentTypes özelliğini ayarlamak üzere paketteki info.plist dosyasını düzenlemelisiniz. Bkz. Mac OS X Developer Library: Information Property List Key Reference, CFBundleURLTypes (Mac OS X Geliştirici Kütüphanesi: Bilgi Özelliği Listesi Anahtarı Başvurusu CFBundleURLTypes).

URI işleyici kaydı
Uygulamanızın verilen bir URI şemasını kullanarak URL'yi başlatmayı ele alması için CFBundleURLTypes özelliğini ayarlamak üzere paketteki info.plist dosyasını düzenlemelisiniz. Bkz. Mac OS X Geliştirici Kütüphanesi: Bilgi Özelliği Listesi Anahtar Referansı, CFBundleDocumentTypes.

Masaüstü bilgisayarlar için AIR paketlerini dağıtma

AIR uygulamaları, uygulama kodunu ve tüm varlıkları içeren bir AIR paketi olarak dağıtılabilir. Bu paketi, indirme, e-posta gibi genel yöntemlerden biriyle veya CD-ROM gibi bir fiziksel ortamla dağıtabilirsiniz. Kullanıcılar AIR dosyasını çift tıklatarak uygulamayı yükleyebilir. Kullanıcıların web sayfasında tek bir bağlantıyı tıklatarak AIR uygulamanızı (ve gerekirse Adobe® AIR® uygulamasını)yüklemelerine izin vermek için AIR tarayıcı içi API'sini (web tabanlı ActionScript kütüphanesi) kullanabilirsiniz.

AIR uygulamaları ayrıca yerel yükleyiciler olarak paketlenebilir ve dağıtılabilir (başka bir deyişle Windows'ta EXE dosyaları, MAC'te DMG dosyaları ve Linux'ta DEB veya RPM dosyaları olarak). Yerel yükleme paketleri ilgili platform kurallarına göre dağıtılabilir ve yüklenebilir. Uygulamanızı yerel paket olarak dağıtırken, AIR dosyası biçimi avantajlarından bazıları kaybedersiniz. Yani, tek bir yükleme dosyası artık birçok platformda kullanılamaz, AIR güncelleme çerçevesi artık kullanılamaz ve tarayıcı içi API'si artık kullanılamaz.

Masaüstünde bir AIR uygulamasını yükleme ve çalıştırma

Yalnızca AIR dosyasını alıcıya gönderin. Örneğin AIR dosyasını e-posta eki olarak veya bir web sayfasında bağ olarak gönderebilirsiniz.

Kullanıcı AIR dosyasını indirdikten sonra, dosyayı yüklemek için şu talimatları izler:

1 AIR dosyasını çift tıklatın.

Adobe AIR bilgisayarda önceden yüklü olmalıdır.

2 Yükleme penceresinde, varsayılan ayarları seçili halde bırakın ve ardından Devam düğmesini tıklatın.

Windows'ta AIR otomatik olarak şunları yapar:

- Uygulamayı Program Dosyaları dizinine yükler
- Uygulama için bir masaüstü kısayolu oluşturur
- Başlat Menüsü kısayolu oluşturur.
- Program Ekle / Kaldır Denetim Masası'na uygulama için bir giriş ekler

Mac OS'de, uygulama varsayılan olarak Uygulamalar dizinine eklenir.

Uygulama zaten yüklenmişse yükleyici, kullanıcıya, uygulamanın var olan sürümünü açma veya indirilen AIR dosyasındaki sürüme güncelleme seçenekleri arasında seçim yapma olanağı tanır. Yükleyici AIR dosyasındaki uygulama kimliğini ve yayıncı kimliğini kullanarak uygulamayı tanımlar.

3 Yükleme işlemi tamamlandığında, Son düğmesini tıklatın.

Mac OS'de bir uygulamanın güncellenen sürümünü yüklemek için, kullanıcının uygulama dizinine yüklemek üzere yeterli sistem ayrıcalığına sahip olması gerekir. Kullanıcı, Windows ve Linux'ta yönetici ayrıcalıklarına ihtiyaç duyar.

Ayrıca uygulama yeni bir sürümü, ActionScript veya JavaScript kullanarak da yükleyebilir. Daha fazla bilgi için bkz. "AIR uygulamalarını güncelleme" sayfa 254.

AIR uygulaması yüklendikten sonra kullanıcı uygulamayı çalıştırmak için, diğer masaüstü uygulamalarında da olduğu gibi yalnızca uygulama simgesini çift tıklatır.

- Windows'ta uygulamasının simgesini çift tıklatın (masaüstüne veya bir klasöre yüklüdür) veya Başlat menüsünden uygulamayı seçin.
- Linux'ta, uygulama simgesini (masaüstüne veya bir klasöre yüklenen) çift tıklatın veya uygulamalar menüsünden uygulamayı seçin.
- Mac OS'de uygulamayı yüklendiği klasörde çift tıklatın. Varsayılan yükleme dizini /Applications dizinidir.

Not: *Linux'ta yalnızca AIR 2.6 veya öncesi için geliştirilmiş olan AIR uygulamaları yüklenebilir.*

AIR *kesintisiz yükleme* özelliği sayesinde kullanıcı AIR uygulamasını web sayfasındaki bir bağı tıklatarak yükleyebilir. AIR *tarayıcı başlatma* özelliği sayesinde kullanıcı yüklenmiş olan AIR uygulamasını web sayfasındaki bir bağı tıklatarak çalıştırabilir. Bu özellikler aşağıdaki bölümde anlatılmaktadır.

Bir web sayfasından masaüstü AIR uygulamalarını yükleme ve çalıştırma

AIR tarayıcı içi API'si, web sayfasından AIR uygulamasını yüklemenize ve çalıştırmanıza izin verir. AIR tarayıcı içi API'si, Adobe tarafından barındırılan bir SWF kütüphanesi olan *air.swf* içinde sağlanır. AIR SDK, bir AIR uygulamasını (ve gerekirse çalışma zamanını) yüklemek, güncellemek veya başlatmak için bu kütüphaneyi kullanan örnek bir "rozet" uygulaması içerir. Sağlanan örnek rozeti değiştirebilir veya doğrudan air.swf kütüphanesini kullanan kendi rozet web uygulamanızı oluşturabilirsiniz.

Web sayfası rozetiyle herhangi bir AIR uygulaması yüklenebilir. Ancak, yalnızca uygulama tanımlayıcı dosyalarında `<allowBrowserInvocation>true</allowBrowserInvocation>` öğesini içeren uygulamalar bir web rozeti tarafından başlatılabilir.

Daha fazla Yardım konusu

"AIR.SWF tarayıcı içi API'si" sayfa 246

Masaüstü bilgisayarlarda kurumsal dağıtım

BT yöneticileri Adobe AIR çalışma zamanı ve AIR uygulamalarını, standart masaüstü konuşlandırma araçlarını kullanarak sessiz bir şekilde yükleyebilir. BT yöneticileri şunları yapabilir:

- Microsoft SMS, IBM Tivoli gibi araçları veya önyükleyici kullanan sessiz yükleme işlemlerine olanak sağlayan herhangi bir konuşlandırma aracını kullanarak Adobe AIR çalışma zamanını sessizce yükleyebilir
- Çalışma zamanını konuşlandırmak için kullanılan araçların aynılarını kullanarak AIR uygulamasını sessizce yükleyebilir

Daha fazla bilgi için, bkz. *Adobe AIR Yönetici Kılavuzu (http://www.adobe.com/go/learn_air_admin_guide_tr).*

Yükleme günlükleri masaüstü bilgisayarlarda

AIR çalışma zamanının kendisi veya bir AIR uygulaması yüklendiğinde yükleme günlükleri kaydedilir. Gerçekleşen herhangi bir yükleme veya güncelleme sorununun nedenini belirlemenize yardımcı olması için günlük dosyalarını inceleyebilirsiniz.

Günlük dosyaları aşağıdaki konumlarda oluşturulur:

- Mac: standart sistem günlüğü(`/private/var/log/system.log`)

  Console (Konsol) uygulamasını (genellikle Utilities (İzlenceler) klasöründe bulunur) açarak Mac sistem günlüğünü görüntüleyebilirsiniz.

- Windows XP: `C:\Documents and Settings\<kullanıcı adı>\Local Settings\Application Data\Adobe\AIR\logs\Install.log`
- Windows Vista, Windows 7: `C:\Users\<kullanıcı adı>\AppData\Local\Adobe\AIR\logs\Install.log`
- Linux: `/home/<kullanıcı adı>/.appdata/Adobe/AIR/Logs/Install.log`

Not: *Bu günlük dosyaları AIR'in AIR 2'den önceki sürümlerinde oluşturulmuyordu.*

Bölüm 7: Mobil aygıtlar için AIR uygulamaları geliştirme

Mobil aygıtlarda AIR uygulamaları yerel uygulamalar olarak dağıtılır. Bunlar AIR dosya biçimini değil, aygıtın uygulama biçimini kullanır. Şu anda AIR Android APK paketlerini ve iOS IPA paketlerini destekler. Uygulamanızın yayın sürümünü oluşturduktan sonra uygulamanızı standart platform mekanizması ile dağıtabilirsiniz. Bu genellikle Android için Android Market ve iOS için Apple App Store anlamına gelir.

Mobil aygıtlar için AIR uygulamaları oluşturmak üzere AIR SDK ve Flash Professional, Flash Builder veya başka bir ActionScript geliştirme aracını kullanabilirsiniz. HTML tabanlı mobil AIR uygulamaları şu anda desteklenmiyor.

Not: *Research In Motion (RIM) BlackBerry Playbook kendi AIR için SDK geliştirmesini sağlar. Playbook geliştirmesiyle ilgili bilgi için bkz. RIM: BlackBerry Tablet OS Development (RIM: BlackBerry Tablet OS Geliştirme).*

Not: *Bu belge AIR 2.6 SDK ve üstünü kullanarak iOS uygulamalarının nasıl geliştirileceğini açıklar. AIR 2.6 ve üstüyle oluşturulan uygulamalar iOS 4 ve üstüyle çalışan iPhone 3Gs, iPhone 4 ve iPad cihazlarına yüklenebilir. iOS'nin önceki sürümleri için AIR uygulamaları oluşturmak üzere iPhone uygulamaları oluşturma bölümünde açıklandığı şekilde AIR 2 Packager for iPhone uygulamasını kullanmanız gerekir.*

Gizlilikle ilgili en iyi uygulamalar hakkında daha fazla bilgi için bkz. Adobe AIR SDK Gizlilik Kılavuzu.

AIR uygulamalarını çalıştırmak için gereken tam sistem gereksinimleri için bkz. Adobe AIR sistem gereksinimleri.

Geliştirme ortamınızı ayarlama

Mobil platformların normal AIR, Flex ve Flash geliştirme ortamı kurulumunun ötesinde ek kurulum gereksinimleri vardır. (Temel AIR geliştirme ortamını ayarlamayla ilgili daha fazla bilgi için bkz. “AIR geliştirme için Adobe Flash Platform araçları” sayfa 16.)

Android kurulumu

AIR 2.6 ve üstünde Android için normalde özel bir kurulum gerekmez. Android ADB aracı AIR SDK'ye dahildir (lib/android/bin klasöründe bulunur). AIR SDK, aygıtta uygulama paketleri yüklemek, kaldırmak ve çalıştırmak için ADB aracını kullanır. Ayrıca ADB'yi sistem günlüklerini görüntülemek için de kullanabilirsiniz. Android taklitçisi oluşturmak ve çalıştırmak için ayrı Android SDK'yi indirmeniz gerekir.

Uygulamanız öğeleri AIR'nin geçerli sürümünün geçerli olarak tanımadığı uygulama tanımlayıcısında `<manifestAdditions>` öğesine ekliyorsa, Android SDK'nin daha güncel bir sürümünü yüklemeniz gerekir. AIR_ANDROID_SDK_HOME ortam değişkenini veya `-platformsdk` komut satırını SDK'nin dosya yoluna ayarlayın. AIR paketleme aracı olan ADT, `<manifestAdditions>` öğesinde girişleri doğrulamak için bu SDK'yi kullanır.

AIR 2.5'te, Google'dan Android SDK'nin ayrı bir kopyasını indirmeniz gerekir. AIR_ANDROID_SDK_HOME ortam değişkenini Android SDK klasörüne başvuracak şekilde ayarlayabilirsiniz. Bu ortam değişkenini ayarlamazsanız ADT komut satırında `-platformsdk` argümanında Android SDK yolunu belirtmeniz gerekir.

Daha fazla Yardım konusu

"ADT ortam değişkenleri" sayfa 185

"Path ortam değişkenleri" sayfa 300

iOS kurulumu

Bir aygıtta iOS uygulaması yüklemek ve test etmek ve uygulamayı dağıtmak için Apple iOS Developer programına (ücretli bir hizmettir) katılmanız gerekir. iOS Developer programına katıldığınızda bir aygıtta test etmek ve ardından dağıtmak üzere uygulama yüklemek için gerekli olan aşağıdaki öğeleri ve dosyaları Apple'dan alabileceğiniz iOS Provisioning Portal sitesine erişebilirsiniz. Bu öğeler ve dosyalar şunları içerir:

- Geliştirme ve dağıtım sertifikaları
- Uygulama kimlikleri
- Geliştirme ve dağıtım ön hazırlık dosyaları

Mobil uygulama tasarımında dikkat edilmesi gerekenler

İşletim bağlamı ve mobil aygıtların fiziksel özellikleri dikkatli bir şekilde yapılmış kodlama ve tasarım gerektirir. Örneğin, mümkün olduğu kadar hızlı yürütülmesini sağlamak için kodu düzenlemek çok önemlidir. Kod en iyileştirmesi yalnızca bir noktaya kadar devam edebilir. Cihaz kısıtlamaları dahilinde çalışan akıllı tasarım ayrıca, görsel sunumunuzun, oluşturma sistemini aşırı derecede yormasını önlemeye yardımcı olabilir.

Kod

Kodunuzun daha hızlı çalışmasını sağlamak her zaman faydalıyken, birçok aygıttaki daha düşük işlemci hızı yan kod yazarken harcanan sürenin faydalarını artırır. Ek olarak, mobil aygıtlar neredeyse her zaman pil gücüyle çalışır. Aynı sonuca daha az işlem ile ulaşmak daha az pil kullanır.

Tasarım

Uygulamanızın kullanıcı deneyimi tasarımını yaparken, küçük ekran boyutu, dokunmatik ekran etkileşim modu gibi faktörler ve cep telefonu kullanıcısının sürekli değişen ortamı bile düşünülmelidir.

Kod ve tasarım bir arada

Uygulamanızda animasyon kullanılıyorsa, görüntü oluşturmanın en iyi duruma getirilmesi çok önemlidir. Ancak, tek başına kodu en iyi duruma getirmek genellikle yeterli değildir. Uygulamanın görsel yönlerini kodun bunları verimli bir şekilde oluşturmasına izin verecek şekilde tasarlamalısınız.

Önemli en iyi duruma getirme teknikleri Flash Platform için İçeriği En İyi Duruma Getirme kılavuzunda açıklanmıştır. Kılavuzdaki teknikler tüm Flash ve AIR içeriği için geçerlidir, ancak mobil aygıtlarda düzgün bir şekilde çalışan uygulamalar geliştirmek için gereklidir.

- Paul Trani: Tips and Tricks for Mobile Flash Development (Mobil Flash Geliştirmesi için İpuçları ve Püf Noktaları)
- roguish: GPU Test App AIR for Mobile (AIR for Mobile için GPU Test Uygulaması)
- Jonathan Campos: Optimization Techniques for AIR for Android apps (AIR for Android Uygulamaları için En İyileştirme Teknikleri)
- Charles Schulze: AIR 2.6 Game Development: iOS included (AIR 2.6 Oyun Geliştirmesi: iOS dahil)

Uygulama yaşam döngüsü

Uygulamanız odağı başka bir uygulama nedeniyle kaybettiğinde, AIR kare hızını saniye başına 4 kareye düşürür ve grafik oluşturmayı durdurur. Bu kare hızının altında, ağ akışı ve yuva bağlantıları kesilir. Uygulamanız bu tür bağlantılar kullanmıyorsa, kare hızını daha da azaltabilirsiniz.

Uygun olduğunda, ses oynatımını durdurmalı ve dinleyicileri coğrafi konum ve akselerometre sensörlerine kaldırmalısınız. AIR NativeApplication nesnesi activate ve deactivate olaylarını gönderir. Bu olayları etkin ve arka plan durumu arasındaki geçişi yönetmek için kullanın.

Birçok mobil işletim sistemi uyarı vermeden arka plan uygulamalarını sonlandırır. Uygulama durumunu sıkça kaydederek, uygulamanız arka plandan etkin duruma dönerken veya tekrardan başlatılırken kendini kabul edilebilir bir duruma döndürebilmelidir.

Bilgi yoğunluğu

Mobil aygıtların fiziksel boyutu masaüstündekinden küçük olsa da, piksel yoğunluğu (inç başına piksel sayısı) daha yüksektir. Aynı font boyutu mobil aygıtta masaüstü bilgisayardakinden fiziksel olarak daha küçük olan harfler üretir. Okunabilirliği sağlamak için genellikle daha büyük yazı tipi kullanmanız gerekir. Genel olarak, 14 nokta kolaylıkla okunabilen en küçük font boyutudur.

Mobil aygıtlar çoğunlukla hareket halinde ve zayıf aydınlatma koşulları altında kullanılır. Ekranda ne kadar bilgiyi gerçekçi bir şekilde okunaklı olarak görüntüleyebileceğinizi düşünün. Bir masaüstünde aynı piksel boyutlarına sahip bir ekranda görüntüleyebileceğinizden daha az olacaktır.

Ayrıca, kullanıcı ekrana dokunurken kullanıcının parmağının ve elinin ekranın bir kısmının görünmesini engellediğini göz önünde bulundurun. Etkileşimli öğeleri, kullanıcıların anlık bir dokunmadan daha fazla etkileşim kurmaları gerekiyorsa ekranın yanlarına ve altına yerleştirin.

Metin girişi

Birçok aygıt metin girişi için sanal klavye kullanır. Sanal klavyeler ekranın bir parçasını görünmez hale getirir ve genellikle kullanımı zordur. Keyboard olaylarına güvenmekten kaçının (yazılım tuşları dışında).

Giriş metni alanlarını kullanmanın alternatiflerini uygulamayı düşünün. Örneğin, kullanıcının sayısal bir değer girmesini sağlamak için bir metin alanı gerekmez. Değeri arttırmak veya azaltmak için iki düğme verebilirsiniz.

Yazılım tuşları

Mobil aygıtlar çeşitli sayıda yazılım tuşu içerir. Yazılım tuşları farklı işlevlere sahip olmak üzere programlanabilir düğmelerdir. Uygulamanızda bu tuşlar için platform kurallarını izleyin.

Ekran yönlendirme değişiklikleri

Mobil içeriği dikey veya yatay yönde görüntülenebilir. Uygulamanızın ekran yönlendirme değişikliklerini nasıl işleme koyacağını düşünün. Daha fazla bilgi için bkz. Sahne alanı yönlendirmesi.

Ekran karartma

AIR video oynatılırken ekranın kararmasını otomatik olarak engellemez. Aygıtın güç tasarrufu moduna girip girmeyeceğini kontrol etmek için AIR NativeApplication nesnesinin `systemIdleMode` özelliğini kullanabilirsiniz. (Bazı platformlarda bu özelliğinin çalışması için uygun izinleri istemeniz gerekir.)

Gelen telefon çağrıları

Kullanıcı arama yaptığında veya çağrı aldığında AIR otomatik olarak sesi kapatır. Android'de, uygulamanız arka plandayken sesi oynatıyorsa, uygulama tanımlayıcısında Android READ_PHONE_STATE iznini ayarlamanız gerekir. Aksi halde, Android çalışma zamanının telefon çağrılarını algılamasını ve sesi otomatik olarak kapatmasını engeller. Bkz. “Android izinleri” sayfa 74.

Vuruş hedefleri

Düğmeleri ve kullanıcının tıklattığı diğer kullanıcı arabirimi öğelerini tasarlarken vuruş hedeflerinin boyutunu düşünün. Bu öğeleri dokunmatik bir ekranın bir parmak ile rahatça etkinleştirilebileceği büyüklükte oluşturun. Ayrıca, hedefler arasında yeterince boşluk olduğundan emin olun. Tipik yüksek dpi'li bir telefon ekranı için vuruş hedef alanı her iki tarafta yaklaşık 44 piksele 57 piksel olmalıdır.

Uygulama paketi yükleme boyutu

Mobil aygıtlar genellikle uygulamaları ve verileri yüklemek için masaüstü bilgisayarlardan çok daha az alana sahiptir. Kullanılmayan varlıkları ve kütüphaneleri kaldırarak paket boyutunu en aza indirin.

Android'de, uygulama paketi uygulama yüklendiğinde ayrı dosyalara açılmaz. Bunun yerine, varlıklara erişildiğinde varlıklar geçici bir depoya sıkıştırılır. Bu sıkıştırılmış varlık depolamasının kapladığı alanı en aza indirmek için varlıklar tamamen yüklendiğinde dosyayı ve URL akışlarını kapatın.

Dosya sistemi erişimi

Farklı mobil işletim sistemleri farklı dosya sistemi kısıtlamaları uygular ve bu kısıtlamalar masaüstü işletim sistemleri tarafından uygulananlardan farklı olma eğilimindedir. Bu nedenle dosyaları ve verileri kaydetme yerleri platformdan platforma değişebilir.

Dosya sistemlerindeki farklılığın bir sonucu, AIR File sınıfı tarafından sağlanan yaygın dizinlerin kısayollarının her zaman kullanılabilir olmamasıdır. Aşağıdaki tablo, hangi kısayolların Android ve iOS'de kullanılabileceğini gösterir:

	Android	**iOS**
File.applicationDirectory	URL (yerel yol değil) aracılığıyla salt okunur	Salt okunur
File.applicationStorageDirectory	Kullanılabilir	Kullanılabilir
File.cacheDirectory	Kullanılabilir	Kullanılabilir
File.desktopDirectory	Sd kartın kökü	Kullanılamaz
File.documentsDirectory	Sd kartın kökü	Kullanılabilir
File.userDirectory	Sd kartın kökü	Kullanılamaz
File.createTempDirectory()	Kullanılabilir	Kullanılabilir
File.createTempFile()	Kullanılabilir	Kullanılabilir

iOS uygulamalarına yönelik Apple talimatlarında, farklı durumlardaki dosyalar için hangi depolama konumlarının kullanılması gerektiği ile ilgili belirli kurallar sağlanır. Örneğin, talimatlardan biri şu şekildedir: Yalnızca kullanıcı girişli veriler veya farklı şekilde yeniden oluşturulamayan ya da yeniden indirilemeyen veriler içeren dosyalar, uzaktan yedekleme için atanan bir dizine depolanmalıdır. Yedekleme ve önbelleğe alma konusunda Apple talimatlarına nasıl uyulacağı ile ilgili bilgi için bkz. Dosya yedeklemeyi ve önbelleğe almayı denetleme.

UI bileşenleri

Adobe, Flex çerçevesinin mobil aygıtlar için en iyi duruma getirilmiş bir sürümünü geliştirmiştir. Daha fazla bilgi için bkz. Flex ve Flash Builder ile Mobil Uygulamalar Geliştirme.

Mobil uygulamalar için uygun olan topluluk bileşeni projeleri de mevcuttur. Bunlar şunları içerir:

- Josh Tynjala'nın sunduğu Feathers UI controls for Starling (Starling için Feathers UI denetimleri)
- Derrick Grigg'in Minimal Comps'un dış görünümü değiştirilebilen sürümü
- Todd Anderson'ın as3flobile bileşenleri

Stage 3D hızlandırmalı grafik oluşturma

AIR 3.2'den itibaren mobil için AIR Stage 3D hızlandırmalı grafik oluşturmayı desteklemektedir. Stage3D ActionScript API'leri gelişmiş 2B ve 3B özellikleri sağlayan bir düşük düzey GPU hızlandırmalı API kümesidir. Bu düşük düzey API'ler geliştiricilere önemli performans kazançları için GPU donanım hızlandırmasından faydalanma esnekliğini sağlar. Ayrıca Stage3D ActionScript API'lerini destekleyen oyun motorlarını da kullanabilirsiniz.

Daha fazla bilgi için bkz. Oyun motorları, 3B ve Stage 3D.

Video düzgünleştirme

Performansı geliştirmek için AIR'de video düzgünleştirme devre dışıdır.

Yerel özellikler

AIR 3.0+

Birçok mobil platform, standart AIR API ile henüz erişilebilir olmayan özellikleri sağlamaktadır. AIR 3'ten itibaren, AIR'yi yerel kod kütüphanelerinizle genişletebilirsiniz. Bu yerel uzantı kütüphaneleri işletim sisteminde kullanılabilir olan veya hatta belirli bir aygıta özgü olan özelliklere erişebilir. Yerel uzantılar, iOS'taki C'ye ve Java'ya veya Android'deki C'ye yazılabilir. Yerel uzantıları geliştirme ile ilgili bilgi için Adobe AIR yerel uzantılarına giriş bölümüne bakın.

Mobil aygıtlar için AIR uygulamaları oluşturma iş akışı

Mobil (veya diğer) aygıtlar için AIR uygulaması oluşturma iş akışı genellikle masaüstü uygulaması oluşturma işlemine çok benzer. Önemli iş akışı farklılıkları bir uygulamayı paketleme, hata ayıklama ve yükleme söz konusu olduğunda ortaya çıkar. Örneğin, AIR for Android uygulamaları AIR paketi biçimi yerine yerel Android APK paketi biçimini kullanır. Dolayısıyla, bunlar da standart Android yükleme ve güncelleme mekanizmalarını kullanır.

AIR for Android

Aşağıdaki adımlar Android için AIR uygulaması geliştirirken gerçekleştirilen tipik adımlardır:

- ActionScript veya MXML kodunu yazın.
- Bir AIR uygulama tanımlayıcı dosyası oluşturun (2.5 veya daha üstü ad alanını kullanarak).
- Uygulamayı derleyin.
- Uygulamayı Android paketi (.apk) olarak paketleyin.

- AIR çalışma zamanını aygıta veya Android taklitçisine yükleme (harici bir çalışma zamanı kullanılıyorsa sabit çalışma zamanı AIR 3.7 ve üstünde varsayılandır).
- Uygulamayı aygıtta (veya Android taklitçisinde) yükleyin.
- Uygulamayı aygıtta başlatın.

Bu adımları gerçekleştirmek için Adobe Flash Builder, Adobe Flash Professional CS5 veya komut satırı araçlarını kullanabilirsiniz.

AIR uygulamanız bittiğinde ve APK dosyası olarak paketlendiğinde, Android Market'a gönderebilir veya başka yöntemlerle dağıtabilirsiniz.

AIR for iOS

Aşağıdaki adımlar iOS için AIR uygulaması geliştirirken gerçekleştirilen tipik adımlardır:

- iTunes programını yükleyin.
- Apple iOS Provisioning Portal'da gerekli geliştirici dosyalarını ve kimlikleri oluşturun. Bu öğeler şunları içerir:
  - Geliştirici sertifikası
  - Uygulama Kimliği
  - Ön hazırlık profili

  Ön hazırlık profilini oluştururken uygulamayı yüklemeyi planladığınız test aygıtlarının kimliklerini listelemelisiniz.
- Geliştirme sertifikasını ve özel anahtarı P12 anahtar deposu dosyasına dönüştürün.
- Uygulamanın ActionScript veya MXML kodunu yazın.
- Uygulamayı bir ActionScript veya MXML derleyicisiyle derleyin.
- Uygulama için simge resmi ve başlangıç ekranı resmi oluşturun.
- Uygulama tanımlayıcı dosyasını oluşturun (2.6 veya üstü ad alanını kullanarak).
- IPA dosyasını ADT kullanarak paketleyin.
- Ön hazırlık profilinizi test aygıtına yerleştirmek için iTunes'u kullanın.
- Uygulamayı iOS aygıtınıza yükleyip test edin. IPA dosyasını yüklemek için USB (AIR 3.4 ve sonraki sürümlerinde USB desteği mevcuttur) üzerinden iTunes veya ADT kullanabilirsiniz.

AIR uygulamanız bittiğinde, bir dağıtım sertifikası ve ön hazırlık profili kullanarak tekrar paketleyebilirsiniz. Ardından Apple App Store'a gönderilmeye hazır olur.

Mobil uygulama özelliklerini ayarlama

Diğer AIR uygulamalarında olduğu gibi, temel uygulama özelliklerini uygulama tanımlayıcı dosyasında ayarlarsınız. Mobil uygulamalar pencere boyutu ve saydamlık gibi bazı masaüstüne özgü özellikleri yok sayar. Ayrıca mobil uygulamalar kendi platforma özgü özelliklerini de kullanabilirler. Örneğin, Android uygulamaları için `android` öğesini ve iOS uygulamaları için `iPhone` öğesini dahil edebilirsiniz.

Ortak ayarlar

Çeşitli uygulama tanımlayıcısı ayarları tüm mobil aygıt uygulamaları için önemlidir.

Gerekli AIR çalışma zamanı sürümü

Uygulama tanımlayıcı dosyasının ad alanını kullanarak uygulamanızın gerektirdiği AIR çalışma zamanı sürümünü belirtin.

`application` öğesinde atanan ad alanı büyük bir bölümünde uygulamanızın hangi özellikleri kullandığını belirler. Örneğin, uygulamanız AIR 2.7 ad alanını kullanıyorsa ve kullanıcıda yüklü olan sürümü daha sonraki bir sürümse, uygulamanız AIR 2.7 davranışını görmeye devam eder (davranış sonraki sürümde değişmiş olsa bile). Yalnızca ad alanını değiştirdiğinizde ve bir güncelleme yayınladığınızda uygulamanız yeni davranışa ve özelliklere erişir. Güvenlik düzeltmeleri bu kuraldaki önemli istisnalardır.

AIR 3.6 ve önceki sürümlerinde Android gibi çalışma zamanını uygulamadan ayrı kullanan aygıtlarda, gerekli sürüm yoksa kullanıcıdan AIR'i yüklemesi veya yükseltmesi istenir. iPhone gibi çalışma zamanını birlikte kullanan aygıtlarda, bu durum ortaya çıkmaz (bunun sebebi gerekli olan sürümün baştan uygulama ile birlikte paketlenmiş olmasıdır).

Not: *(AIR 3.7 ve sonrası) ADT, varsayılan olarak çalışma zamanını Android uygulamalarla paketler.*

Ad alanını kök `application` öğesinin xmlns niteliğini kullanarak belirtin. Aşağıdaki ad alanları mobil uygulamalar için kullanılmalıdır (hedef aldığınız mobil platforma bağlı olarak):

```
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">
```

Not: *iOS 3 cihazlarının desteği, AIR 2.0 SDK'ye bağlı olarak Packager for iPhone SDK'si tarafından sağlanır. iOS 3 için AIR uygulamaları oluşturmayla ilgili bilgi için bkz. iPhone uygulamaları oluşturma. AIR 2.6 SDK (ve üstü) iPhone 3Gs, iPhone 4 ve iPad cihazlarında iOS 4'ü destekler.*

Daha fazla Yardım konusu

"application" sayfa 206

Uygulama kimliği

Yayınladığınız her uygulama için birkaç ayar benzersiz olmalıdır. Bunlar kimliği, adı ve dosya adını içerir.

Android uygulama kimlikleri

Android'de, kimlik Android paket adına "air" kelimesini AIR kimliğine önek olarak ekleyerek dönüştürülür. Bu nedenle, AIR kimliğiniz *com.example.MyApp* ise Android paket adınız *air.com.example.MyApp* şeklindedir.

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

Ek olarak, kimlik Android işletim sisteminde geçerli bir paket adı değilse geçerli bir ada dönüştürülür. Tire karakterleri alt çizgi olarak değiştirilir ve herhangi bir kimlik bileşenindeki baştaki basamakların önüne büyük "A" getirilir. Örneğin, *3-goats.1-boat* kimliği *air.A3_goats.A1_boat* paket adına dönüştürülür.

Not: *Uygulama kimliğine eklenen önekler Android Market'ta AIR uygulamalarını tanımlamak için kullanılabilir. Uygulamanızın önek nedeniyle bir AIR uygulaması olarak tanımlanmasını istemiyorsanız, APK dosyasının paketini açmanız, uygulama kimliğini değiştirmeniz ve Android için AIR uygulaması analitiğini gerçekleştirmeme bölümünde açıklandığı şekilde yeniden paketlemeniz gerekir.*

iOS uygulama kimlikleri

AIR uygulama kimliğini Apple iOS Provisioning Portal'da oluşturduğunuz uygulama kimliğiyle eşleşecek şekilde ayarlayın.

iOS Uygulama Kimlikleri, paket tanımlayıcısı tarafından takip edilen bir paket çekirdek kimliği içerir. Paket çekirdek kimliği, Apple'ın Uygulama Kimliğine atadığı, 5RM86Z4DJM gibi bir karakter dizesidir. Paket tanımlayıcısı sizin seçtiğiniz ters bir etki alanı stili adı içerir. Paket tanımlayıcısı bir joker uygulama kimliğini gösteren yıldız karakteri (*) ile bitebilir. Paket tanımlayıcısı joker karakter ile bitiyorsa, o joker karakteri herhangi bir geçerli dizeyle değiştirebilirsiniz.

Örneğin:

- Apple uygulama kimliğiniz `5RM86Z4DJM.com.example.helloWorld` ise, uygulama tanımlayıcısında `com.example.helloWorld` kullanmanız gerekir.
- Apple uygulama kimliğiniz `96LPVWEASL.com.example.*` (joker uygulama kimliği) ise, `com.example.helloWorld` veya `com.example.anotherApp` ya da `com.example` ile başlayan başka bir kimlik kullanabilirsiniz.
- Son olarak, Apple uygulama kimliğiniz yalnızca `38JE93KJL.*` paket çekirdek kimliği ve jokerse AIR'de herhangi bir uygulama kimliğini kullanabilirsiniz.

Uygulama kimliği belirtilirken, uygulama kimliğinin paket çekirdeği kimliği kısmını dahil etmeyin.

Daha fazla Yardım konusu

"id" sayfa 222

"filename" sayfa 217

"name" sayfa 230

Uygulama sürümü

AIR 2.5 ve üstünde, uygulama sürümünü `versionNumber` öğesinde belirtin. `version` öğesi artık kullanılamaz. `versionNumber` öğesi için bir değer belirtirken, noktalarla ayrılmış üç numaraya kadar sayı içerebilen, şunun gibi bir sıralama kullanmalısınız: "0.1.2". Sürüm numarasının her bir parçası en fazla üç basamak içerebilir. (Başka bir deyişle, "999.999.999" izin verilen en büyük sürüm sayısıdır.) Sayıda tüm parçaları içermeniz gerekmez; "1" ve "1.0" örnekleri de geçerli sayılardır.

Ayrıca `versionLabel` öğesini kullanan sürüm için de bir etiket belirtebilirsiniz. Bir sürüm etiketi eklediğinizde, Android Uygulama bilgisi ekranı gibi yerlerde sürüm numarasının yerine görüntülenir. Android Market kullanılarak dağıtılan uygulamalar için bir sürüm etiketi belirtilmelidir. AIR uygulama tanımlayıcısında bir `versionLabel` değeri belirtmezseniz, `versionNumber` değeri Android sürüm etiketi alanına atanır.

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

Android'de, AIR `versionNumber` şu formül kullanılarak Android tamsayısı `versionCode` öğesine çevrilir: `a*1000000 + b*1000 + c`. Burada a, b ve c AIR sürüm numarasının bileşenleridir: `a.b.c`.

Daha fazla Yardım konusu

"version" sayfa 238

"versionLabel" sayfa 239

"versionNumber" sayfa 239

Ana uygulama SWF'si

`initialWindow` öğesinin `content` alt öğesinde ana uygulama SWF dosyasını belirtin. Mobil profilde aygıtları hedef aldığınızda, bir SWF dosyası kullanmanız gerekir (HTML tabanlı uygulamalar desteklenmez).

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

Dosyayı AIR paketine dahil etmeniz gerekir (ADT veya IDE'nizi kullanarak) Uygulama tanımlayıcısında ada başvurmak dosyanın otomatik olarak pakete dahil olmasını sağlamaz.

Ana ekran özellikleri

Ana uygulama ekranının ilk görünümünü ve davranışını initialWindow öğesinin çeşitli alt öğeleri denetler.

- **aspectRatio** — Uygulamanın başlangıçta *dikey* biçimde mi (yüksek genişlikten fazla), *yatay* biçimde mi (yükseklik genişlikten az), yoksa *herhangi bir* biçimde mi (sahne alanının yönü otomatik olarak tüm yönlendirmelere ayarlanır) görüntülenmesi gerektiğini belirtir.

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

- **autoOrients** — Kullanıcı aygıtı döndürdüğünde sahne alanının otomatik olarak yönlendirmeyi mi değiştireceğini yoksa kayan klavyeyi açma veya kapatma gibi farklı bir yönlendirmeyle ilişkili bir hareket mi gerçekleştireceğini belirtir. Varsayılan olan *false* olarak ayarlıysa sahne alanı aygıtla birlikte yönlendirmeyi değiştirmez.

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

- **depthAndStencil** — Derinlik veya şablon arabelleğini kullanmayı belirtir. Genelde 3B içeriği ile çalışırken bu arabellekleri kullanırsınız.

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

- **fullScreen** — Aygıtın tam aygıt görünümüne mi geçmesi gerektiğini yoksa ekranı sistem durum çubuğu gibi bir normal işletim sistemi kromuyla paylaşması mı gerektiğini belirtir.

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

- **renderMode** — Çalışma zamanının uygulamayı grafik işlem birimiyle mi (GPU) yoksa ana, merkezi işlem birimiyle mi (CPU) oluşturması gerektiğini belirtir. Genel olarak, GPU ile oluşturma, oluşturma hızını artırır, ancak belirli harmanlama modları ve PixelBender filtreleri gibi bazı özellikler GPU modunda mevcut değildir. Ek olarak, farklı aygıtlar ve farklı aygıt sürücüleri çeşitli GPU özelliklerine ve sınırlamalarına sahiptir. Özellikle GPU modunu kullanırken her zaman uygulamanızı mümkün olan en geniş aygıt çeşitliliğinde test etmelisiniz.

  Görüntü oluşturma modunu *gpu*, *cpu*, *doğrudan* veya *otomatik* olarak ayarlayabilirsiniz. Varsayılan mod şu anda cpu moduna giden *auto* değeridir.

  Not: *Mobil platformlar için AIR ile Flash içeriğinin GPU hızlandırmasını desteklemek üzere Adobe, renderMode="gpu" yerine renderMode="direct" (yani Stage3D) kullanmanızı önerir. Adobe resmi olarak şu Stage3D tabanlı çerçeveleri desteklemekte ve önermektedir: Starling (2D) ve Away3D (3D). Stage3D ve Starling/Away3D hakkında daha fazla ayrıntı için bkz.* http://gaming.adobe.com/getstarted/.

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

  Not: *Arka planda çalışan uygulamalar için renderMode="direct" öğesini kullanamazsınız.*

  GPU modunun sınırlamaları şunlardır:

  - Flex çerçevesi GPU görüntü oluşturma modunu desteklemez.
  - Filtreler desteklenmez
  - PixelBender harmanlamaları ve filtreleri desteklenmez

- Şu harmanlama modları desteklenmez: katman, alfa, silme, kaplama, sert ışık, açıklaştırma ve koyulaştırma
- Video oynatan bir uygulamada GPU görüntüleme modunun kullanılması önerilmez.
- GPU görüntü oluşturma modunda, sanal klavye açıldığında metin alanları düzgün bir şekilde görünür bir konuma taşınmaz. Kullanıcı metin girerken metin alanının görünür olduğundan emin olmak için metin alanını görünür bir alana taşımak üzere sahne alanının softKeyboardRect özelliğini ve yazılım klavyesi olaylarını kullanın.
- Bir görüntüleme nesnesi GPU ile oluşturulamıyorsa hiç görüntülenmez. Örneğin, görüntüleme nesnesine bir filtre uygulanırsa nesne gösterilmez.

Not: *AIR 2.6 ve üstündeki iOS için GPU uygulama işlemi önceki AIR 2.0 sürümünde kullanılan uygulamadan çok farklıdır. Farklı en iyileştirme noktaları geçerlidir.*

Daha fazla Yardım konusu

"aspectRatio" sayfa 209

"autoOrients" sayfa 210

"depthAndStencil" sayfa 213

"fullScreen" sayfa 221

"renderMode" sayfa 232

Desteklenen profiller

Uygulamanızın hangi aygıt profillerini desteklediğini belirtmek için `supportedProfiles` öğesini kullanabilirsiniz. Mobil aygıtlar için mobileDevice profilini kullanın. Uygulamanızı Adobe Hata Ayıklama Başlatıcısı (ADL) ile çalıştırdığınızda, ADL etkin profil olarak listedeki ilk profili kullanır. Ayrıca desteklenenler listesinde belirli bir profili seçmek için ADL'yi çalıştırırken `-profile` bayrağını da kullanabilirsiniz. Uygulamanız tüm profillerde çalışıyorsa `supportedProfiles` öğesini tamamen dışarıda bırakabilirsiniz. ADL bu durumda varsayılan etkin profil olarak masaüstü profilini kullanır.

Uygulamanızın hem mobil aygıtı hem de masaüstü profillerini desteklediğini belirtmek ve normal olarak uygulamayı mobil profilde test etmek istediğinizde aşağıdaki öğeyi ekleyin:

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

Daha fazla Yardım konusu

"supportedProfiles" sayfa 236

"Aygıt profilleri" sayfa 242

"AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157

Gerekli yerel uzantılar

`mobileDevice` profilini destekleyen uygulamalar yerel uzantıları kullanabilir.

AIR uygulamasının uygulama tanımlayıcısında kullandığı tüm yerel uzantıları bildirin. Aşağıdaki örnek, gerekli iki yerel uzantıyı belirtmeye yönelik sözdizimini gösterir:

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

`extensionID` öğesi uzantı tanımlayıcı dosyasındaki `id` öğesiyle aynı değere sahiptir. Uzantı tanımlayıcı dosyası extension.xml adlı bir XML dosyasıdır. Bu sözdizimi yerel uzantı geliştiricisinden aldığınız ANE dosyasında paketlenir.

Sanal klavye davranışı

Çalışma zamanının, sanal klavye açıldıktan sonra odaklanılmış metin girişi alanının görünümde olduğundan emin olmak için kullandığı otomatik kaydırma ve yeniden boyutlandırma davranışını devre dışı bırakmak için `softKeyboardBehavior` öğesini `none` olarak ayarlayın. Otomatik davranışı devre dışı bırakırsanız, klavye kaldırıldıktan sonra metin girişi alanının veya diğer ilgili içeriğin görünür olduğundan emin olmak uygulamanızın sorumluluğundadır. Klavyenin ne zaman açıldığını ve engellediği alanı belirlemek için sahne alanının `softKeyboardRect` özelliğini SoftKeyboardEvent öğesi ile birlikte kullanabilirsiniz.

Otomatik davranışı etkinleştirmek için öğe değerini `pan` olarak ayarlayın:

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

`pan` varsayılan değer olduğundan, `softKeyboardBehavior` öğesini çıkarmak da otomatik klavye davranışını etkinleştirir.

Not: *GPU görüntü oluşturmayı da kullandığınızda kaydırma davranışı desteklenmez.*

Daha fazla Yardım konusu

"softKeyboardBehavior" sayfa 235

Stage.softKeyboardRect

SoftKeyboardEvent

Android ayarları

Android platformunda, Android işletim sistemi tarafından kullanılan bir uygulama özellikleri dosyası olan Android uygulaması bildirimine bilgi eklemek için uygulama tanımlayıcısının `android` öğesini kullanabilirsiniz. ADT genellikle APK paketi oluşturduğunuzda Android Manifest.xml dosyasını otomatik olarak oluşturur. AIR belirli özelliklerin çalışması için gereken değerlerde bazı özellikleri ayarlar. AIR uygulama tanımlayıcısının android bölümünde ayarlanan diğer özellikler Manifest.xml dosyasının karşılık gelen bölümüne eklenir.

Not: *Birçok AIR uygulaması için uygulamanızın ihtiyacı olan Android izinlerini* `android` *öğesi ile ayarlamanız gerekir ancak genellikle başka özellik ayarlamanıza gerek olmaz.*

Yalnızca dize, tamsayı veya boolean değerlerini alan nitelikleri ayarlayabilirsiniz. Başvuruları uygulama paketindeki kaynaklara ayarlama işlemi desteklenmez.

Not: *Çalışma zamanı, 14 veya üzerinde bir SDK sürümü gerektirir. Daha yüksek sürümler için bir uygulama oluşturmak isterseniz, Bildirimin buna uygun olarak doğru sürümle birlikte* `<uses-sdk android:minSdkVersion=""></uses-sdk>` *içerdiğinden emin olmanız gerekir.*

Ayrılmış Android bildirimi ayarları

AIR uygulama ve çalışma zamanı özelliklerinin düzgün çalıştığından emin olmak için oluşturulan Android bildirim belgesinde çeşitli bildirim girişleri ayarlar. Aşağıdaki ayarları tanımlayamazsınız:

bildirim öğesi

Bildirim öğesinin aşağıdaki niteliklerini ayarlayamazsınız:

- package

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

etkinlik öğesi
Ana etkinlik öğesinin aşağıdaki niteliklerini ayarlayamazsınız:

- android:label
- android:icon

uygulama öğesi
Uygulama öğesinin aşağıdaki niteliklerini ayarlayamazsınız:

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

Android izinleri

Android güvenlik modeli, her uygulamanın güvenlik ve gizlilik iması içeren özellikleri kullanmak için izin istemesini gerektirir. Bu izinler uygulama paketlenirken belirtilmelidir ve çalışma zamanında değiştirilemez. Android işletim sistemi, kullanıcı yüklediğinde uygulamanın hangi izinleri istediğini bildirir. Bir özellik için gerekli olan izin istenmezse, uygulamanız özelliğe eriştiğinde Android işletim sistemi bir istisna atabilir ancak bunun garantisi verilmez. İstisnalar uygulamanıza çalışma zamanı tarafından iletilir. Sessiz bir hata oluşması durumunda, Android sistem günlüğüne bir hata mesajı eklenir.

AIR'de, Android izinlerini uygulama tanımlayıcısının `android` öğesinde belirtirsiniz. İzin eklemek için aşağıdaki biçim kullanılır (PERMISSION_NAME öğesinin bir Android izni adı olduğu yerlerde):

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

`manifest` öğesinin içindeki uses-permissions ifadeleri doğrudan Android bildirim belgesine eklenir.

Çeşitli AIR özelliklerini kullanmak için aşağıdaki izinler gerekir:

ACCESS_COARSE_LOCATION Geolocation sınıfı aracılığıyla uygulamanın WIFI ve hücresel ağ konumu verilerine erişmesine izin verir.

ACCESS_FINE_LOCATION Uygulamanın Geolocation sınıfı aracılığıyla GPS verilerine erişmesine izin verir.

ACCESS_NETWORK_STATE ve ACCESS_WIFI_STATE Uygulamanın NetworkInfo sınıfının ağ bilgilerine erişmesine izin verir.

CAMERA Uygulamanın kameraya erişmesine izin verir.

Not: *Kamera özelliğini kullanmak için izin istediğinizde, Android uygulamanızın da kamera gerektirdiğini varsayar. Kamera uygulamanızın isteğe bağlı bir özelliğiyse, kamera bildirimine gerekli niteliği* `false` *olarak ayarlayarak bir* `uses-feature` *özelliği eklemeniz gerekir. Bkz. "Android uyumluluk filtresi" sayfa 76.*

INTERNET Uygulamanın ağ isteği yapmasına izin verir. Ayrıca uzaktan hata ayıklamaya izin verir.

READ_PHONE_STATE AIR çalışma zamanının telefon çağrıları sırasında sesi kapatmasına izin verir. Uygulamanız arka planda ses oynatıyorsa bu izni ayarlamanız gerekir.

RECORD_AUDIO Uygulamanın mikrofona erişmesine izin verir.

WAKE_LOCK ve DISABLE_KEYGUARD Uygulamanın SystemIdleMode sınıfı ayarlarını kullanarak aygıtın uyku moduna girmesini engellemesine izin verir.

WRITE_EXTERNAL_STORAGE Uygulamanın aygıttaki harici bellek kartına yazmasına izin verir.

Örneğin, şaşırtıcı bir şekilde tüm izinleri gerektiren bir uygulamanın izinlerini ayarlamak için aşağıdakini uygulama tanımlayıcıya ekleyebilirsiniz:

```
<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>
```

Daha fazla Yardım konusu

Android Security and Permissions (Android Güvenlik ve İzinler)

Android Manifest.permission sınıfı

Android özel URI şemaları

AIR uygulamasını bir web sayfasından veya yerel bir Android uygulamasından başlatmak için özel bir URI şemasını kullanabilirsiniz. Özel URI desteği Android bildiriminde belirtilen hedef filtrelerine bağlıdır. Bu nedenle bu teknik diğer platformlarda kullanılamaz.

Özel URI kullanmak için `<android>` bloğunda uygulama tanımlayıcısına hedef filtresi ekleyin. Aşağıdaki örnekte iki `intent-filter` öğesi de belirtilmelidir. Özel şemanızın URI dizesini yansıtmak için `<data android:scheme="my-customuri"/>` ifadesini düzenleyin.

```
<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>
```

Hedef filtresi, Android işletim sistemine uygulamanızın belirli bir eylemi gerçekleştirmek için mevcut olduğunu bildirir. Özel URI durumunda, bu kullanıcının URI şemasını kullanarak bir bağlantıyı tıklattığı anlamına gelir (ve tarayıcı bu durumu nasıl işleyeceğini bilmiyordur).

Uygulamanız özel bir URI aracılığıyla başlatıldığında NativeApplication nesnesi bir `invoke` olayı gönderir. Sorgu parametreleri de dahil olmak üzere bağlantının URL'si InvokeEvent nesnesinin `arguments` dizisine yerleştirilir. Herhangi bir sayıda hedef filtresi kullanabilirsiniz.

Not: *Bir StageWebView örneğindeki bağlantılar özel bir URI şeması kullanan URL'leri açamaz.*

Daha fazla Yardım konusu

Android hedef filtreleri

Android eylemleri ve kategoriler

Android uyumluluk filtresi

Android işletim sistemi, uygulamanızın belirli bir aygıtla uyumlu olup olmadığını belirlemek için uygulama bildirim dosyasında birkaç öğe kullanır. Bu bilgiyi bildirime eklemek isteğe bağlıdır. Bu öğeleri dahil etmezseniz uygulamanız herhangi bir Android aygıtına yüklenebilir. Ancak, herhangi bir Android aygıtında düzgün çalışmayabilir. Örneğin, kamera uygulaması kameraya sahip olmayan bir telefonda faydalı olmaz.

Filtreleme için kullanabileceğiniz Android bildirim etiketleri şunları içerir:

- supports-screens

- uses-configuration
- uses-feature
- uses-sdk (AIR 3 veya üstünde)

Kamera uygulamaları

Uygulamanız için kamera izni isterseniz, Android uygulamanın otomatik odak ve flaş gibi tüm mevcut kamera özelliklerini gerektirdiğini varsayar. Uygulamanız tüm kamera özelliklerini gerektirmiyorsa veya kamera isteğe bağlı bir özellikse, bu özelliklerin isteğe bağlı olduğunu belirtmek için kameranın çeşitli `uses-feature` öğelerini ayarlamanız gerekir. Aksi takdirde, bir özelliğe sahip olmayan veya kameraya sahip olmayan kullanıcılar Android Market'ta uygulamanızı bulamaz.

Aşağıdaki örnek, kamera için nasıl izin isteneceğini ve tüm kamera özelliklerinin nasıl isteğe bağlı yapılacağını gösterir:

```
<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>
```

Ses kaydı uygulamaları

Ses kaydetmek için izin isterseniz Android uygulamanın mikrofon da gerektirdiğini varsayar. Ses kaydı uygulamanızın isteğe bağlı bir özelliğiyse, mikrofonun gerekli olmadığını belirtmek için bir uses-feature etiketi ekleyebilirsiniz. Aksi takdirde, mikrofonsuz aygıta sahip kullanıcılar Android Market'ta bulamaz.

Aşağıdaki örnek, mikrofon donanımını isteğe bağlı bir özellik yaparken nasıl mikrofon kullanmak için izin istenileceğini gösterir:

```
<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>
```

Daha fazla Yardım konusu

Android Developers: Android Compatibility (Android Geliştiricileri: Android Uyumluluğu)

Android Developers: Android feature name constants (Android Geliştiricileri: Android özelliği ad sabitleri)

Yükleme konumu

Android `manifest` öğesinin `installLocation` niteliğini `auto` veya `preferExternal` olarak ayarlayarak uygulamanızın harici bellek kartında yüklenmesine veya buraya taşınmasına izin verebilirsiniz:

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

Android işletim sistemi, uygulamanızın harici belleğe yükleneceğinin garantisini vermez. Bir kullanıcı ayrıca sistem ayarları uygulamasını kullanarak uygulamayı dahili ve harici bellek arasında taşıyabilir.

Harici belleğe yüklendiğinde bile, uygulama deposu dizininin içerikleri, paylaşılan nesneler ve geçici dosyalar gibi uygulama önbelleği ve kullanıcı verileri dahili bellekte depolanır. Fazla dahili bellek kullanmayı engellemek için uygulama depolama dizinine kaydettiğiniz veriler konusunda seçici olun. Büyük miktardaki veriler `File.userDirectory` veya `File.documentsDirectory` konumları (ikisi de Android'de SD kartın köküne eşlenir) kullanılarak SD Karta kaydedilmelidir.

Bir StageWebView nesnesinde Flash Player ve diğer eklentileri etkinleştirme

Android 3.0 ve üstünde, bir StageWebView nesnesinde eklenti içeriğinin görüntülenebilmesi için uygulama Android uygulama öğesinde donanım hızlandırmayı etkinleştirmelidir. Eklentiyle görüntü oluşturmayı etkinleştirmek için `application` öğesinin `android:hardwareAccelerated` niteliğini `true` olarak ayarlayın:

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

Renk derinliği

AIR 3+

AIR 3 ve üstünde, çalışma zamanı ekranı 32 bit renkler oluşturacak şekilde ayarlar. AIR'nin önceki sürümlerinde çalışma zamanı 16 bit renk kullanır. Uygulama tanımlayıcısının <colorDepth> öğesini kullanarak çalışma zamanına 16 bit rengi kullanma talimatı verebilirsiniz:

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

16 bit renk derinliğini kullanmak oluşturma performansını arttırır, ancak rengin aslına uygunluğu azalabilir.

iOS Ayarları

Yalnızca iOS aygıtları için geçerli olan ayarlar uygulama tanımlayıcısında `<iPhone>` öğesinin içine yerleştirilir. `iPhone` öğesi; alt öğe olarak bir `InfoAdditions` öğesi, `requestedDisplayResolution` öğesi, `Entitlements` öğesi, `externalSwfs` öğesi ve `forceCPURenderModeForDevices` öğesi içerebilir.

`InfoAdditions` öğesi uygulama için Info.plist ayarlar dosyasına eklenen anahtar-değer çiftleri belirtmenize olanak tanır. Örneğin, aşağıdaki değerler uygulamanın durum çubuğu stilini belirlerken uygulamanın sürekli bir Wi-Fi erişimi gerektirmediğini ifade eder.

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

InfoAdditions ayarları bir `CDATA` etiketine eklenmiştir.

`Entitlements` öğesi, uygulamanın Entitlements.plist ayar dosyasına eklenen anahtar-değer çiftlerini belirtmenize olanak tanır. Entitlements.plist ayarları, push bildirimleri gibi belirli iOS özelliklerine uygulama erişimi sağlar.

Info.plist ve Entitlements.plist ayarları hakkında daha ayrıntılı bilgi için Apple geliştirici belgelerine bakın.

iOS'ta arka plan görevlerini destekleme

AIR 3.3

Adobe AIR 3.3 ve daha yüksek sürümleri, belirli arka plan davranışlarını etkinleştirerek iOS'ta çoklu görevleri destekler:

- Ses
- Konum güncellemeleri
- Ağ iletişimi
- Arka planda uygulama yürütme işleminden vazgeçme

Not: *swf sürümü 21 ve önceki sürümlerinde AIR, oluşturma modu direct (doğrudan) ayarlandığında iOS ve Android'de arka planda yürütmeyi desteklemez. Bu kısıtlama nedeniyle, Stage3D temelli uygulamalar ses yürütme, konum güncellemeleri, ağ yükleme veya indirme vb. gibi arka plan görevlerini yürütemez. iOS, arka planda OpenGLES veya çağrı oluşturmaya izin vermez. Arka planda OpenGL çağrıları yapmayı deneyen uygulamalar iOS tarafından sonlandırılır. Android, uygulamaları arka planda OpenGLES çağrıları veya ses yürütme gibi diğer arka plan görevlerini gerçekleştirme konusunda kısıtlamaz. swf sürümü 22 ve sonraki sürümlerinde AIR mobil uygulamaları, renderMode direct (doğrudan) olarak ayarlandığında arka planda yürütme gerçekleştiremez. OpenGLES çağrıları arka planda yapılırsa AIR iOS çalışma zamanı bir ActionScript hatası ile sonuçlanır (3768 - Stage3D API'si arka planda yürütme sırasında kullanılmayabilir). Ancak, yerel uygulamalarına arka planda OpenGLES çağrıları yapma izni verildiğinden Android'de hata olmaz. Mobil kaynağın en iyi şekilde kullanımı için arka planda uygulama yürütülürken oluşturma çağrıları yapmayın.*

Arka plan sesi

Arka planda ses oynatmayı ve kayıt yapmayı etkinleştirmek için `InfoAdditions` öğesine aşağıdaki anahtar-değer çiftini ekleyin:

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

Arka planda konum güncellemeleri

Arka planda konum güncellemelerini etkinleştirmek için `InfoAdditions` öğesine aşağıdaki anahtar-değer çiftini ekleyin:

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

Not: *Konum API'leri pili önemli ölçüde boşalttığından, bu özelliği yalnızca gerektiğinde kullanın.*

Arka planda ağ iletişimi

Uygulamanız, arka planda kısa görevler yürütmek amacıyla `NativeApplication.nativeApplication.executeInBackground` özelliğini `true` olarak ayarlar.

Örneğin uygulamanız bir dosya yükleme işlemi başlatabilir. Ardından kullanıcı başka bir uygulamayı öne getirebilir. Uygulama, bir yükleme tamamlama olayı aldığında `NativeApplication.nativeApplication.executeInBackground` öğesini `false` olarak ayarlayabilir.

iOS, arka plan görevlerine zaman sınırı koyduğundan, `NativeApplication.nativeApplication.executeInBackground` özelliğinin `true` olarak ayarlanması, uygulamanın süresiz olarak çalışacağı garantisini vermez. iOS, arka planda işlemeyi durdurduğunda AIR `NativeApplication.suspend` olayını gönderir.

Arka planda yürütme işleminden vazgeçme

Uygulamanız, aşağıdaki anahtar-değer çiftinin `InfoAdditions` öğesine eklenmesiyle arka planda yürütme işleminden açıkça vazgeçebilir.

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

Ayrılmış iOS InfoAdditions ayarları

AIR, uygulama ve çalışma zamanı özelliklerinin düzgün çalıştığından emin olmak için oluşturulan Info.plist dosyasında çeşitli girişler ayarlar. Aşağıdaki ayarları tanımlayamazsınız:

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

Not: *MinimumOSVersion değerini tanımlayabilirsiniz. MinimumOSVersion tanımı Air 3.3 ve sonraki sürümlerde mevcuttur.*

Farklı iOS aygıtı modellerini destekleme

iPad desteği için `InfoAdditions` öğenize `UIDeviceFamily` öğesi için uygun anahtar-değer ayarlarını dahil edin. `UIDeviceFamily` ayarı bir dize dizisidir. Her dize desteklenen aygıtları tanımlar. `<string>1</string>` ayarı iPhone ve iPod Touch için desteği tanımlar. `<string>2</string>` ayarı, iPad desteğini tanımlar. `<string>3</string>` ayarı, tvOS desteğini tanımlar. Bu dizelerden yalnızca birini belirtirseniz, yalnızca bu aygıt ailesi desteklenir. Örneğin, aşağıdaki ayar sınırları iPad'i destekler:

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

Aşağıdaki ayar iki aygıt ailesini de destekler (iPhone/iPod Touch ve iPad):

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

Ayrıca AIR 3.7 ve sonrasında, CPU oluşturma modunu belirtilen bir aygıt kümesi için zorlamak ve GPU oluşturma modunu kalan iOS aygıtları için etkinleştirmek üzere `forceCPURenderModeForDevices` etiketini kullanabilirsiniz.

Bu etiketi `iPhone` etiketinin alt öğesi olarak ekleyin ve boşlukla ayrılmış bir aygıt modeli ad listesi belirtin. Geçerli aygıt modeli ad listesi için bkz. “forceCPURenderModeForDevices” sayfa 220.

Örneğin, eski iPod'larda, iPhone'larda ve iPad'lerde CPU modunu kullanmak ve diğer tüm aygıtlarda GPU modunu etkinleştirmek için, uygulama tanımlayıcıda aşağıdakileri belirtin:

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

Yüksek çözünürlüklü ekranlar

`requestedDisplayResolution` öğesi uygulamanızın yüksek çözünürlüklü ekranlara sahip iOS aygıtlarında *standard* çözünürlüğü mü yoksa *yüksek* çözünürlük modunu mu kullanması gerektiğini belirtir.

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

Yüksek çözünürlük modunda, yüksek çözünürlüklü ekrandaki her pikseli bireysel olarak ele alabilirsiniz. Standart modda, aygıt ekranı uygulamanıza standart çözünürlüklü ekran olarak görünür. Bu modda tek bir piksel çizmek, yüksek çözünürlüklü ekranda dört pikselin rengini ayarlayacaktır.

Varsayılan değer `standard` değeridir. Unutmayın, iOS aygıtlarını hedeflemek için `requestedDisplayResolution` öğesi, `iPhone` öğesinin alt öğesi olarak kullanılır (`InfoAdditions` öğesi veya `initialWindow` öğesi değil).

Farklı aygıtlarda farklı ayarlar kullanmak istiyorsanız, varsayılan değerinizi `requestedDisplayResolution` öğesinin değeri olarak belirleyin. Karşıt değeri kullanması gereken aygıtları belirtmek için `excludeDevices` niteliğini kullanın. Örneğin, yüksek çözünürlük modu aşağıdaki kodla, bu modu destekleyen tüm aygıtlarda kullanılır (standart modu kullanan 3. nesil iPad'ler hariç):

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

`excludeDevices` niteliği AIR 3.6 ve üst sürümlerinde kullanılabilir.

Daha fazla Yardım konusu

"requestedDisplayResolution" sayfa 233

Renaun Erickson: Developing for both retina and non-retina iOS screens using AIR 2.6 (AIR 2.6'yı kullanarak hem retina hem de retina olmayan iOS ekranları için geliştirme)

iOS özel URI şemaları

Uygulamanızın bir web sayfasındaki bağlantı tarafından veya aygıttaki başka bir yerel uygulama tarafından çağrılmasına izin vermek için özel bir URI şeması kaydedebilirsiniz. Bir URI şemasını kaydetmek için InfoAdditions öğesine bir CFBundleURLTypes anahtarı ekleyin. Aşağıdaki örnek, bir uygulamanın *example://foo* biçimine sahip URL'ler tarafından çağrılmasına izin vermek için *com.example.app* adlı bir URI şeması kaydeder.

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

Uygulamanız özel bir URI aracılığıyla başlatıldığında NativeApplication nesnesi bir `invoke` olayı gönderir. Sorgu parametreleri de dahil olmak üzere bağlantının URL'si InvokeEvent nesnesinin `arguments` dizisine yerleştirilir. Herhangi bir sayıda özel URI şeması kullanabilirsiniz.

***Not:** Bir StageWebView örneğindeki bağlantılar özel bir URI şeması kullanan URL'leri açamaz.*

***Not:** Başka bir uygulama zaten bir şema kaydetmişse, uygulama o URI şeması için kaydedildiğinden, uygulama bunu değiştiremez.*

iOS uyumluluk filtrelemesi

Uygulamanızın yalnızca belirli donanım ve yazılım özelliklerine sahip aygıtlarda kullanılması gerekiyorsa, `InfoAdditions` öğesi içindeki UIRequiredDeviceCapabilities dizisine girişler ekleyin. Örneğin, aşağıdaki giriş bir uygulamanın bir fotoğraf kamerası ve bir mikrofon gerektirdiğini gösterir:

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

Bir aygıtta karşılık gelen özellikler yoksa uygulama yüklenemez. AIR uygulamalarıyla ilgili özellik ayarları şunları içerir:

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

AIR 2.6 ve üstü *armv7* ve *opengles-2* öğelerini gerekli özellikler listesine otomatik olarak ekler.

***Not:** Uygulamanızın kullanması için bu özellikleri uygulama tanımlayıcısına dahil etmeniz gerekmez. UIRequiredDeviceCapabilities ayarlarını yalnızca kullanıcıların uygulamanızı düzgün çalışmayacağı aygıtlara yüklemesini engellemek için kullanın.*

Duraklatma yerine çıkma

Kullanıcı bir AIR uygulamasından başka bir uygulamaya geçtiğinde, uygulama arka plana geçer ve duraklar. Uygulamanızın duraklamak yerine tamamen çıkmasını isterseniz, `UIApplicationExitsOnSuspend` özelliğini `YES` olarak ayarlayın.

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

Harici, yalnızca varlık içeren SWF'leri yükleyerek indirme boyutunu küçültme

AIR 3.7

Uygulamanız tarafından kullanılan SWF'lerin bir alt kümesini paketleyerek ve çalışma zamanında `Loader.load()` yöntemiyle kalan (yalnızca varlık içeren) harici SWF'lerini yükleyerek ilk uygulama indirme boyutunuzu küçültebilirsiniz. Bu özelliği kullanmak için uygulamayı, ADT harici olarak yüklenen SWF dosyalarındaki tüm ActionScript ByteCode (ABC) öğelerini ana uygulama SWF'sine taşıyıp, yalnızca varlık içeren bir SWF dosyası bırakacak şekilde paketlemeniz gerekir. Bunun, bir uygulamanın yüklenmesinin ardından herhangi bir kodun indirilmesini yasaklayan Apple Store kuralıyla uyumlu olması gerekir.

ADT, harici olarak yüklenen SWF'leri (kesilen SWF'ler olarak da adlandırılır) desteklemek için şunları yapar:

- Çalıştırma zamanında yüklenmesi için çizgiyle sınırlandırılmış SWF listesine erişmek üzere `<iPhone>` öğesinin `<externalSwfs>` alt öğesinde belirtilen metin dosyasını okur.

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

- ABC kodunu her bir harici olarak yüklenen SWF'den ana çalıştırılabilir dosyaya aktarır.
- .ipa dosyasındaki harici olarak yüklenen SWF'leri atlar.
- Kesilen SWF'leri `.remoteStrippedSWFs` dizinine kopyalar. Bu SWF'leri bir web sunucusunda barındırdığınızda uygulamanız bunları gerektiğinde çalışma zamanında yükler.

Çalışma zamanında yüklenecek SWF dosyalarını, aşağıdaki örnekte gösterildiği gibi, adlarını metin dosyasında satır başına bir tane gelecek şekilde belirterek gösterin:

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

Belirtilen dosya yolu, uygulama tanımlayıcı dosyasına göre değişir. Ayrıca bu SWF'leri `adt` komutunda varlık olarak belirtmelisiniz.

Not: *Bu özellik yalnızca standart paketleme için geçerlidir. ADT, hızlı paketlemede (örneğin, yorumlayıcı benzeticisini veya hata ayıklamayı kullanarak) kesilen SWF'ler oluşturmaz.*

Örnek kod dahil olmak üzere bu özellikle ilgili daha fazla bilgi için Adobe mühendisi Abhinav Dhandh tarafından yayınlanan External hosting of secondary SWFs for AIR apps on iOS (iOS'ta AIR uygulamaları için ikincil SWF'lerin harici olarak barındırılması) web günlüğüne bakın.

Coğrafi Konum Desteği

Coğrafi Konum desteği için aşağıdaki anahtar-değer çiftlerinden birini `InfoAdditions` öğesine ekleyin:

```
<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>
```

Uygulama simgeleri

Aşağıdaki tablo her mobil platformda kullanılan simge boyutlarını listeler:

Simge boyutu	Platform
29x29	iOS
36x36	Android
40x40	iOS
48x48	Android, iOS
50x50	iOS
57x57	iOS
58x58	iOS
60x60	iOS
72x72	Android, iOS
75x75	iOS
76x76	iOS
80x80	iOS
87x87	iOS
96x96	Android
100x100	iOS
114x114	iOS
120x120	iOS
144x144	Android, iOS
152x152	iOS
167 x 167	iOS
180x180	iOS
192x192	Android
512x512	Android, iOS
1024 x 1024	iOS

Uygulama tanımlayıcı dosyasının simge öğesinde simge dosyalarının yolunu belirtin:

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

Belirli bir boyutta bir simge sağlamazsanız, bir sonraki en büyük boyuttaki simge kullanılır ve sığması için ölçeklenir.

Android'de simgeler

Android'de, uygulama tanımlayıcısında belirtilen simgeler uygulama Başlatıcısı simgesi olarak kullanılır. Uygulama Başlatıcısı simgesi 36x36 piksel, 48x48 piksel, 72x72 piksel, 96x96 piksel, 144x144 piksel ve 192x192 piksel PNG görüntüleri kümesi olarak sağlanmalıdır. Bu simge boyutları, sırasıyla düşük yoğunluklu, orta yoğunluklu ve yüksek yoğunluklu ekranlar için kullanılır.

Geliştiricilerin Google Play Store'a uygulama gönderimi sırasında 512x512 piksel simgeyi göndermesi gerekmektedir.

iOS'deki simgeler

Uygulama tanımlayıcısında tanımlanan simgeler bir iOS uygulaması için aşağıdaki yerlerde kullanılır:

- 29x29 piksel simge— Daha düşük çözünürlüklü iPhone'lar/iPod'lar için projektör arama simgesi ve daha düşük çözünürlüklü iPad'ler için Ayarlar simgesi.
- 40x40 piksel simge— Daha düşük çözünürlüklü iPad'ler için spotlight arama simgesi.
- 48x48 piksel simge—AIR, bu görüntüye bir kenarlık ekler ve onu daha düşük çözünürlüklü iPad'lerde spotlight araması için 50x50 simge olarak kullanır.
- 50x50 piksel simge— Daha düşük çözünürlüklü iPad'ler için spotlight araması.
- 57x57 piksel simge— Daha düşük çözünürlüklü iPhone'lar/iPod'lar için uygulama simgesi.
- 58x58 piksel simge— Retina Ekranlı iPhone'lar/iPod'lar için spotlight simgesi ve Retina Ekranlı iPad'ler için Ayarlar simgesi.
- 60x60 piksel simge— Daha düşük çözünürlüklü iPhone'lar/iPod'lar için uygulama simgesi.
- 72x72 piksel simge (isteğe bağlı)—Daha düşük çözünürlüklü iPad'ler için Uygulama Simgesi.
- 76x76 piksel simge (isteğe bağlı)—Daha düşük çözünürlüklü iPad'ler için Uygulama Simgesi.
- 80x80 piksel simge— Yüksek çözünürlüklü iPhone'lar/iPod'lar/iPad'ler için spotlight araması.
- 100x100 piksel simge— Retina Ekranlı iPad'ler için spotlight araması.
- 114x114 piksel simge— Retina ekranlı iPhone/iPod'lar için Uygulama Simgesi.
- 120x120 piksel simge— Yüksek çözünürlüklü iPhone'lar/iPod'lar için uygulama simgesi.
- 152x152 piksel simge— Yüksek çözünürlüklü iPad'ler için uygulama simgesi.
- 167x167 piksel simge— Yüksek çözünürlüklü iPad Pro için uygulama simgesi.
- 512x512 piksel simge— Daha düşük çözünürlüklü iPhone'lar/iPod'lar/iPad'ler için uygulama simgesi). iTunes bu simgeyi gösterir. 512-piksel PNG dosyası yalnızca uygulamanızın geliştirme sürümlerini test etmek için kullanılır. Son uygulamayı Apple Uygulama Deposu'na gönderdiğinizde, 512 görüntüsünü ayrı olarak, bir JPG dosyası biçiminde gönderirsiniz. Bu dosya IPA'da yer almaz.
- 1024x1024 piksel simge— Retina Ekranlı iPhone'lar/iPod'lar/iPad'ler için uygulama simgesi.

iOS simgeye bir parlaklık efekti verir. Efekti kaynak görüntünüze uygulamanız gerekmez. Bu varsayılan parlak efekti kaldırmak için, aşağıdakini uygulama tanımlayıcı dosyasındaki `InfoAdditions` öğesine ekleyin:

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

Not: *Adobe'nin Apple iOS app store'da bulunan AIR uygulamalarının sayısını takip edebilmesi için uygulama meta verileri iOS'ta uygulama simgelerine png meta verileri olarak eklenir. Uygulamanızın bu simge meta verisi nedeniyle bir AIR uygulaması olarak tanımlanmasını istemiyorsanız IPA dosyasının paketini açmanız, simge meta verisini kaldırmanız ve dosyayı yeniden paketlemeniz gerekir. Bu prosedür iOS için AIR uygulaması analitiğini gerçekleştirmeme adlı makalede açıklanmaktadır.*

Daha fazla Yardım konusu

"icon" sayfa 222

"imageNxN" sayfa 223

Android Developers: Icon Design Guidelines (Android Geliştiricileri: Simge Tasarımı Yönergeleri)

iOS Kullanıcı Arabirimi Kılavuzları: Özel Simge ve Görüntü Oluşturma Kılavuzları

iOS başlatma görüntüleri

Uygulama simgelerine ek olarak, *Default.png* adlı en az bir başlatma görüntüsü sağlamanız gerekir. İsteğe bağlı olarak, farklı başlangıç yönlendirmeleri, farklı çözünürlükler (yüksek çözünürlüklü retina ekranı ve 16:9 boyut oranı da dahil olmak üzere) ve farklı aygıtlar için ayrı başlatma görüntüleri dahil edebilirsiniz. Ayrıca uygulamanız URL ile başlatıldığında kullanılacak farklı başlatma görüntüleri de ayarlayabilirsiniz.

Başlatma görüntüsü dosyalarına uygulama tanımlayıcısında başvurulmaz ve bunların kök uygulama dizinine yerleştirilmeleri gerekir. (Dosyaları bir altdizine *koymayın*.)

Dosya adlandırma şeması

Görüntüye göre aşağıdaki şemaya göre ad verin:

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

Gerekli olan tek bölüm, dosya adının *taban adı* kısmıdır. Bu kısım *Default* (büyük harfle D) veya uygulama tanımlayıcısında yer alan `InfoAdditions` öğesindeki `UILaunchImageFile` anahtarını kullanarak belirttiğiniz addır.

Ekran boyutu değiştiricisi kısmı, standart ekran boyutlarından birinde olmadığı zamanlarda ekranın boyutunu belirler. Bu değiştirici yalnızca iPhone 5 ve iPod touch (5. nesil) gibi 16:9 boyut oranına sahip ekranları olan iPhone ve iPod touch modelleri için geçerlidir. Bu değiştirici için yalnızca `-568h` değeri desteklenir. Bu cihazlar yüksek çözünürlüklü (retina) ekranları desteklediğinden, ekran boyutu değiştiricisi her zaman için `@2x` ölçek değiştiricisine de sahip olan bir görüntüyle kullanılır. Bu cihazlar için varsayılan başlatma görüntüsü adının tamamı `Default-568h@2x.png` şeklindedir.

urischeme kısmı URI şemasını tanımlamak için kullanılan dizedir. Bu kısım yalnızca uygulamanız bir veya daha fazla özel URL şemasını destekliyorsa geçerli olur. Örneğin, uygulamanız `example://foo` gibi bir bağlantıyla çağrılabiliyorsa, başlatma görüntüsü dosya adının şema kısmı olarak `-example` öğesini kullanın.

yönlendirme kısmı, cihazın uygulama çalıştırıldığı zamanki yönlendirmesine bağlı olarak birden fazla başlatma görüntüsü belirlenmesi için bir yol sunar. Bu kısım yalnızca iPad uygulamalarına yönelik görüntüler için geçerlidir. Bu bölüm, uygulama başladığında aygıtın bulunduğu yönlendirmeye işaret eden aşağıdaki değerlerden biri olabilir:

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

Ölçek kısmı yüksek çözünürlüklü (retina) ekranlara yönelik kullanılan başlatma görüntüleri için `@2x` (iPhone 4, iPhone 5 ve iPhone 6 için) veya `@3x` (iPhone 6 plus için) şeklindedir. (Standart çözünürlüklü ekranlar için kullanılan görüntüler için ölçek kısmını atlayın.) iPhone 5 ve iPod touch (5. nesil) gibi daha uzun cihazlara yönelik başlatma görüntüleri için taban adı kısmından sonra ve diğer kısımlardan önce ekran boyutu değiştiricisi olan `-528h` değerini de belirtmeniz gerekir.

cihaz kısmı, elde taşınan cihazlara ve telefonlara yönelik başlatma görüntülerini belirlemek için kullanılır. Bu kısım, uygulamanız tek bir uygulama ikilisi ile hem elde taşınan cihazları hem de tablet bilgisayarları destekleyen bir genel uygulamaysa kullanılır. Olası değer `~ipad` veya `~iphone` şeklinde olmalıdır (hem iPhone için hem de iPod Touch için).

iPhone için yalnızca dikey en boy oranına sahip görüntüleri dahil edebilirsiniz. Ancak, iPhone 6 plus'ta yatay görüntüler de eklenebilir. Standart çözünürlüklü cihazlar için 320 x 480 piksele sahip görüntüler, yüksek çözünürlüklü cihazlar için 640 x 960 piksele sahip görüntüler ve iPhone 5 ve iPod touch (5. nesil) gibi 16:9 boyut oranı cihazlar için 640 x 1136 piksele sahip görüntüler kullanın.

iPad için, aşağıdaki gibi görüntü ekleyebilirsiniz:

- AIR 3.3 ve öncesi — Tam ekran olmayan görüntüler: Hem yatay (normal çözünürlük için 1024 x 748, yüksek çözünürlük için 2048 x 1496) hem de dikey (normal çözünürlük için 768 x 1004, yüksek çözünürlük için 1536 x 2008) en boy oranlı görüntüleri dahil edin.
- AIR 3.3 ve sonrası — Tam ekran görüntüler: Hem yatay (normal çözünürlük için 1024 x 768, yüksek çözünürlük için 2048 x 1536) hem de dikey (normal çözünürlük için 768 x 1024, yüksek çözünürlük için 1536 x 2048) en boy oranlı görüntüleri dahil edin. Bir tam ekran görüntüyü tam ekran olmayan bir uygulama için paketlediğinizde, en üstteki 20 pikselin (yüksek çözünürlük için 40 piksel) durum çubuğuyla kaplandığını unutmayın. Bu alanda önemli bilgiler görüntülemekten kaçının.

Örnekler

Aşağıdaki tablo, olası en geniş aygıt ve yönlendirme aralığını destekleyen ve `example://` şemasını kullanarak URL'lerle başlatılabilecek varsayımsal bir uygulama için dahil edebileceğiniz örnek başlatma görüntüleri setini gösterir:

Dosya adı	**Görüntü boyutu**	**Kullanım**
Default.png	320 x 480	iPhone, standart çözünürlük
Default@2x.png	640 x 960	iPhone, yüksek çözünürlük
Default-568h@2x.png	640 x 1136	iPhone, yüksek çözünürlük, 16:9 boyut oranı
Default-Portrait.png	768 x 1004 (AIR 3.3 ve önceki sürümleri) 768 x 1024 (AIR 3.4 ve sonraki sürümleri)	iPad, dikey yönlendirme
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 ve önceki sürümleri) 1536 x 2048 (AIR 3.4 ve sonraki sürümleri)	iPad, yüksek çözünürlük, dikey yönlendirme
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 ve önceki sürümleri) 768 x 1024 (AIR 3.4 ve sonraki sürümleri)	iPad, baş aşağı dikey yönlendirme

Dosya adı	Görüntü boyutu	Kullanım
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 ve önceki sürümleri) 1536 x 2048 (AIR 3.4 ve sonraki sürümleri)	iPad, yüksek çözünürlük, baş aşağı dikey yönlendirme
Default-Landscape.png	1024 x 768	iPad, sol yatay yönlendirme
Default-LandscapeLeft@2x.png	2048 x 1536	iPad, yüksek çözünürlük, sol yatay yönlendirme
Default-LandscapeRight.png	1024 x 768	iPad, sağ yatay yönlendirme
Default-LandscapeRight@2x.png	2048 x 1536	iPad, yüksek çözünürlük, sağ yatay yönlendirme
Default-example.png	320 x 480	Standart iPhone'da example:// URL
Default-example@2x.png	640 x 960	Yüksek çözünürlüklü iPhone'da example:// URL
Default-example~ipad.png	768 x 1004	iPad'de dikey yönlendirmelerde example:// URL
Default-example-Landscape.png	1024 x 768	iPad'de yatay yönlendirmelerde example:// URL

Bu örnek yalnızca bir yaklaşımı gösterir. Örneğin iPad için `Default.png` görüntüsünü kullanabilir ve `Default~iphone.png` ve `Default@2x~iphone.png` ile iPhone ve iPod için belirli başlatma görüntülerini belirtebilirsiniz.

Ayrıca bkz.

iOS Application Programming Guide: Application Launch Images (iOS Uygulama Programlama Kılavuzu: Uygulama Başlatma Görüntüleri)

İOS aygıtları için pakette başlatma görüntüleri

Aygıtlar	Çözünürlük (piksel)	Başlatma görüntüsü adı	Yönlendirme
iPhone'lar			
iPhone 4 (retina olmayan)	640 x 960	Default~iphone.png	Dikey
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Dikey
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	Dikey
iPhone 6, iPhone7	750 x 1334	Default-375w-667h@2x~iphone.png	Dikey
iPhone 6+, iPhone7+	1242 x 2208	Default-414w-736h@3x~iphone.png	Dikey
iPhone 6+, iPhone7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	Yatay
iPad'ler			
iPad 1, 2	768 x 1024	Default-Portrait~ipad.png	Dikey
iPad 1, 2	768 x 1024	Default-PortraitUpsideDown~ipad.png	Baş aşağı dikey
iPad 1, 2	1024 x 768	Default-Landscape~ipad.png	Sol yatay

iPad 1, 2	1024 x 768	Default-LandscapeRight~ipad.png	Sağ yatay
iPad 3, Air	1536 x 2048	Default-Portrait@2x~ipad.png	Dikey
iPad 3, Air	1536 x 2048	Default-PortraitUpsideDown@2x~ipad.png	Baş aşağı dikey
iPad 3, Air	2048 x 1536	Default-LandscapeLeft@2x~ipad.png	Sol yatay
iPad 3, Air	2048 x 1536	Default-LandscapeRight@2x~ipad.png	Sağ yatay
iPad Pro	2048 x 2732	Default-Portrait@2x.png	Dikey
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Yatay

Sanat kılavuzları

Doğru boyutlarda olduğu sürece, başlatma görüntüsü için istediğiniz resmi oluşturabilirsiniz. Ancak, çoğunlukla ideal olan görüntünüzü uygulamanın ilk haliyle eşleştirmektir. Uygulamanızın başlangıç ekranının bir ekran görüntüsünü alarak böyle bir başlangıç görüntüsü oluşturabilirsiniz:

1. Uygulamanızı iOS aygıtında açın. Kullanıcı arabiriminin ilk ekranı göründüğünde, Home (Ana Sayfa) düğmesini (ekranın altında) basılı tutun). Home (Ana Sayfa) düğmesini basılı tutarken Power/Sleep (Güç/Uyku) düğmesine basın (aygıt menüsünün en üstünde). Bu bir ekran görüntüsü alır ve bu görüntüyü Film Rulosu'na gönderir.
2. iPhoto veya başka bir fotoğraf aktarma uygulamasından fotoğraf aktararak görüntüyü geliştirme bilgisayarınıza aktarın.

Uygulamanız birden çok dile yerelleştirilmişse başlangıç görüntüsüne metin eklemeyin. Başlangıç görüntüsü sabittir ve metin, diğer diller ile eşleşmez.

Ayrıca bkz.

iOS Human Interface Guidelines: Launch images (iOS Kullanıcı Arabirimi Kılavuzları: Başlatma görüntüleri)

Yoksayılan ayarlar

Mobil aygıtlardaki uygulamalar, yerel pencereler veya masaüstü işletim sistemi özellikleri için geçerli olan uygulama ayarlarını yok sayar. Yoksayılan ayarlar şunlardır:

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- resizable
- systemChrome
- title

- transparent
- visible
- width
- x
- y

Mobil AIR uygulamasını paketleme

Mobil aygıt için planlanan bir AIR uygulamasına yönelik uygulama paketini oluşturmak için ADT -package komutunu kullanın. -target parametresi paketin kullanılmak için oluşturulduğu mobil platformu belirtir.

Android paketleri

Android'deki AIR uygulamaları AIR paketi biçimi yerine Android uygulama paketi biçimini (APK) kullanır.

APK hedef türünü kullanarak ADT tarafından üretilmiş paketler Android Market'a gönderilebilecek bir biçimdedir. Android Market, gönderilen uygulamaların kabul edilmesi için karşılanması gereken gerekliliklere sahiptir. Son paketinizi oluşturmadan önce en son gereklilikleri incelemeniz gerekir. Bkz. Android Developers: Publishing on the Market (Android Geliştiricileri: Market'a Yayınlama).

iOS uygulamalarının aksine, Android uygulamanızı imzalamak için normal AIR kodu imzalama sertifikası kullanabilirsiniz ancak Android Market'a bir uygulama göndermek için sertifika, sertifikanın en az 2033'e kadar geçerli olmasını gerektiren Market kurallarına uymalıdır. ADT -certificate komutunu kullanarak bu tür bir sertifika oluşturabilirsiniz.

Uygulamanızın Google pazarından AIR indirmesi gerektirmesine izin vermeyen alternatif bir pazara uygulama göndermek için ADT'nin `-airDownloadURL` parametresini kullanarak alternatif bir indirme URL'si belirtebilirsiniz. AIR çalışma zamanının gerekli sürümüne sahip olmayan bir kullanıcı uygulamanızı başlattığında belirtilen URL'ye yönlendirilir. Daha fazla bilgi için bkz. “ADT package komutu” sayfa 164.

Varsayılan olarak ADT, Android uygulamasını paylaşılan çalışma zamanı ile paketler. Bu nedenle, uygulamayı çalıştırmak için kullanıcı, cihaza ayrı AIR çalışma zamanı yüklemelidir.

Not: *ADT'yi sabit çalışma zamanı kullanan bir APK oluşturmaya zorlamak için* `target apk-captive-runtime` *öğesini kullanın.*

iOS paketleri

iOS'deki AIR uygulamaları yerel AIR biçimi yerine iOS paketi biçimini (IPA) kullanır.

ipa-app-store hedef türünü ve doğru kod imzalama sertifikasını ve ön hazırlık profilini kullanarak ADT tarafından üretilen paketler Apple App Store'a gönderilebilecek biçimdedir. Geçici dağıtım için uygulama paketlemek üzere *ipa-ad-hoc* hedef türünü kullanın.

Uygulamanızı imzalamak için Apple tarafından verilen doğru geliştirici sertifikasını kullanmanız gerekir. Test yapıları oluşturmak için farklı sertifikalar kullanılır ve bunlar daha sonra uygulamanın gönderilmesinden önceki son paketleme için kullanılır.

Ant kullanarak iOS uygulamasının nasıl paketlendiğiyle ilgili bir örnek için bkz. Piotr Walczyszyn: Packaging AIR application for iOS devices with ADT command and ANT script (ADT komutu ve ANT komut dosyasıyla iOS cihazları için AIR uygulaması paketleme)

ADT ile paketleme

AIR SDK 2.6 ve üstü sürümleri hem iOS hem de Android için paketlemeyi destekler. Paketlemeden önce tüm ActionScript, MXML ve uzantı kodları derlenmelidir. Ayrıca, bir kod imzalama sertifikanızın olması gerekir.

ADT komutları ve seçenekleriyle ilgili ayrıntılı bir başvuru için bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163.

Android APK paketleri

APK paketi oluşturma

APK paketi oluşturmak için hedef türünü yayın sürümleri için *apk*, hata ayıklama yapıları için *apk-debug* veya taklitçide çalıştırmak üzere sürüm modu yapıları için *apk-emulator* olarak ayarlayarak ADT package komutunu kullanın.

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

Tüm komutu tek bir satıra yazın; yukarıdaki satır sonları yalnızca okumayı kolaylaştırmak için verilmiştir. Ayrıca, örnek ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu varsayar. (Yardım için bkz. "Path ortam değişkenleri" sayfa 300.)

Komutu uygulama dosyalarını içeren dizinden çalıştırmalısınız. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf ve bir simgeler dizinidir.

Komutu gösterildiği şekilde çalıştırdığınızda ADT sizden anahtar deposu şifresini ister. (Yazdığınız şifre karakterleri görüntülenmez; yazmanız bittiğinde Enter tuşuna basın.)

Not: *Varsayılan olarak, tüm AIR Android uygulamalarının paket adında* `air.` *ön eki bulunur. Bu varsayılan davranıştan vazgeçmek için bilgisayarınızda* `AIR_NOANDROIDFLAIR` *ortam değişkenini true olarak ayarlayın.*

Yerel uzantıları kullanan bir uygulama için APK paketi oluşturma

Yerel uzantılar kullanan bir uygulama için APK paketi oluşturmak üzere normal paketleme seçeneklerinin yanı sıra `-extdir` seçeneğini de ekleyin. Kaynakları/kitaplıkları paylaşan birden çok ANE'nin olduğu durumlarda ADT yalnızca tek bir kaynağı/kitaplığı seçer ve bir uyarı vermeden önce yinelenen girişleri yok sayar. Bu seçenek uygulamanın kullandığı ANE dosyalarını içeren dizini belirtir. Örneğin:

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

AIR çalışma zamanının kendi sürümünü içeren bir APK paketi oluşturma

Hem uygulamayı hem de AIR çalışma zamanının sabit sürümünü içeren bir APK paketi oluşturmak için `apk-captive-runtime` hedefini kullanın. Bu seçenek uygulamanın kullandığı ANE dosyalarını içeren dizini belirtir. Örneğin:

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

Bu tekniğin olası dezavantajları şunlardır:

- Önemli güvenlik düzeltmeleri, Adobe bir güvenlik yamasını yayınladığında kullanıcılar tarafından otomatik olarak kullanılabilir durumda olmaz
- Uygulamanın daha fazla RAM kullanmasına neden olur

***Not:** Çalışma zamanını paketlediğinizde ADT, `INTERNET` ve `BROADCAST_STICKY` izinlerini uygulamanıza ekler. Bu izinleri AIR çalışma zamanı gerektirir.*

Bir hata ayıklama APK paketi oluşturma

Uygulamanın hata ayıklayıcıyla birlikte kullanabileceğiniz bir sürümünü oluşturmak için hedef olarak apk-debug öğesini kullanın ve bağlantı seçenekleri belirtin:

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

-connect bayrağı aygıttaki AIR çalışma zamanına ağ üzerinde uzak hata ayıklayıcıya nerede bağlanacağını söyler. USB üzerinden hata ayıklamak için, bunun yerine hata ayıklama bağlantısı için kullanmak üzere TCP bağlantı noktası belirterek `-listen` bayrağını belirtmeniz gerekir:

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

Çoğu hata ayıklama özelliğinin çalışması için hata ayıklama etkinken uygulama SWF ve SWC'lerini de derlemeniz gerekir. `-connect` ve `-listen` bayraklarının tam açıklaması için bkz. “Hata ayıklayıcı bağlantısı seçenekleri” sayfa 180.

***Not:** Varsayılan olarak ADT, Android uygulamanızı apk-debug hedefiyle paketlerken uygulamayla birlikte AIR çalışma zamanının sabit bir kopyasını paketler. ADT'yi harici çalışma zamanı kullanan bir APK oluşturmaya zorlamak için `AIR_ANDROID_SHARED_RUNTIME` ortam değişkenini `true` olarak ayarlayın.*

Android'de, ağ üzerinde hata ayıklayıcıyı çalıştıran bilgisayara bağlanabilmesi için uygulamanın İnternet'e erişim izni olması gerekir. Bkz. “Android izinleri” sayfa 74.

Android taklitçisinde kullanmak için APK paketi oluşturma

Android taklitçisinde hata ayıklama APK paketi kullanabilirsiniz ancak sürüm modu paketi kullanamazsınız. Taklitçide kullanmak için sürüm modu APK paketi oluşturmak üzere hedef türünü *apk-emulator* olarak ayarlayarak ADT package komutunu kullanın:

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

Örnek, ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu varsayar. (Yardım için bkz. "Path ortam değişkenleri" sayfa 300.)

AIR veya AIRI dosyasından APK paketi oluşturma

Varolan bir AIR veya AIRI dosyasından doğrudan APK paketi oluşturabilirsiniz:

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

AIR dosyası, uygulama tanımlayıcı dosyasında AIR 2.5 (veya üstü) ad alanını kullanmalıdır.

Android x86 platformu için APK paketi oluşturma

AIR 14'ten itibaren `-arch` argümanı Android x86 platformu için bir APK paketlemek üzere kullanılabilir. Örneğin:

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

iOS paketleri

iOS'de, ADT SWF dosyası bayt kodunu ve diğer kaynak dosyaları yerel iOS uygulamasına dönüştürür.

1. Flash Builder, Flash Professional veya bir komut satırı derleyicisini kullanarak SWF dosyasını oluşturun.
2. Bir komut kabuğu veya terminal açarak iPhone uygulamasının proje klasörüne gidin.
3. Ardından aşağıdaki sözdizimini kullanarak IPA dosyasını oluşturmak için ADT aracını kullanın:

```
adt -package
                                    -target [ipa-test | ipa-debug | ipa-app-store | ipa-ad-hoc |
                                    ipa-debug-interpreter | ipa-debug-interpreter-simulator
                                    ipa-test-interpreter | ipa-test-interpreter-simulator]
                                    -provisioning-profile PROFILE_PATH
                                    SIGNING_OPTIONS
                                    TARGET_IPA_FILE
                                    APP_DESCRIPTOR
                                    SOURCE_FILES
                                    -extdir extension-directory
                                    -platformsdk path-to-iossdk or path-to-ios-simulator-sdk
```

adt uygulamasının tam yolunu dahil etmek için başvuruyu `adt` olarak değiştirin. adt uygulaması AIR SDK'nin bin alt dizinine yüklenir.

Oluşturmak istediğiniz iPhone uygulaması türüne uygun olan `-target` seçeneğini belirleyin:

- `-target ipa-test`—Uygulamanın geliştirici iPhone aygıtınızda test edilecek sürümünü hızlı bir şekilde derlemek için bunu seçin. Ayrıca daha da hızlı derleme için `ipa-test-interpreter` seçeneğini veya iOS Simulator'da çalıştırmak için `ipa-test-interpreter-simulator` seçeneğini de kullanabilirsiniz.

- `-target ipa-debug`—Uygulamanın geliştirici iPhone aygıtınızda test edilecek bir hata ayıklama sürümünü hızlı bir şekilde derlemek için bunu seçin. Bu seçenekle, iPhone uygulamasından `trace()` çıktısı almak için bir hata ayıklama oturumu kullanabilirsiniz.

  Hata ayıklayıcısını çalıştıran geliştirici bilgisayarın IP adresini belirtmek için aşağıdaki `-connect` seçeneklerinden (`CONNECT_OPTIONS`) birini kullanabilirsiniz:

  - `-connect`—Uygulama, uygulamayı derlemek için kullanılan geliştirici bilgisayarında wifi yoluyla bir hata ayıklama oturumuna bağlanmaya çalışır.
  - `-connect IP_ADDRESS`—Uygulama, belirtilen IP adresine sahip bilgisayarda wifi yoluyla bir hata ayıklama oturumuna bağlanmaya çalışır. Örneğin:

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

  - `-connect HOST_NAME`—Uygulama, belirtilen ana makine adına sahip bilgisayarda wifi yoluyla bir hata ayıklama oturumuna bağlanmaya çalışır. Örneğin:

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

  `-connect` seçeneği isteğe bağlıdır. Belirtilmezse, sonuç olarak ortaya çıkan hata ayıklama uygulaması ana makinedeki bir hata ayıklayıcıya bağlanmaya çalışmaz. Başka bir seçenek olarak, “USB üzerinden FDB ile uzaktan hata ayıklama” sayfa 104 konusunda açıklanan USB hata ayıklamayı etkinleştirmek için, `-connect` yerine `-listen` öğesini de belirtebilirsiniz.

  Bir hata ayıklama bağlantısı girişimi başarısız olursa, uygulama kullanıcıya hata ayıklama ana makinesinin IP adresini girmesini istediği bir iletişim kutusu sunar. Aygıt wifi'a bağlı değilse bir bağlantı girişimi başarısız olabilir. Aygıt bağlıysa ancak hata ayıklayan ana makinenin güvenlik duvarının arkasında değilse de oluşabilir.

  Ayrıca daha hızlı derleme için `ipa-debug-interpreter` seçeneğini veya iOS Simulator'da çalıştırmak için `ipa-debug-interpreter-simulator` seçeneğini de kullanabilirsiniz.

  Daha fazla bilgi için bkz. “Mobil AIR uygulamasında hata ayıklama” sayfa 98.

- `-hedef ipa-geçici`—Geçici olarak konuşlandırılacak bir uygulama oluşturmak için bunu seçin. Bkz. Apple iPhone geliştirici merkezi
- `-hedef ipa-uygulama-deposu`—IPA dosyasının Apple Uygulama Deposuna konuşlandırılacak son sürümünü oluşturmak için bunu seçin.

`PROFILE_PATH` öğesini uygulamanızın temel hazırlık dosyası yoluyla değiştirin. Ön hazırlık profilleriyle ilgili daha fazla bilgi için bkz. “iOS kurulumu” sayfa 64.

Uygulamanızı iOS Simulator'da çalışacak şekilde oluştururken iOS Simulator SDK'ye işaret etmek için `-platformsdk` seçeneğini kullanın.

`SIGNING_OPTIONS` öğesini iPhone geliştirici sertifikası ve şifresini belirtecek şekilde değiştirin. Aşağıdaki sözdizimini kullanın:

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

P12_FILE_PATH öğesini P12 sertifika dosyanızın yoluyla değiştirin. *PASSWORD* öğesini sertifika şifresiyle değiştirin. (Aşağıdaki örneği inceleyin.) P12 sertifika dosyasıyla ilgili daha fazla bilgi için bkz. “Bir geliştirici sertifikasını P12 anahtar deposu dosyasına dönüştürme” sayfa 194.

Not: *iOS Simulator için paketleme yaparken kendinden imzalı bir sertifika kullanabilirsiniz.*

`UYGULAMA_AÇIKLAYICI`'yı uygulama açıklayıcı dosyasını belirtecek şekilde değiştirin.

KAYNAK_DOSYALAR'ı projeye dahil edilecek diğer varlıklardan önce projenin başlıca SWF dosyasını belirtecek şekilde değiştirin. Flash Professional'daki uygulama ayarları iletişim kutusunda veya özel bir uygulama tanımlayıcı dosyasında tanımladığınız tüm simgelerin yollarını dahil edin. Ayrıca, başlangıç ekranı resim dosyasını, Default.png'yi de ekleyin.

Uygulamanın kullandığı ANE dosyalarını (yerel uzantıları) içeren dizini belirtmek için `-extdir` *`extension-directory`* seçeneğini kullanın. Uygulama yerel uzantılar kullanmıyorsa bu seçeneği dahil etmeyin.

Önemli: *Uygulamanızda `Resources` adında bir alt dizin oluşturmayın. Çalışma zamanı IPA paketi yapısına uyum sağlamak için otomatik olarak bu ada sahip bir klasör oluşturur. Kendi Resources klasörünüzü oluşturmanız kritik bir çakışmaya neden olur.*

Hata ayıklama için bir iOS paketi oluşturma

Test aygıtlarına yüklemek üzere bir iOS paketi oluşturmak için hedef türünü *ios-debug* olarak ayarlayarak ADT package komutunu kullanın. Bu komutu çalıştırmadan önce zaten Apple'dan bir geliştirme kod imzalama sertifikası ve ön hazırlık profili almış olmanız gerekir.

```
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
```

Not: *Ayrıca daha hızlı derleme için `ipa-debug-interpreter` seçeneğini veya iOS Simulator'da çalıştırmak için `ipa-debug-interpreter-simulator` seçeneğini de kullanabilirsiniz*

Tüm komutu tek bir satıra yazın; yukarıdaki satır sonları yalnızca okumayı kolaylaştırmak için verilmiştir. Ayrıca, örnek ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu varsayar. (Yardım için bkz. “Path ortam değişkenleri” sayfa 300.)

Komutu uygulama dosyalarını içeren dizinden çalıştırmalısınız. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf, bir simgeler dizini ve Default.png dosyasıdır.

Uygulamayı Apple tarafından verilen doğru dağıtım sertifikasını kullanarak imzalamanız gerekir; diğer kod imzalama sertifikaları kullanılamaz.

Wifi hata ayıklama için `-connect` seçeneğini kullanın. Uygulama, belirtilen IP veya ana bilgisayar adında çalışan Flash Hata Ayıklayıcısı (FDB) ile bir hata ayıklama oturumu başlatmaya çalışır. USB hata ayıklama için `-listen` seçeneğini kullanın. Öncelikle uygulamayı, ardından da çalışan uygulama için bir hata ayıklama oturumu başlatan FDB'yi başlatabilirsiniz. Daha fazla bilgi için bkz. “Flash hata ayıklayıcıya bağlanma” sayfa 102.

Apple App Store'a göndermek için bir iOS paketi oluşturma

Apple App store'a göndermek üzere bir iOS paketi oluşturmak için hedef türünü *ios-app-store* olarak ayarlayarak ADT package komutunu kullanın. Bu komutu çalıştırmadan önce zaten Apple'dan bir dağıtım kod imzalama sertifikası ve ön hazırlık profili almış olmanız gerekir.

```
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
```

Tüm komutu tek bir satıra yazın; yukarıdaki satır sonları yalnızca okumayı kolaylaştırmak için verilmiştir. Ayrıca, örnek ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu varsayar. (Yardım için bkz. "Path ortam değişkenleri" sayfa 300.)

Komutu uygulama dosyalarını içeren dizinden çalıştırmalısınız. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf, bir simgeler dizini ve Default.png dosyasıdır.

Uygulamayı Apple tarafından verilen doğru dağıtım sertifikasını kullanarak imzalamanız gerekir; diğer kod imzalama sertifikaları kullanılamaz.

Önemli: *Apple, App Store'a uygulamaları yüklemek için Apple Application Loader (Uygulama Yükleyici) programını kullanmanızı gerektirir. Apple Application Loader (Uygulama Yükleyici) programını yalnızca OS X için yayınlar. Bu nedenle, Windows bilgisayar kullanarak iPhone için bir AIR uygulaması geliştirebilseniz bile, uygulamayı App Store'a göndermek için OS X (sürüm 10.5.3 veya üstü) kullanan bir bilgisayara erişiminiz olması gerekir. Application Loader (Uygulama Yükleyici) programını Apple iOS Developer Center'dan alabilirsiniz.*

Geçici dağıtım için iOS paketi oluşturma

Geçici dağıtım için bir iOS paketi oluşturmak üzere hedef türünü *ios-ad-hoc* olarak ayarlayarak ADT package komutunu kullanın. Bu komutu çalıştırmadan önce zaten Apple'dan uygun bir geçici dağıtım kod imzalama sertifikası ve ön hazırlık profili almış olmanız gerekir.

```
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
```

Tüm komutu tek bir satıra yazın; yukarıdaki satır sonları yalnızca okumayı kolaylaştırmak için verilmiştir. Ayrıca, örnek ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu varsayar. (Yardım için bkz. "Path ortam değişkenleri" sayfa 300.)

Komutu uygulama dosyalarını içeren dizinden çalıştırmalısınız. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf, bir simgeler dizini ve Default.png dosyasıdır.

Uygulamayı Apple tarafından verilen doğru dağıtım sertifikasını kullanarak imzalamanız gerekir; diğer kod imzalama sertifikaları kullanılamaz.

Yerel uzantıları kullanan bir uygulama için iOS paketi oluşturma

Yerel uzantıları kullanan bir uygulama için iOS paketi oluşturmak üzere `-extdir` seçeneği ile ADT paketi komutunu kullanın. ADT komutunu hedef (`ipa-app-store`, `ipa-debug`, `ipa-ad-hoc`, `ipa-test`) için uygun olarak kullanın. Örneğin:

```
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
```

Tüm komutu tek bir satıra yazın; yukarıdaki satır sonları yalnızca okumayı kolaylaştırmak için verilmiştir.

Yerel uzantılarla ilgili olarak, örnek `extensionsDir` adlı dizinin komutu çalıştırdığınız dizinde olduğunu varsayar. `extensionsDir` dizini uygulamanın kullandığı ANE dosyalarını içerir.

Mobil AIR uygulamasında hata ayıklama

Mobil AIR uygulamanızda çeşitli yollarla hata ayıklayabilirsiniz. Uygulama mantığı sorunlarını ortaya çıkarmanın en basit yolu geliştirme bilgisayarınızda ADL veya iOS Simulator kullanarak hata ayıklamaktır. Ayrıca uygulamanızı bir aygıta yükleyebilir ve masaüstü bilgisayarda çalıştırarak Flash hata ayıklayıcı ile uzaktan hata ayıklayabilirsiniz.

ADL kullanarak aygıt benzetimi

Birçok mobil uygulama özelliğini test etmenin ve bunlarda hata ayıklamanın en hızlı ve kolay yolu, Adobe Hata Ayıklama Başlatıcısı (ADL) yardımcı programını kullanarak uygulamanızı geliştirme bilgisayarında çalıştırmaktır. ADL kullanılacak profili belirlemek için uygulama tanımlayıcısında `supportedProfiles` öğesini kullanır. Birden fazla profil listelenmişse ADL listedeki ilk profili kullanır. Ayrıca `supportedProfiles` listesindeki diğer profillerden birini seçmek için ADL'nin `-profile` parametresini de kullanabilirsiniz. (Uygulama tanımlayıcısına bir `supportedProfiles` öğesi dahil etmezseniz, `-profile` argümanı için herhangi bir profil belirtilebilir.) Örneğin, mobil aygıt profilini benzetmek üzere uygulamayı başlatmak için aşağıdaki komutu kullanın:

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

Masaüstünde mobil profilin bu şekilde benzetimini yaparken, uygulama daha çok bir hedef mobil aygıtla eşleşen bir ortamda çalışır. Mobil profilin parçası olmayan ActionScript API'leri mevcut değildir. Ancak, ADL farklı mobil aygıtların özellikleri arasında ayrım yapmaz. Örneğin, gerçek hedef aygıtınız yazılım tuşları kullanmasa bile uygulamanıza benzetimi yapılmış yazılım tuşu basımları gönderebilirsiniz.

ADL menü komutları aracılığıyla aygıt yönlendirmesi değişikliklerinin ve yazılım tuşu girdilerinin benzetimlerini destekler. Mobil aygıt profilinde ADL'yi çalıştırdığınızda, ADL aygıt dönüşü veya yazılım tuşu girdisi girmenize izin veren bir menü (uygulama penceresinde veya masaüstü menü çubuğunda) görüntüler.

Yazılım tuşu girdisi
ADL, mobil aygıttaki Back (Geri), Menu (Menü) ve Search (Ara) düğmeleri için yazılım tuşu düğmelerinin benzetimini yapar. Mobil profil kullanılarak ADL başlatıldığında görüntülenen menüyü kullanarak bu tuşları benzetilen aygıta gönderebilirsiniz.

Aygıt döndürme
ADL, ADL mobil profil kullanılarak başlatıldığında görüntülenen menü ile aygıt döndürmesini benzetmenize izin verir. Benzetilen aygıtı sağa veya sola döndürebilirsiniz.

Dönüş benzetimi yalnızca otomatik yönlendirme sağlayan bir uygulamayı etkiler. Uygulama tanımlayıcısında `autoOrients` öğesini `true` olarak ayarlayarak bu özelliği etkinleştirebilirsiniz.

Ekran boyutu
ADL `-screensize` parametresini ayarlayarak uygulamanızı farklı boyuttaki ekranlarda test edebilirsiniz. Kodu önceden tanımlanmış ekran türlerinden biri için veya normal ve ekranı kaplayan ekranların piksel boyutlarını temsil eden dört değeri içeren dize için iletebilirsiniz.

Dikey mizanpaj için piksel boyutlarını her zaman belirtin. Bu, genişlik değerinin yükseklik değerinden daha küçük olarak belirtilmesi anlamına gelir. Örneğin, aşağıdaki komut Motorola Droid'de kullanılan ekranın benzetimini yapmak için ADL'yi açar:

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Önceden tanımlanmış ekran türleri için bkz. “ADL kullanımı” sayfa 157.

Sınırlamalar

Masaüstü profilinde desteklenmeyen bazı API'lerin ADL tarafından benzetimi yapılamaz. Benzetimi yapılmayan API'ler şunları içerir:

- Accelerometer
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- Bu özellikleri desteklemeyen masaüstü işletim sistemlerindeki çoklu dokunma ve hareketler
- SystemIdleMode

Uygulamanız bu sınıfları kullanıyorsa, özellikleri gerçek bir aygıtta veya taklitçide test etmelisiniz.

Benzer şekilde, masaüstünde ADL altında çalışırken çalışan, ancak tüm mobil aygıt türlerinde çalışmayan API'ler vardır. Bunlar şunları içerir:

- Speex ve AAC ses codec'i
- Erişilebilirlik ve ekran okuyucu desteği
- RTMPE
- ActionScript bayt kodu içeren SWF dosyalarını yükleme
- PixelBender gölgelendiricileri

ADL yürütme ortamını tam olarak kopyalamadığından, bu özellikleri hedef aygıtlarda kullanan uygulamaları test ettiğinizden emin olun.

iOS Simulator kullanan aygıt simülasyonu

iOS Simulator (yalnızca Mac), iOS uygulamalarının çalıştırılması ve hata ayıklama için hızlı bir yöntem sunar. iOS simulator ile test ederken geliştirici sertifikasına veya temel hazırlık profiline ihtiyaç duymazsınız. Yine de, kendinden imzalı dahi olsa bir p12 sertifikası oluşturmanız gerekir.

Varsayılan olarak, ADT her zaman iPhone simülatörünü başlatır. Sanal aygıtı değiştirmek için aşağıdakileri yapın:

- Kullanılabilir simülatörleri görüntülemek için aşağıdaki komutu kullanın.

  ```
  xcrun simctl list devices
  ```

  Çıktı aşağıda gösterilene benzer şekilde görüntülenir.

```
== 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)
....
```

- `AIR_IOS_SIMULATOR_DEVICE` ortam değişkenini aşağıdaki gibi ayarlayarak belirli bir simülatör seçebilirsiniz:

  ```
  export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
  ```

Ortam değişkenini ayarladıktan sonra işlemi yeniden başlatın ve seçtiğiniz simülatör aygıtında uygulamayı çalıştırın.

***Not:** iOS Simulator ile ADT'yi kullanırken her zaman iOS Simulator SDK'nin yolunu belirterek `-platformsdk` seçeneğini dahil etmeniz gerekir.*

iOS Simulator'da bir uygulama çalıştırmak için:

1 Aşağıdaki örnekte gösterildiği şekilde, `-target ipa-test-interpreter-simulator` veya `-target ipa-debug-interpreter-simulator` ile birlikte adt -package komutunu kullanın:

```
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
```

***Not:** Bundan böyle simülatörler kullanılırken imzalama seçenekleri gerekli değildir, bu nedenle ADT'de mevcut olmayacağından `-keystore` bayrağında herhangi bir değer belirtilebilir.*

2 Uygulamayı iOS Simulator'a yüklemek için aşağıdaki örnekte gösterildiği şekilde, adt -installApp komutunu kullanın:

```
adt -installApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -package sample_ipa_name.ipa
```

3 Uygulamayı iOS Simulator'da çalıştırmak için aşağıdaki örnekte gösterildiği şekilde, adt -launchApp komutunu kullanın:

***Not:** Varsayılan olarak, `adt -launchApp` komutu uygulamayı iPhone simülatöründe çalıştırır. Uygulamayı iPad simülatöründe çalıştırmak için `AIR_IOS_SIMULATOR_DEVICE = "iPad"` ortam değişkenini dışa aktarın ve ardından `adt -launchApp` komutunu kullanın.*

```
adt -launchApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -appid sample_ipa_name
```

iOS Simulator'da bir yerel uzantıyı test etmek için, aşağıdaki extension.xml örneğinin gösterdiği şekilde, extension.xml dosyasında `iPhone-x86` platformunu kullanın ve `nativeLibrary` öğesinde `library.a` (statik kütüphane) seçeneğini belirleyin:

```
<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>
```

Not: *iOS Simulator'da bir yerel uzantıyı test ederken, aygıt için derlenen statik kütüphaneyi (`.a` dosyası) kullanmayın. Bunun yerine simulator için derlenen statik kütüphaneyi kullandığınızdan emin olun.*

İzleme ifadeleri

Mobil uygulamanızı masaüstünde çalıştırdığınızda, izleme çıktısı hata ayıklayıcısına veya ADL'yi başlatmak için kullanılan terminal penceresine yazdırılır. Uygulamanızı bir aygıtta veya taklitçide çalıştırdığınızda izleme çıktısını görüntülemek için bir uzaktan hata ayıklama oturumu ayarlayabilirsiniz. Desteklendiği yerlerde, izleme çıktısının aygıt veya işletim sistemi oluşturucusu tarafından sağlanan yazılım geliştirme araçlarını kullanarak da görüntüleyebilirsiniz.

Tüm durumlarda, çalışma zamanının izleme ifadeleri sağlaması için uygulamadaki SWF dosyaları hata ayıklama etkinken derlenmelidir.

Android'de uzak izleme ifadeleri

Android aygıt veya taklitçisinde çalışırken, Android SDK'ye dahil olan Android Debug Bridge (ABD) yardımcı programını kullanarak Android sistem günlüğünde izleme ifadesi çıktısını görebilirsiniz. Uygulamanızın çıktısını görmek için şu kodu geliştirme bilgisayarınızda komut isteminden veya terminal penceresinden çalıştırın:

```
tools/adb logcat air.MyApp:I *:S
```

MyApp, uygulamanızın AIR uygulaması kimliğidir. `*:S` argümanı diğer tüm işlemlerin çıktılarını bastırır. İzleme çıktısına ek olarak uygulamanızla ilgili sistem bilgilerini görmek için logcat filtresi belirtimine ActivityManager öğesini dahil edebilirsiniz.

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

Bu komut örnekleri Android SDK klasöründen ADB'yi çalıştırdığınızı veya `SDK` klasörünü path ortamı değişkenine eklediğinizi varsayar.

Not: *AIR 2.6+ sürümünde ADB yardımcı programı AIR SDK'ye dahildir ve lib/android/bin klasöründe bulunabilir.*

iOS'de uzak izleme ifadeleri

iOS aygıtında çalışan bir uygulamadan izleme ifadelerinin çıktısını görmek için Flash Hata Ayıklayıcı'yı (FDB) kullanarak uzaktan hata ayıklama oturumu kurmanız gerekir.

Daha fazla Yardım konusu

Android Debug Bridge: Enable logcat Logging (Android Debug Bridge: Logcat Günlüğünü Etkinleştirme)

"Path ortam değişkenleri" sayfa 300

Flash hata ayıklayıcıya bağlanma

Mobil aygıtta çalışan bir uygulamada hata ayıklamak için geliştirme bilgisayarınızda Flash hata ayıklayıcıyı çalıştırabilir ve ağ üzerinden bağlanabilirsiniz. Uzaktan hata ayıklamayı etkinleştirmek için aşağıdakileri yapmanız gerekir:

- Android'de uygulama tanımlayıcısında android:permission.INTERNET iznini belirtin.
- Hata ayıklama etkin durumdayken uygulama SWF'lerini derleyin.
- Uygulamayı Android için `-target apk-debug` ile birlikte, iOS için `-target ipa-debug` ile birlikte paketleyin ve `-connect` (wifi hata ayıklama) ya da `-listen` (USB hata ayıklama) bayrağını kullanın.

Wifi üzerinden uzaktan hata ayıklama için aygıtın, IP adresi veya tam nitelikli etki alanı ile Flash hata ayıklayıcısını çalıştıran bilgisayarın TCP bağlantı noktası 7935'e erişmesi gerekir. USB üzerinden uzaktan hata ayıklama için, aygıtın TCP bağlantı noktası 7936'ya veya `-listen` bayrağında belirtilen bağlantı noktasına erişmesi gerekir.

iOS için ayrıca `-target ipa-debug-interpreter` veya `-target ipa-debug-interpreter-simulator` seçeneğini de belirtebilirsiniz.

Flash Professional ile uzaktan hata ayıklama

Uygulama hata ayıklamaya hazır olduğunda ve uygulama tanımlayıcısında izinler ayarlandığında şunu yapın:

1. AIR Android Ayarları iletişim kutusunu açın.
2. Dağıtma sekmesinde:
   - Dağıtım türü için "Aygıtta hata ayıklama" öğesini seçin
   - Yayımlandıktan sonra kısmı için "Uygulamayı bağlı Android aygıtına yükle" öğesini seçin
   - Yayımladıktan sonra kısmı için "Uygulamayı bağlı Android aygıtında başlat" öğesinin seçimini kaldırın
   - Gerekirse yolu Android SDK'ye ayarlayın.
3. Yayınla'yı tıklatın.

   Uygulamanız yüklenir ve aygıtta başlatılır.
4. AIR Android Ayarları iletişim kutusunu kapatın.
5. Flash Professional menüsünden Hata Ayıkla > Uzaktan Hata Ayıklama Oturumunu Başlat > ActionScript 3 öğesini seçin.

   Flash Professional, Çıktı panelinde "Player'ın bağlanması bekleniyor" yazısını görüntüler.
6. Uygulamayı aygıtta başlatın.
7. Adobe AIR bağlantı iletişim kutusunda Flash hata ayıklayıcıyı çalıştıran bilgisayarın IP adresini veya ana bilgisayar adını girin, ardından Tamam'ı tıklatın.

FDB ile ağ üzerinden uzaktan hata ayıklama

Bir aygıtta çalışan uygulamada komut satırı Flash Hata Ayıklayıcı (FDB) ile hata ayıklamak için önce geliştirme bilgisayarınızda hata ayıklayıcıyı çalıştırın ve ardından aygıtta uygulamayı başlatın. Aşağıdaki yöntem aygıtta bir uygulama derlemek, paketlemek ve uygulamada hata ayıklamak için AMXMLC, FDB ve ADT araçlarını kullanır. Örnekler birleştirilmiş Flex ve AIR SDK kullandığınızı ve bin dizininin path ortam değişkeninize dahil olduğunu varsayar. (Bu varsayım yalnızca komut örneklerini basitleştirmek için yapılır.)

1. Terminal veya komut istemi penceresi açın ve uygulamanın kaynak kodunu içeren dizine gidin.
2. Hata ayıklamayı etkinleştirerek uygulamayı amxmlc ile derleyin:

   ```
   amxmlc -debug DebugExample.as
   ```

3. `apk-debug` veya `ipa-debug` hedeflerini kullanarak uygulamayı paketleyin:

   ```
   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
   ```

   Hata ayıklama için her zaman aynı ana bilgisayar adını veya IP adresini kullanıyorsanız, değeri `-connect` bayrağının arkasına koyabilirsiniz. Uygulama, bu IP adresi veya ana bilgisayar adına otomatik olarak bağlanmayı dener. Aksi takdirde, hata ayıklamayı her başlattığınızda bilgileri aygıta girmeniz gerekir.

4. Uygulamayı yükleyin.

   Android'de, ADT `-installApp` komutunu kullanabilirsiniz:

   ```
   adt -installApp -platform android -package DebugExample.apk
   ```

   iOS'ta, ADT `-installApp` komutunu veya iTunes'u kullanarak uygulamayı yükleyebilirsiniz.

5. İkinci bir terminal veya komut penceresinde FDB'yi çalıştırın:

   ```
   fdb
   ```

6. FDB penceresinde `run` komutunu yazın:

   ```
   Adobe fdb (Flash Player Debugger) [build 14159]
                                       Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                       (fdb) run
                                       Waiting for Player to connect
   ```

7. Uygulamayı aygıtta başlatın.
8. Uygulama aygıtta veya taklitçide açıldığında Adobe AIR bağlantı iletişim kutusu açılır. (Uygulamayı paketlerken -connect seçeneğiyle bir ana bilgisayar adı veya IP adresi belirttiyseniz, uygulama o adresi kullanarak otomatik olarak bağlanmayı dener.) Uygun adresi girin ve Tamam'a dokunun.

   Bu modda hata ayıklayıcıya bağlanmak için aygıtın adresi veya ana bilgisayar adını çözümleyebilmesi ve TCP bağlantı noktası 7935'e bağlanması gerekir. Bir ağ bağlantısı gereklidir.

9. Uzak çalışma zamanı hata ayıklayıcıya bağlandığında, FDB `break` komutuyla kesme noktalarını ayarlayabilir ve ardından `continue` komutuyla yürütmeyi başlatabilirsiniz:

```
(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
```

USB üzerinden FDB ile uzaktan hata ayıklama

AIR 2.6 (Android) AIR 3.3 (iOS)

USB bağlantısı üzerinden bir uygulamada hata ayıklamak için uygulamayı `-connect` seçeneği yerine `-listen` seçeneğini kullanarak paketleyebilirsiniz. `-listen` seçeneğini belirlemenizin ardından uygulamayı başlattığınızda çalışma zamanı, TCP bağlantı noktası 7936'daki Flash hata ayıklayıcısını (FDB) bir bağlantı için dinler. Ardından FDB'yi `-p` seçeneğiyle çalıştırırsınız ve FDB de bağlantıyı başlatır.

Android için USB hata ayıklama yordamı

Masaüstü bilgisayarda çalışan Flash hata ayıklayıcısının aygıt veya taklitçide çalışan AIR çalışma zamanına bağlanması için aygıt bağlantı noktasını masaüstü bağlantı noktasına iletmek üzere Android Debug Bridge (ADB - Android SDK yardımcı programı) veya iOS Debug Bridge'i (IDB - AIR SDK yardımcı programı) kullanmanız gerekir.

1. Terminal veya komut istemi penceresi açın ve uygulamanın kaynak kodunu içeren dizine gidin.
2. Hata ayıklamayı etkinleştirerek uygulamayı amxmlc ile derleyin:

   ```
   amxmlc -debug DebugExample.as
   ```

3. İlgili hata ayıklama hedefini (`apk-debug` gibi) kullanarak uygulamayı paketleyin ve `-listen` seçeneğini belirleyin:

   ```
   adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12
   DebugExample.apk DebugExample-app.xml DebugExample.swf
   ```

4. Aygıtı bir USB kablosuyla hata ayıklama bilgisayarına bağlayın. (Bu yöntemi ayrıca bir taklitçide çalışan uygulamada hata ayıklamak için de kullanabilirsiniz. Bu durumda USB bağlantısı gerekli veya mümkün değildir.)
5. Uygulamayı yükleyin.

   ADT `-installApp` komutunu kullanabilirsiniz:

   ```
   adt -installApp -platform android -package DebugExample.apk
   ```

6. TCP bağlantı noktası 7936'yı Android ADB yardımcı programını kullanarak aygıt veya taklitçiden masaüstü bilgisayara iletin:

   ```
   adb forward tcp:7936 tcp:7936
   ```

7. Uygulamayı aygıtta başlatın.
8. Terminal veya komut penceresinde -p seçeneğini kullanarak FDB'yi çalıştırın:

   ```
   fdb -p 7936
   ```

9. FDB penceresinde `run` komutunu yazın:

   ```
   Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
   ```

10 FDB yardımcı programı uygulamaya bağlanmayı dener.

11 Uzak bağlantı kurulduğunda, FDB `break` komutuyla kesme noktalarını ayarlayabilir ve `continue` komutuyla yürütmeyi başlatabilirsiniz:

```
(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
```

***Not:** Bağlantı noktası numarası 7936 hem AIR çalışma zamanı hem de FDB tarafından USB hata ayıklama için varsayılan olarak kullanılır. ADT -listen bağlantı noktası parametresi ve FDB -p bağlantı noktası parametresi ile kullanmak üzere farklı bağlantı noktaları belirtebilirsiniz. Bu durumda ADT'de belirtilen bağlantı noktası numarasını FDB'de belirtilen bağlantı noktası numarasına iletmek için Android Debug Bridge yardımcı programını kullanmanız gerekir: `adb forward tcp:adt_listen_port# tcp:fdb_port#`*

iOS için USB hata ayıklama yordamı

Masaüstünde çalışan Flash hata ayıklayıcısının aygıt veya taklitçide çalışan AIR çalışma zamanına bağlanması için, aygıt bağlantı noktasını masaüstü bağlantı noktasına iletmek üzere iOS Debug Bridge'i (IDB - AIR SDK yardımcı programı) kullanmanız gerekir.

1 Terminal veya komut istemi penceresi açın ve uygulamanın kaynak kodunu içeren dizine gidin.

2 Hata ayıklamayı etkinleştirerek uygulamayı amxmlc ile derleyin:

```
amxmlc -debug DebugExample.as
```

3 İlgili hata ayıklama hedefini (`ipa-debug` veya `ipa-debug-interpreter` gibi) kullanarak uygulamayı paketleyin ve `-listen` seçeneğini belirleyin:

```
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 Aygıtı bir USB kablosuyla hata ayıklama bilgisayarına bağlayın. (Bu yöntemi ayrıca bir taklitçide çalışan uygulamada hata ayıklamak için de kullanabilirsiniz. Bu durumda USB bağlantısı gerekli veya mümkün değildir.)

5 Uygulamayı iOS aygıtınıza yükleyip başlatın. AIR 3.4 ve sonraki sürümlerinde, uygulamayı USB üzerinden yüklemek için `adt -installApp` öğesini kullanabilirsiniz.

6 `idb -devices` komutunu (IDB, *`air_sdk_root`*`/lib/aot/bin/iOSBin/idb` konumundadır) kullanarak aygıt işleyiciyi belirleyin:

```
./idb -devices
                                    List of attached devices
                                    Handle    UUID
                                        1     91770d8381d12644df91fbcee1c5bbdacb735500
```

***Not:** (AIR 3.4 ve sonraki sürümleri) Aygıt işleyiciyi belirlemek için, `idb -devices` yerine `adt -devices` öğesini kullanabilirsiniz.*

7 Masaüstünüzdeki bir bağlantı noktasını bir önceki adımda yer alan IDB yardımcı programını ve Aygıt Kimliği'ni kullanarak, `adt -listen` parametresinde (varsayılan 7936 olmak üzere bu durumda bağlantı noktası 16000'dir) belirtilen bağlantı noktasına iletin:

```
idb -forward 7936 16000 1
```

Bu örnekte, 7936 masaüstü bağlantı noktasıdır, 16000 ise bağlı aygıtın dinlediği bağlantı noktasıdır ve 1 de bağlı aygıtın Aygıt Kimliğidir.

8 Terminal veya komut penceresinde -p seçeneğini kullanarak FDB'yi çalıştırın:

```
fdb -p 7936
```

9 FDB penceresinde `run` komutunu yazın:

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 FDB yardımcı programı uygulamaya bağlanmayı dener.

11 Uzak bağlantı kurulduğunda, FDB `break` komutuyla kesme noktalarını ayarlayabilir ve `continue` komutuyla yürütmeyi başlatabilirsiniz:

Not: *Bağlantı noktası numarası 7936 hem AIR çalışma zamanı hem de FDB tarafından USB hata ayıklama için varsayılan olarak kullanılır. IDB -listen bağlantı noktası parametresi ve FDB -p bağlantı noktası parametresi ile birlikte kullanmak üzere farklı bağlantı noktaları belirtebilirsiniz.*

Mobil aygıtlara AIR ve AIR uygulamaları yükleme

Uygulamanızın son kullanıcıları AIR çalışma zamanını ve AIR uygulamalarını, aygıtlarının normal uygulama ve dağıtım mekanizmalarını kullanarak yükleyebilirler.

Örneğin, Android'de kullanıcılar uygulamaları Android Market'tan yükleyebilir. Ya da Uygulama ayarlarında bilinmeyen kaynaklardan gelen uygulamaların yüklenmesine izin vermişlerse, kullanıcılar bir web sayfasında bağlantıyı tıklatarak veya uygulama paketini aygıtlarına kopyalayıp açarak yükleyebilirler. Bir kullanıcı Android uygulamasını yüklemeyi denerse ancak henüz AIR çalışma zamanını yüklememişse, çalışma zamanını yükleyebileceği Market'a otomatik olarak yönlendirilir.

iOS'de uygulamaları son kullanıcılara dağıtmanın iki yolu vardır. Birincil dağıtım kanalı Apple App Store'dur. Ayrıca sınırlı sayıda kullanıcının App Store'a gitmeden uygulamanızı yüklemesine izin vermek için geçici dağıtımı da kullanabilirsiniz.

AIR çalışma zamanını ve geliştirme uygulamalarını yükleme

Mobil aygıtlarda AIR uygulamaları yerel paketler olarak yüklendiğinden, uygulama yükleme normal platform özelliklerini test amacıyla kullanabilirsiniz. Destekleniyorsa, AIR çalışma zamanını ve AIR uygulamalarını yüklemek için ADT komutlarını kullanabilirsiniz. Şu anda, bu yaklaşım Android'de desteklenmektedir.

iOS'de, iTunes'u kullanarak test için uygulama yükleyebilirsiniz. Test uygulamaları özellikle uygulama geliştirme için verilmiş Apple kod imzalama sertifikasıyla imzalanmalı ve geliştirme ön hazırlık profiliyle paketlenmelidir. Bir AIR uygulaması iOS'de bağımsız bir pakettir. Ayrı bir çalışma zamanı kullanılmaz.

AIR kullanarak AIR uygulamalarını yükleme

AIR uygulamaları geliştirirken, hem çalışma zamanı hem de uygulamalarınızı yüklemek ve kaldırmak için ADT'yi kullanabilirsiniz. (ADT'yi kendiniz çalıştırmak zorunda kalmanızı engellemek için IDE'niz bu komutları entegre edebilir.)

AIR ADT yardımcı programını kullanarak AIR çalışma zamanını bir aygıta veya taklitçiye yükleyebilirsiniz. Aygıt için sağlanan SDK'nin yüklenmesi gerekir. `-installRuntime` komutunu kullanın:

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

`-package` parametresi belirtilmemişse, yüklü AIR SDK'nizde mevcut olanların arasından aygıta veya taklitçiye uygun olan çalışma zamanı paketi seçilir.

Android veya iOS (AIR 3.4 ve üzeri) üzerinde bir AIR uygulaması yüklemek için aynı `-installApp` komutunu kullanın:

```
adt -installApp -platform android -device deviceID -package path-to-app
```

`-platform` argümanı için ayarlanmış değer yükleme yaptığınız aygıtla eşleşmelidir.

Not: *Yeniden yüklemeden önce AIR çalışma zamanının veya AIR uygulamasının varolan sürümleri kaldırılmalıdır.*

iOS aygıtlarında AIR uygulamaları yükleme (iTunes'u kullanarak)

Test amacıyla bir iOS aygıtına AIR uygulaması yüklemek için:

1. iTunes uygulamasını açın.
2. Bunu zaten yaptıysanız, bu uygulamanın ön hazırlık profilini iTunes'a ekleyin. iTunes'ta File (Dosya) > Add To Library (Kütüphaneye Ekle) seçimini yapın. Ardından ön hazırlık profilini (dosya türü olarak mobileprovision öğesine sahip olan) seçin.
3. Bazı iTunes sürümleri uygulamanın aynı versiyonu zaten yüklüyse uygulamayı yenisiyle değiştirmez. Bu durumda, uygulamayı aygıttan ve iTunes içerisindeki uygulamalar listesinden silin.
4. Uygulamaya ait IPA dosyasını çift tıklatın. iTunes'da uygulama listesinde görüntülenmelidir.
5. Aygıtı bilgisayarın USB bağlantı noktasına bağlayın.
6. iTunes'ta aygıtın Uygulama sekmesini kontrol edin ve uygulamanın yüklenecek uygulamalar listesinde seçili olduğundan emin olun.
7. iTunes uygulamasının sol taraftaki listesinden aygıtı seçin. Ardından Sync (Senkr) düğmesine basın. Senkr tamamlandığında, Hello World uygulaması iPhone aygıtınızda görünür.

Yeni sürüm yüklü değilse, aygıtınızdan ve iTunes'taki uygulamalar listesinden silin ve ardından bu işlemi yineleyin. O anda yüklü olan sürüm aynı uygulama kimliği ve sürümüne sahipse bu durum söz konusu olabilir.

Daha fazla Yardım konusu

"ADT installRuntime komutu" sayfa 175

"ADT installApp komutu" sayfa 172

Bir aygıtta AIR uygulamaları çalıştırma

Aygıt kullanıcı arabirimini kullanarak yüklü AIR uygulamalarını başlatabilirsiniz. Desteklendiği yerlerde AIR ADT yardımcı programını kullanarak uygulamaları uzaktan da başlatabilirsiniz:

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

`-appid` argümanının değeri başlatılacak AIR uygulamasının AIR uygulama kimliği olmalıdır. AIR uygulama tanımlayıcısında belirtilen değeri kullanın (paketleme sırasında *air.* öneki eklenmeden).

Yalnızca tek bir aygıt veya taklitçi takılıysa ve çalışıyorsa `-device` bayrağını çıkartabilirsiniz. `-platform` argümanı için ayarlanmış değer yükleme yaptığınız aygıtla eşleşmelidir. Şu anda yalnızca *android* değeri desteklenmektedir.

AIR çalışma zamanını ve uygulamalarını kaldırma

Aygıt işletim sistemi tarafından sağlanan uygulamaları kaldırmak için normal yolları kullanabilirsiniz. Destekleniyorsa, AIR çalışma zamanını ve uygulamalarını kaldırmak için AIR ADT yardımcı programını da kullanabilirsiniz. Çalışma zamanını kaldırmak için `-uninstallRuntime` komutunu kullanın:

```
adt -uninstallRuntime -platform android -device deviceID
```

Bir uygulamayı kaldırmak için `-uninstallApp` komutunu kullanın:

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Yalnızca tek bir aygıt veya taklitçi takılıysa ve çalışıyorsa `-device` bayrağını çıkartabilirsiniz. `-platform` argümanı için ayarlanmış değer yükleme yaptığınız aygıtla eşleşmelidir. Şu anda yalnızca *android* değeri desteklenmektedir.

Bir taklitçi ayarlama

AIR uygulamanızı aygıt taklitçisinde çalıştırmak için, geliştirme bilgisayarınızda bir taklitçi örneği oluşturmak ve çalıştırmak üzere genellikle aygıt SDK'sini kullanmanız gerekir. Ardından taklitçide AIR çalışma zamanının ve AIR uygulamanızın taklitçi sürümünü yükleyebilirsiniz. Taklitçideki uygulamaların gerçek aygıtta olduğundan daha yavaş çalıştığını unutmayın.

Bir Android taklitçisi oluşturma

1. Android SDK ve AVD Manager uygulamasını başlatın:
   - Windows'ta Android SDK dizininin kökünde bulunan SDK Setup.exe dosyasını çalıştırın.
   - Mac OS'de, Android SDK dizininin araçlar alt dizininde bulunan android uygulamasını çalıştırın.
2. Ayarlar seçeneğini belirleyin ve "Force https://" seçeneğini belirleyin.
3. Kullanılabilir Paketler öğesini seçin. Kullanılabilir Android SDK'lerin bir listesini görmelisiniz.
4. Uyumlu bir Android SDK (Android 2.3 veya üstü) seçin ve Seçili Olanı Yükle düğmesini tıklatın.
5. Sanal Aygıtlar seçeneğini belirleyip Yeni düğmesini tıklatın.
6. Aşağıdaki ayarları yapın:
   - Sanal aygıtınız için bir ad
   - Android 2.3, API düzey 8 gibi hedef API
   - SD Kartı için bir boyut (1024 gibi)
   - Bir dış görünüm (Varsayılan HVGA gibi)
7. AVD Oluştur düğmesini tıklatın.

Sanal Aygıt oluşturma işleminin sistem konfigürasyonunuza bağlı olarak biraz zaman alabileceğini unutmayın.

Artık yeni Sanal Aygıt'ı başlatabilirsiniz.

1. AVD Manager uygulamasında Sanal Aygıt'ı seçin. Yukarıda oluşturduğunuz sanal aygıt listelenmelidir.
2. Sanal Aygıt'ı seçin ve Başlat düğmesini tıklatın.

3 Sonraki ekranda Başlat düğmesini tıklatın.

Masaüstünüzde bir taklitçi penceresinin açıldığını görmelisiniz. Bu işlem birkaç saniye sürebilir. Ayrıca Android işletim sisteminin başlaması da biraz zaman alabilir. Bir taklitçide *apk-debug* ve *apk-emulator* ile paketlenmiş uygulamaları yükleyebilirsiniz. *apk* hedefiyle paketlenmiş uygulamalar taklitçide çalışmaz.

Daha fazla Yardım konusu

http://developer.android.com/guide/developing/tools/othertools.html#android

http://developer.android.com/guide/developing/tools/emulator.html

Mobil AIR uygulamalarını güncelleştirme

Mobil AIR uygulamaları yerel paketler olarak dağıtılır. Bu nedenle, platformda diğer uygulamaların standart güncelleme mekanizmalarını kullanın. Genellikle bu, orijinal uygulamayı dağıtmak için kullanılan aynı pazar yerine veya uygulama mağazasına gönderimi içerir.

Mobile AIR uygulamaları AIR Updater sınıfını veya çerçevesini kullanamaz.

Android'de AIR uygulamalarını güncelleme

Android Market'ta dağıtılan uygulamalar için aşağıdakilerin tümü doğru olduğu sürece Market'a yeni bir sürüm yerleştirerek uygulamayı güncelleyebilirsiniz (bu ilkeler AIR tarafından değil Market tarafından uygulanır):

- APK paketi aynı sertifika ile imzalanmış.
- AIR kimliği aynı.
- Uygulama tanımlayıcısındaki `versionNumber` değeri daha büyük. (Kullanılıyorsa `versionLabel` değerini de artırmanız gerekir.)

Android Market'tan uygulamanızı indiren kullanıcılara aygıt yazılımları tarafından güncellemenin mevcut olduğu bildirilir.

Daha fazla Yardım konusu

Android Developers: Publishing Updates on Android Market (Android Geliştiricileri: Android Market'ta Güncellemeleri Yayınlama)

iOS'de AIR uygulamalarını güncelleme

iTunes app store ile dağıtılan AIR uygulamaları için, aşağıdakiler doğru olduğu sürece güncellemeyi mağazaya göndererek uygulamayı güncelleyebilirsiniz (bu ilkeler AIR tarafından değil Apple app store tarafından uygulanır):

- Kod imzalama sertifikası ve temel hazırlık profilleri aynı Apple ID'ye verilir
- IPA paketi aynı Apple Paket Kimliğini kullanır
- Güncelleme desteklenen aygıt havuzunu azaltmaz (başka bir deyişle, orijinal uygulamanız iOS 3'le çalışan aygıtları destekliyorsa, iOS 3 desteğini kaldıran bir güncelleme oluşturamazsınız).

Önemli: *AIR SDK 2.6 ve üstü sürümleri iOS 3'ü desteklemediğinden ve AIR 2 desteklediğinden, AIR 2 kullanılarak geliştirilmiş ve yayınlanmış iOS uygulamalarını AIR 2.6 ve üstü kullanılarak geliştirilmiş bir güncellemeyle güncelleyemezsiniz.*

Push bildirimlerini kullanma

Push bildirimleri, uzak bildirim sağlayıcılarının bir mobil aygıt üzerinde çalışan uygulamalara bildirim göndermesine olanak tanır. AIR 3.4, Apple Push Notification hizmetini (APNs) kullanan iOS aygıtları için push bildirimlerini destekler.

***Not:** Bir AIR for Android uygulaması için push bildirimlerini etkinleştirmek üzere, Adobe evangelisti Piotr Walczyszyn tarafından geliştirilen as3c2dm gibi bir yerel uzantı kullanın.*

Bu bölümün geri kalanında, AIR uygulamasında iOS uygulamaları için push bildirimlerinin nasıl etkinleştirildiği açıklanmaktadır.

***Not:** Bu konuda, Apple geliştirici kimliğine ve iOS geliştirme iş akışı konusunda bilgiye sahip olduğunuz ve bir iOS aygıtı üzerinde en az bir uygulama dağıtmış olduğunuz varsayılır.*

Push bildirimlerine genel bakış

Apple Push Notification hizmeti (APNs), uzak bildirim sağlayıcılarının, iOS aygıtları üzerinde çalışan uygulamalara bildirim göndermesine olanak tanır. APNs şu bildirim türlerini destekler:

- Uyarılar
- Rozetler
- Sesler

APNs hakkında tüm bilgiler için bkz. developer.apple.com.

Uygulamanızda push bildirimlerini kullanmak, birkaç özelliği beraberinde getirir:

- **İstemci uygulama** - Push bildirimlerine kaydolur, uzak bildirim sağlayıcılarıyla iletişim kurar ve push bildirimlerini alır.
- **iOS** - İstemci uygulama ve APNs arasındaki etkileşimi yönetir.
- **APNs** - İstemci kaydı sırasında bir tokenID sağlar ve uzak bildirim sağlayıcılarından gelen bildirimleri iOS'a iletir.
- **Uzak bildirim sağlayıcısı** - tokenId-istemci uygulama bilgilerini saklar ve bildirimleri APNs'ye iletir.

Kayıt iş akışı

Push bildirimlerinin sunucu tarafı bir hizmete kaydedilmesine yönelik iş akışı şu şekildedir:

1. İstemci uygulama, iOS'un push iletme bildirimlerini etkinleştirmesini ister.
2. iOS isteği APNs'ye iletir.
3. APNs sunucusu, tokenId öğesini iOS'a döndürür.
4. iOS, tokenId öğesini istemci uygulamaya döndürür.
5. İstemci uygulama (uygulamaya özgü bir mekanizma kullanarak), tokenId öğesini uzak bildirim sağlayıcısına sağlar ve bu da tokenId öğesini push bildirimleri için saklar.

Bildirim iş akışı

Bildirim iş akışı şu şekildedir:

1. Uzak bildirim sağlayıcısı bir bildirim oluşturur ve bildirim yükünü, tokenId öğesiyle birlikte APNs'ye iletir.
2. APNs, bildirimi aygıt üzerindeki iOS'a iletir.
3. iOS bildirim yükünü uygulamaya iletir.

Push bildirim API'si

AIR 3.4, iOS push bildirimlerini destekleyen bir API kümesini kullanıma sunmuştur. Bu API'ler `flash.notifications` paketindedir ve şu sınıfları içerir:

- `NotificationStyle` - Bildirim türlerine yönelik sabit değerleri tanımlar: `ALERT`, `BADGE` ve `SOUND`.C
- `RemoteNotifier` - Push bildirimlerine abone olmanıza ve bunlara yönelik aboneliğinizi iptal etmenize olanak sağlar.
- `RemoteNotifierSubscribeOptions` - Hangi bildirim türlerinin alınacağını seçmenize olanak sağlar. Çoklu bildirim türlerine kaydolan bir dize vektörü tanımlamak üzere `notificationStyles` özelliğini kullanın.

AIR 3.4 ayrıca `RemoteNotifier` tarafından gönderilen `flash.events.RemoteNotificationEvent` öğesini de şu şekilde içerir:

- Bir uygulamanın aboneliği başarıyla oluşturulduğunda ve APNs'den yeni bir tokenId alındığında.
- Yeni bir uzak bildirim alındığında.

Ayrıca, `RemoteNotifier`, abone olma işlemi sırasında bir hatayla karşılaşırsa `flash.events.StatusEvent` öğesini gönderir.

Bir uygulamada push bildirimlerini yönetme

Uygulamanızı push bildirimlerine kaydetmek için şu adımları gerçekleştirmelisiniz:

- Uygulamanızda push bildirimlerine abone olan bir kod oluşturun.
- Uygulama XML dosyasında push bildirimlerini etkinleştirin.
- iOS Push Hizmetleri'ni etkinleştiren bir temel hazırlık profili ve sertifikası oluşturun.

Aşağıdaki vurgulanmış örnek kod, push bildirimlerine abone olur ve push bildirimi olaylarını işler:

```
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());
}
}
}
```

Uygulama XML dosyasında push bildirimlerini etkinleştirme

Uygulamanızda push bildirimleri kullanmak için, `Entitlements` etiketi içinde aşağıdakileri sağlayın (`iphone` etiketi altında):

```
<iphone>
    ...
        <Entitlements>
            <![CDATA[
                <key>aps-environment</key>
                <string>development</string>
            ]]>
        </Entitlements>
    </iphone>
```

Uygulamayı App Store'a iletmeye hazır olduğunuzda, üretim için geliştirmeye yönelik bir `<string>` öğesi:

```
<string>production</string>
```

Uygulamanız yerelleştirilmiş dizeleri destekliyorsa, aşağıdaki örnekte gösterildiği şekilde `intialWindow` etiketi altında `supportedLanguages` etiketinde dilleri belirtin:

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

iOS Push Hizmetleri'ni etkinleştiren bir temel hazırlık profili ve sertifikası oluşturma

Uygulama-APNs iletişimini etkinleştirmek için, uygulamayı aşağıdaki gibi iOS Push Hizmetleri'ni etkinleştiren bir temel hazırlık profili ve sertifikası ile paketlemeniz gerekir:

1. Apple geliştirici hesabınızda oturum açın.
2. Provisioning Portal'a gidin.
3. App IDs (Uygulama Kimlikleri) sekmesini tıklatın.
4. New App ID (Yeni Uygulama Kimliği) düğmesini tıklatın.
5. Bir açıklama ve paket kimliği belirtin (paket kimliğinde * kullanmamalısınız).

6 Submit (Gönder) öğesini tıklatın. Provisioning Portal, Uygulama Kimliğinizi oluşturur ve App IDs (Uygulama Kimlikleri) sayfasını yeniden görüntüler.

7 Configure (Yapılandır) öğesini tıklatın (Uygulama Kimliğinizin sağında). Configure App ID (Uygulama Kimliğini Yapılandır) sayfası görüntülenir.

8 Enable for Apple Push Notification (Apple Push Notification hizmeti için etkinleştir) onay kutusunu işaretleyin. Biri geliştirme/test için, diğeri ise üretim için olmak üzere iki tür push SSL sertifikası olduğuna dikkat edin.

9 Development Push SSL Certificate (Geliştirme Push SSL Sertifikası) öğesinin sağındaki Configure (Yapılandır) düğmesini tıklatın. Generate Certificate Signing Request (CSR) (Sertifika İmzalama İsteği Oluştur) sayfası görüntülenir.

10 Sayfanın talimatları doğrultusunda, Keychain Access yardımcı programını kullanarak bir CSR oluşturun.

11 SSL sertifikasını oluşturun.

12 SSL sertifikasını indirip yükleyin.

13 (İsteğe Bağlı) Üretim Push SSL sertifikası için 9 -12 arası adımları tekrarlayın.

14 Bitti'yi tıklatın. Configure App ID (Uygulama Kimliğini Yapılandır) sayfası görüntülenir.

15 Bitti'yi tıklatın. App IDs (Uygulama Kimlikleri) sayfası görüntülenir. Uygulama Kimliğinize yönelik Push Bildiriminin yanındaki yeşil daireye dikkat edin.

16 SSL sertifikalarınız daha sonra uygulama ve sağlayıcı iletişimi için kullanılacağından, bunları kaydettiğinizden emin olun.

17 Provisioning Profiles (Temel Hazırlık Profilleri) sayfasını görüntülemek için Provisioning (Temel Hazırlık) sekmesini tıklatın.

18 Yeni Uygulama Kimliğiniz için bir temel hazırlık profili oluşturun ve indirin.

19 Certificates (Sertifikalar) sekmesini tıklatın ve yeni temel hazırlık profili için yeni bir sertifika indirin.

Push bildirimleri için ses kullanma

Uygulamanız için sesli bildirimleri etkinleştirmek üzere, ses dosyalarını tüm diğer varlıkları paketlediğiniz şekilde paketleyin, ancak bunların SWF ve app-xml dosyalarıyla aynı dizinde bulunduğundan emin olun. Örneğin:

```
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, aşağıdaki ses veri formatlarını destekler (aiff, wav veya caf dosyaları içinde):

- Doğrusal PCM
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Yerelleştirilmiş uyarı bildirimleri kullanma

Uygulamanızda yerelleştirilmiş uyarı bildirimleri kullanmak için, yerelleştirilmiş dizeleri lproj klasörleri biçiminde paketleyin. Örneğin, İspanyolca dilinde uyarıları şu şekilde destekleyebilirsiniz:

1 Proje içinde app-xml dosyasıyla aynı düzeyde bir es.lproj klasörü oluşturun.

2 es.lproj klasörünün içinde, - Localizable.Strings adında bir metin dosyası oluşturun.

3 Localizable.Strings dosyasını bir metin düzenleyicide açın ve mesaj anahtarlarını ve ilgili yerelleştirilmiş dizeleri ekleyin. Örneğin:

```
"PokeMessageFormat" = "La notificación de alertas en español."
```

4 Dosyayı kaydedin.

5 Uygulama bu anahtar değerine sahip bir uyarı bildirimi alırsa ve aygıt dili İspanyolca ise, çevrilmiş uyarı metni görüntülenir.

Uzak bildirim sağlayıcısını yapılandırma

Uygulamanıza push bildirimleri göndermek için bir uzak bildirim sağlayıcısına gereksinim duyarsınız. Bu sunucu uygulaması, sağlayıcı gibi davranarak push girdinizi kabul edip bildirimi ve bildirim verilerini APNs'ye iletir ve APNs de push bildirimini bir istemci uygulamaya gönderir.

Bir uzak bildirim sağlayıcısından gelen bildirimleri iletme hakkında ayrıntılı bilgi için, Apple Geliştirici Kütüphanesi'ndeki Provider Communication with Apple Push Notification Service (Apple Push Notification Service ile Sağlayıcı İletişimi) konusuna bakın.

Uzak bildirim sağlayıcısı seçenekleri

Uzak bildirim sağlayıcısına yönelik seçenekler arasında şunlar bulunur:

- APNS-php açık kaynak sunucusunu temel alarak kendi sağlayıcınızı oluşturun. http://code.google.com/p/apns-php/ adresini kullanarak bir PHP sunucusu ayarlayabilirsiniz. Bu Google Code projesi, belirli gereksinimlerinizle örtüşen bir arabirim tasarlamanıza olanak tanır.
- Bir hizmet sağlayıcı kullanın. Örneğin, http://urbanairship.com/, hazır bir APNs sağlayıcısı sunmaktadır. Bu hizmete kaydolduktan sonra, ilk adım olarak, şuna benzer bir kod yardımıyla aygıt belirtecinizi sağlamanız gerekir.

```
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);
                                    }
```

Böylece Urban Airship araçlarını kullanarak test bildirimleri gönderebilirsiniz.

Uzak bildirim sağlayıcısına yönelik sertifikalar

SSL sertifikasını ve özel anahtarını (daha önce oluşturulan), uzak bildirim sağlayıcısının sunucusu üzerindeki uygun konuma kopyalamanız gerekir. Normalde bu iki dosyayı tek bir `.pem` dosyasında birleştirirsiniz. Bunu yapmak için şu adımları gerçekleştirin:

1. Bir terminal penceresi açın.
2. Şu komutu yazarak SSL sertifikasından bir `.pem` dosyası oluşturun:

   ```
   openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
   ```

3. Şu komutu yazarak özel anahtardan (`.p12`) bir .pem dosyası oluşturun:

   ```
   openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
   ```

4. Şu komutu yazarak iki .pem dosyasını tek bir dosyada birleştirin:

   ```
   cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
   ```

5. Birleştirilen `.pem` dosyasını, sunucu tarafı push uygulamanızı oluştururken sunucu sağlayıcısına iletin.

Daha fazla bilgi için Apple Local and Push Notification Programming Guide (Apple Yerel ve Push Bildirim Programlama Kılavuzu) içindeki Installing the SSL Certificate and Key on the Server (SSL Sertifikasını ve Anahtarını Sunucuya Yükleme) konusuna bakın.

Bir uygulamada push bildirimlerini işleme

Bir uygulamada push bildirimlerinin işlenmesi, şunları gerektirir:

- Push bildirimlerine yönelik global kullanıcı yapılandırması ve kabulü
- Ayrı push bildirimlerinin kullanıcı tarafından kabulü
- Push bildirimlerini ve bildirim yük verilerini işleme

Push bildirimlerine yönelik yapılandırma ve kabul

Bir kullanıcı push bildirimi etkin bir uygulamayı ilk kez başlattığında iOS, İzin Verme ve Tamam düğmelerini içeren ***appname*** **Size Push Bildirimleri Göndermek İstiyor** iletişim kutusunu görüntüler. Kullanıcı Tamam'ı seçerse, uygulama, abone olduğu tüm bildirim stillerini alabilir. Kullanıcı İzin Verme'yi seçerse, herhangi bir bildirim almaz.

Not: *Kullanıcılar, her push bildirimleri etkin uygulama için alabileceği belirli bildirim türlerini denetlemek için Ayarlar > Bildirimler öğesine de gidebilir.*

Apple, bir uygulama her etkinleştirildiğinde push bildirimlerine abone olmasını önerir. Uygulamanız `RemoteNotifier.subscribe()` öğesini çağırdığında, bu aygıttaki bir uygulamayı benzersiz şekilde tanımlayan 32 baytlık benzersiz bir sayısal tokenId'ye sahip `token` türünde bir `RemoteNotificationEvent` öğesi alır.

Aygıt bir push bildirimi aldığında, Kapat ve Başlat düğmelerini içeren bir açılır pencere görüntüler. Kullanıcı Kapat öğesine dokunursa hiçbir şey olmaz; ancak Başlat öğesine dokunursa iOS uygulamayı çağırır ve uygulama, aşağıda açıklandığı şekilde, `notification` türünde bir `flash.events.RemoteNotificationEvent` öğesi alır.

Push bildirimlerini ve yük verilerini işleme

Uzak bildirim sağlayıcısı bir aygıta bildirim gönderdiğinde (tokenID öğesini kullanarak), uygulamanızın çalışır durumda olup olmamasından bağımsız olarak `notification` türünde bir `flash.events.RemoteNotificationEvent` alır. Bu noktada, uygulamanız uygulamaya özgü bildirim işleme gerçekleştirir. Uygulamanız bildirim verilerini işliyorsa, buna JSON formatlı `RemoteNotificationEvent.data` özelliği üzerinden erişirsiniz.

Bölüm 8: Televizyon aygıtları için AIR uygulamaları geliştirme

Televizyonlar için AIR özellikleri

Aygıt Adobe AIR for TV içeriyorsa televizyonlar, dijital video kaydedicileri ve Blu-ray oynatıcılar gibi TV aygıtları için Adobe® AIR® uygulamaları oluşturabilirsiniz. AIR for TV, örneğin, yüksek performanslı video ve grafikler için aygıtın donanım hızlandırıcılarının kullanılmasıyla TV aygıtları için en iyi duruma getirilmiştir.

TV aygıtları için olan AIR uygulamaları HTML değil SWF tabanlı uygulamalardır. AIR for TV uygulamanız, donanım hızlandırıcının yanı sıra "oturma odası" ortamına iyi uyan diğer AIR özelliklerinden de faydalanabilir.

Aygıt profilleri

AIR benzer özelliklere sahip hedef aygıt setlerini tanımlamak için profiller kullanır. AIR for TV uygulamaları için aşağıdaki profilleri kullanın:

- `tv` profili. Bir AIR for TV aygıtını hedef alan AIR uygulamaları için bu profili kullanın.
- `extendedTV` profili. AIR for TV uygulamanız yerel uzantıları kullanıyorsa bu profili kullanın.

Bu profiller için tanımlanan ActionScript özellikleri “Aygıt profilleri” sayfa 242 içinde ele alınmıştır. AIR for TV uygulamalarının belirli ActionScript farklılıkları Adobe Flash Platform için ActionScript 3.0 Başvurusu bölümünde verilmiştir.

TV için AIR profilleriyle ilgili ayrıntılar için bkz. “Desteklenen profiller” sayfa 139.

Donanım hızlandırma

Televizyon aygıtları AIR uygulamanızdaki grafiklerin ve videonun performansını önemli derecede artıran donanım hızlandırıcıları sağlar. Bu donanım hızlandırıcılarından faydalanmak için bkz. “AIR for TV uygulaması tasarımında dikkat edilmesi gerekenler” sayfa 121.

İçerik koruma

AIR for TV, Hollywood gişe bombalarından bağımsız filmlere ve TV dizilerine kadar yüksek kaliteli video içeriğinde zengin tüketici deneyimi oluşturulmasını sağlar. İçerik sağlayıcıları, Adobe araçlarını kullanarak etkileşimli uygulamalar oluşturabilir. Adobe sunucu ürünlerini kendi içerik dağıtım alt yapılarına entegre edebilir veya Adobe'nin ekosistem ortaklarından biriyle çalışabilirler.

İçerik koruma yüksek kaliteli video dağıtımı için önemli bir gereksinimdir. AIR for TV, büyük film stüdyoları da dahil olmak üzere içerik sahiplerinin sıkı güvenlik gereksinimlerini karşılayan bir içerik koruma ve parasallaşma çözümü olan Adobe® Flash® Access™ uygulamasını destekler.

Flash Access aşağıdakileri destekler:

- Video akışa alma ve indirme.
- Reklam destekli iş modellerini, aboneliği, kirayı ve elektronik satışı da içeren çeşitli iş modelleri.
- HTTP Dinamik Akışı, Flash® Media Server'ı kullanarak RTMP (Gerçek Zamanlı Medya Protokolü) üzerinden akış ve HTTP ile aşamalı indirme de dahil olmak üzere farklı içerik sunma teknolojileri.

Ayrıca, AIR for TV daha düşük güvenlik gereksinimleri olan varolan akış çözümleri için RTMP'nin şifreli sürümü olan RTMPE'ye yönelik yerleşik desteğe sahiptir. RTMPE ve ilgili SWF doğrulama teknolojileri Flash Media Server'da desteklenir.

Daha fazla bilgi için Adobe Flash Access konusuna bakın.

Çok kanallı ses

AIR 3'ten itibaren, TV için AIR bir HTTP sunucusundan aşamalı olarak indirilen videolar için çok kanallı sesi desteklemektedir. Bu destek şu codec'leri içerir:

- AC-3 (Dolby Digital)
- E-AC-3 (Gelişmiş Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

Not: *Adobe Flash Media Server'dan akışa alınan videolar için çok kanallı ses desteği henüz mevcut değildir.*

Oyun girdisi

AIR 3'ten itibaren TV için AIR, uygulamaların oyun çubukları, oyun kumandaları ve oyun kontrol cihazları gibi takılı oyun girdisi cihazlarıyla iletişim kurmasına izin veren ActionScript API'lerini destekler. Bu cihazlar oyun girdisi cihazları olarak adlandırılsa da yalnızca oyunlar değil, diğer TV için AIR uygulamaları da cihazları kullanabilir.

Farklı özelliklere sahip geniş bir oyun girdisi cihazı aralığı bulunmaktadır. Bu nedenle, uygulamanın farklı (ve bilinmeme olasılığı olan) oyun girdisi cihazlarıyla sorunsuz çalışmasını sağlamak için API'de cihazlar genelleştirilir.

GameInput sınıfı oyun girdisi ActionScript API'lerine giriş noktasıdır. Daha fazla bilgi için bkz. GameInput.

Stage 3D hızlandırmalı grafik oluşturma

AIR 3'ten itibaren TV için AIR Stage 3D hızlandırmalı grafik oluşturmayı desteklemektedir. Stage3D ActionScript API'leri gelişmiş 2B ve 3B özellikleri sağlayan bir düşük düzey GPU hızlandırmalı API kümesidir. Bu düşük düzey API'ler geliştiricilere önemli performans kazançları için GPU donanım hızlandırmasından faydalanma esnekliğini sağlar. Ayrıca Stage3D ActionScript API'lerini destekleyen oyun motorlarını da kullanabilirsiniz.

Daha fazla bilgi için bkz. Oyun motorları, 3B ve Stage 3D.

Yerel uzantılar

Uygulamanız `extendedTV` profilini hedeflediğinde ANE (AIR yerel uzantısı) paketlerini kullanabilir.

Genellikle, aygıt üreticisi, aksi takdirde AIR tarafından desteklenmeyen aygıt özelliklerine erişim sağlamak için ANE paketleri sağlar. Örneğin, yerel bir uzantı bir televizyondaki kanalları değiştirmenize veya video oynatıcısındaki kayıttan oynatmayı duraklatmanıza olanak verebilir.

ANE paketlerini kullanan bir AIR for TV uygulamasını kullandığınızda, uygulamayı bir AIR dosyası yerine bir AIRN dosyasına paketleyebilirsiniz.

TV için AIR cihazlarının yerel uzantıları her zaman *cihazda paketlenmiş* yerel uygulamalardır. Cihazda paketlenmiş, uzantı kütüphanelerinin TV için AIR cihazında yüklü olduğu anlamına gelir. Uygulama paketinize dahil ettiğiniz ANE paketi *asla* uzantının yerel kütüphanelerini içermez. Bazen yerel uzantının yalnızca ActionScript sürümünü içerir. Bu yalnızca ActionScript sürümü uzantının bir saplaması veya benzeticisidir. Aygıt üreticisi, aygıta yerel kütüphaneler de dahil olmak üzere gerçek uzantıyı yükler.

Yerel uzantılar geliştiriyorsanız aşağıdakileri unutmayın:

- Aygıtları için bir TV için AIR yerel uzantısı oluşturuyorsanız, her zaman aygıt üreticisine danışın.
- Bazı TV için AIR aygıtlarında, yalnızca aygıt üreticisi yerel uzantılar oluşturur.
- Tüm TV için AIR aygıtlarında, hangi yerel uzantıların yükleneceğine aygıt üreticisi karar verir.
- TV için AIR yerel uzantıları oluşturmaya yönelik geliştirme araçları üreticiye göre farklılık gösterir.

AIR uygulamanızda yerel uzantıları kullanmayla ilgili daha fazla bilgi için bkz. "Adobe AIR için yerel uzantıları kullanma" sayfa 147.

Yerel uzantı oluşturmayla ilgili daha fazla bilgi için bkz. Adobe AIR için Yerel Uzantılar Geliştirme.

AIR for TV uygulaması tasarımında dikkat edilmesi gerekenler

Video hakkında dikkat edilmesi gerekenler

Video kodlama yönergeleri

Bir TV aygıtına video akışı sağlarken Adobe aşağıdaki kodlama yönergelerini önerir:

Video codec'i:	H.264, Ana veya Yüksek profil, aşamalı kodlama
Çözünürlük:	720i, 720p, 1080i veya 1080p
Kare hızı:	Saniye başına 24 kare veya saniye başına 30 kare
Ses codec'i:	AAC-LC veya AC-3, 44,1 kHz, stereo veya şu çok kanallı ses codec'leri: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio veya DTS-HD Master Audio
Bileşik bit hızı:	kullanılabilir bant genişliğine bağlı olarak en fazla 8M bps
Ses bit hızı:	en fazla 192 Kbps
Piksel en boy oranı:	1 × 1

Adobe AIR for TV aygıtlarına teslim edilen videolar için H.264 codec'ini kullanmanızı önerir.

Not: *AIR for TV ayrıca Sorenson Spark veya On2 VP6 codec'leriyle kodlanan videoları da destekler. Ancak, donanım bu codec'lerin kodunu çözmez veya bunları sunmaz. Bunun yerine, çalışma zamanı bu codec'lerin kodunu yazılım kullanarak çözer ve codec'leri yazılımla sunar. Bu nedenle, video çok daha düşük bir kare hızında oynar. Bu nedenle, mümkün olduğunda H.264 kullanın.*

StageVideo sınıfı

AIR for TV, donanım kodu çözmeyi ve H.264 kodlu videonun sunumunu destekler. Bu özelliği etkinleştirmek için StageVideo sınıfını kullanın.

Aşağıdakilerle ilgili bilgi için *ActionScript 3.0 Geliştirici Kılavuzu* içindeki Donanım hızlandırmalı sunum için StageVideo sınıfını kullanma bölümüne bakın:

- StageVideo sınıfı ve ilgili sınıflar.
- StageVideo sınıfının kullanım sınırlamaları.

AIR for TV, H.264 kodlu video için Video nesnesini kullanan varolan AIR uygulamalarını en iyi şekilde desteklemek için *dahili olarak* StageVideo nesnesini kullanır. Bu, video oynatmanın donanım kodu çözme ve sunma işleminden faydalandığı anlamına gelir. Ancak, Video nesnesi bir StageVideo nesnesiyle aynı kısıtlamaları tabidir. Örneğin, uygulama videoyu döndürmeye çalışırsa, videoyu çalışma zamanı değil donanım sunduğundan dönüş gerçekleşmez.

Ancak, yeni uygulamalar yazdığınızda H.264 kodlu video için StageVideo nesnesini kullanın.

StageVideo sınıfının kullanımıyla ilgili bir örnek için bkz. TV'de Flash Platform için video ve içerik sunma.

Video teslimi yönergeleri

Bir AIR for TV aygıtında, ağın kullanılabilir bant genişliği video oynatımı sırasında değişebilir. Örneğin, bu değişiklikler başka bir kullanıcı aynı İnternet bağlantısını kullanmaya başladığında oluşabilir.

Bu nedenle, Adobe video teslim sisteminizin uyarlanabilir bit hızı özellikleri kullanmasını önerir. Örneğin, sunucu tarafında, Flash Media Server uyarlanabilir bit hızı özelliklerini destekler. İstemci tarafında, Açık Kaynak Medya Çerçevesi'ni (OSMF) kullanabilirsiniz.

Aşağıdaki protokoller bir AIR for TV uygulamasına ağ üzerinden video içeriği sağlamak için kullanılabilir:

- HTTP ve HTTPS Dinamik Akışı (F4F biçimi)
- RTMP, RTMPE, RTMFP, RTMPT ve RTMPTE Akışı
- HTTP ve HTTPS Aşamalı İndirme

Daha fazla bilgi için, şu konulara bakın:

- Adobe Flash Media Server Geliştirici Kılavuzu
- Open Source Media Framework (Açık Kaynak Medya Çerçevesi)

Ses ile ilgili dikkat edilmesi gerekenler

ActionScript ses çalma konusunda, AIR for TV'de diğer AIR uygulamalarında olduğundan farklı değildir. Bilgi için *ActionScript 3.0 Geliştirici Kılavuzu* içindeki Ses ile çalışma bölümüne bakın.

TV için AIR'deki çok kanallı ses desteğiyle ilgili olarak şunları göz önünde bulundurun:

- TV için AIR, bir HTTP sunucusundan aşamalı olarak indirilen videolar için çok kanallı sesi desteklemektedir. Adobe Flash Media Server'dan akışa alınan videolar için çok kanallı ses desteği henüz mevcut değildir.
- AIR for TV uygulaması çok sayıda ses codec'ini desteklese de AIR for TV *aygıtlarının* tümü tüm codec'leri desteklemez. AIR for TV aygıtının AC-3 gibi bir çok kanallı özel ses codec'ini destekleyip desteklemediğini kontrol etmek için flash.system.Capabilities yöntemi `hasMultiChannelAudio()` öğesini kullanın.

  Örneğin, sürekli olarak bir sunucudan bir video dosyası indiren bir uygulama düşünün. Sunucuda çok kanallı farklı ses codec'lerini destekleyen H.264 video dosyaları vardır. Uygulama, sunucudan hangi video dosyasının isteneceğini belirlemek için `hasMultiChannelAudio()` öğesini kullanabilir. Alternatif olarak uygulama, sunucuya `Capabilities.serverString` içinde yer alan dizeyi gönderebilir. Dize, hangi çok kanallı ses codec'lerinin kullanılabilir olduğunu belirterek sunucunun uygun video dosyasını seçmesini sağlar.
- DTS ses codec'lerinden birini kullanırken, `hasMultiChannelAudio()` öğesinin `true` ayarını döndürdüğü, ancak DTS sesinin oynatılmadığı durumlar bulunur.

Örneğin, eski bir amfiye bağlı, S/PDIF çıkışına sahip bir Blu-ray oynatıcısı olduğunu düşünelim. Eski amfi DTS'yi desteklemez, ancak S/PDIF'in Blu-ray oynatıcısına durumu bildirecek herhangi bir protokolü yoktur. Blu-ray oynatıcısı eski amfiye DTS akışını gönderirse, kullanıcı herhangi bir ses duymaz. Bu nedenle, DTS kullanırken en iyi uygulama olarak, sesin çalınmaması durumunda kullanıcının bunu fark edebilmesi için bir kullanıcı arabirimi sağlayın. Böylece, uygulamanız farklı bir codec'e geçebilir.

Aşağıdaki tablo AIR for TV uygulamalarında farklı ses codec'lerinin ne zaman kullanılacağını özetler. Tablo ayrıca AIR for TV aygıtlarının bir ses codec'inin kodunu çözmek için donanım hızlandırıcılarını ne zaman kullanacağını gösterir. Donanım kodu çözme performansı artırır ve CPU'yu boşaltır.

Ses codec'i	**AIR for TV aygıtında kullanılabilirlik**	**Donanım kodu çözme**	**Bu ses codec'i ne zaman kullanılmalı?**	**Daha fazla bilgi**
AAC	Her Zaman	Her Zaman	H.264 ile kodlanmış videolarda. İnternet'ten müzik akışı hizmeti gibi ses akışı uygulamalarında.	Yalnızca ses AAC akışı kullanırken ses akışını bir MP4 kabında kapsülleyin.
mp3	Her Zaman	Hayır	Uygulamanın SWF dosyalarındaki sesler için. Sorenson Spark veya On2 VP6 ile kodlanan videolarda.	Ses için mp3 kullanan bir H.264 video, TV için AIR aygıtlarında kayıttan yürütülmez.
AC-3 (Dolby Digital) E-AC-3 (Gelişmiş Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Kontrol edin	Evet	H.264 ile kodlanmış videolarda.	Normalde AIR for TV, sesin kodunu çözen ve oynatan harici bir ses/video alıcısına çok kanallı bir ses akışı iletir.
Speex	Her Zaman	Hayır	Canlı bir ses akışı alma.	Ses için Speex kullanan bir H.264 video, TV için AIR cihazlarında kayıttan yürütülmez. Speex'i yalnızca Sorenson Spark veya On2 VP6 ile kodlanmış videolarla kullanın.
NellyMoser	Her Zaman	Hayır	Canlı bir ses akışı alma.	Ses için NellyMoser kullanan bir H.264 video, TV için AIR aygıtlarında kayıttan yürütülmez. NellyMoser'ı yalnızca Sorenson Spark veya On2 VP6 ile kodlanmış videolarla kullanın.

Not: *Bazı video dosyaları iki ses akışı içerir. Örneğin, bir video dosyası hem AAC akışını hem AC3 akışını içerebilir. AIR for TV, bu tür video dosyalarını desteklemez ve bu tür bir dosyanın kullanımı video için hiçbir sesin olmamasıyla sonuçlanabilir.*

Grafik donanım hızlandırması

Donanım grafik hızlandırması kullanma

AIR for TV aygıtları 2D grafik işlemleri için donanım hızlandırması sağlar. Aygıtın donanım grafik hızlandırıcıları aşağıdaki işlemleri gerçekleştirmek için İşlemcideki yükü azaltır:

- Bitmap görüntüsü oluşturma

- Bitmap ölçekleme
- Bitmap karıştırma
- Kesintisiz dikdörtgen dolgusu

Bu donanım grafiği hızlandırması AIR for TV uygulamasındaki birçok grafik işleminin yüksek performanslı olabileceği anlamına gelir. Bu işlemlerden bazıları şunlardır:

- Geçişleri kaydırma
- Geçişleri ölçekleme
- Çoğalarak girme ve azalarak çıkma
- Alfa ile birden çok görüntüyü birleştirme

Bu tür işlemler için donanım grafik hızlandırmasının performans yararlarını elde etmek için aşağıdaki tekniklerden birini kullanın:

- MovieClip nesnelerinde ve genellikle değişmeyen içeriğe sahip diğer görüntüleme nesnelerinde `cacheAsBitmap` özelliğini `true` olarak ayarlayın. Ardından bu nesnelerde kayan geçişler, solan geçişler ve alfa karışımları gerçekleştirin.
- Ölçeklemek veya çevirmek (x ve y yeniden konumlandırması uygulamak) istediğiniz görüntüleme nesnelerinde `cacheAsBitmapMatrix` özelliğini kullanın.

  Aygıtın donanım hızlandırıcıları ölçekleme ve çevirme işlemleri için Matrix sınıfı işlemlerini kullanarak işlemleri gerçekleştirir. Alternatif olarak, `cacheAsBitmap` özelliği `true` olarak ayarlı olan bir görüntüleme nesnesinin boyutlarını değiştirdiğiniz bir senaryoyu düşünün. Boyutlar değiştiğinde, çalışma zamanının yazılımı bitmap'i yeniden çizer. Yazılım ile yeniden çizme işlemi, Matris işlemini kullanarak donanım hızlandırmayla ölçekleme işleminden daha düşük performans sunar.

  Örneğin, kullanıcı seçtiğinde genişleyen bir görüntüyü gösteren bir uygulamayı düşünün. Görüntüye genişliyormuş izlenimini vermek için birkaç defa Matris ölçekleme işlemini kullanın. Ancak, orijinal görüntünün ve son görüntünün boyutuna bağlı olarak, son görüntünün kalitesi kabul edilemez olabilir. Bu nedenle, genişletme işlemleri tamamlandıktan sonra görüntüleme nesnesinin boyutlarını sıfırlayın. `CacheAsBitmaptrue` olduğundan, çalışma zamanı yazılımı görüntüleme nesnesini yalnızca bir kez yeniden çizer ve yüksek kalitede görüntü oluşturur.

  Not: *Genellikle, AIR for TV aygıtları donanım hızlandırmalı döndürme ve eğriltme işlemlerini desteklemez. Bu nedenle, Matrix sınıfında döndürme ve eğriltme işlemi belirtirseniz, AIR for TV tüm Matris işlemlerini yazılımda gerçekleştirir. Bu yazılım işlemleri performansı düşürebilir.*
- Özel bir bitmap'leri arabelleğe alma davranışı oluşturmak için BitmapData sınıfını kullanın.

Bitmap önbelleğe alma ile ilgili daha fazla bilgi için şunlara bakın:

- Görüntüleme nesnelerini önbelleğe alma
- Bitmap önbelleğe alma
- Manuel bitmap önbelleğe alma

Grafik belleğini yönetme

Hızlandırılmış grafik işlemlerini gerçekleştirmek için donanım hızlandırıcıları özel grafik belleği kullanır. Uygulamanız tüm grafik belleğini kullanıyorsa, AIR for TV grafik işlemleri için yazılımı kullanmaya döndüğünden uygulama daha yavaş çalışır.

Uygulamanızın grafik belleği kullanımını yönetmek için:

- Bir görüntüyü veya diğer bitmap verisini kullanmayı bitirdiğinizde, bununla ilişkili grafik belleğini serbest bırakın. Bunu yapmak için Bitmap nesnesinin `bitmapData` özelliğinin `dispose()` yöntemini çağırın. Örneğin:

  ```
  myBitmap.bitmapData.dispose();
  ```

  Not: *BitmapData nesnesinin başvurusunun bırakılması, grafik belleğini hemen bırakmaz. Çalışma zamanının çöp toplayıcısı sonunda grafik belleğini boşaltır, ancak `dispose()` yöntemini çağırmak uygulamanıza daha fazla kontrol sağlar.*

- Hedef aygıtınızdaki donanım grafik hızlandırmasını daha iyi anlamak için Adobe'nin sağladığı bir AIR uygulaması olan PerfMaster Deluxe uygulamasını kullanın. Bu uygulama çeşitli işlemleri yürütmek için saniye başına kareyi gösterilir. Aynı işlemin farklı uygulamalarını karşılaştırmak için PerfMaster Deluxe uygulamasını kullanın. Örneğin, bir bitmap görüntüsünü taşıma işlemiyle bir vektör görüntüsünü taşıma işlemini karşılaştırın. PerfMaster Deluxe TV için Flash Platform'da bulunabilir.

Görüntüleme listesini yönetme

Bir görüntüleme nesnesini görünür yapmak için nesnenin `visible` özelliğini `false` olarak ayarlayın. Böylece nesne hala görüntüleme listesinde olur ancak AIR for TV nesneyi oluşturmaz ve görüntülemez. Bu teknik yalnızca küçük bir işleme yükü oluşturduğundan sık sık görünüme girip çıkan nesneler için faydalıdır. Ancak, `visible` özelliğini `false` olarak ayarlamak nesnenin kaynaklarından herhangi birini serbest bırakmaz. Bu nedenle, bir nesneyi görüntülemeniz bittiğinde veya en azından uzun bir süre kullanmayacağınızda nesneyi görüntüleme listesinden kaldırın. Ayrıca nesneye yapılan tüm başvuruları `null` olarak ayarlayın. Bu eylemler çöp toplayıcısının nesnenin kaynaklarını bırakmasına neden olur.

PNG ve JPEG görüntüsü kullanımı

Uygulamalardaki yaygın iki görüntü biçimi PNG ve JPEG'dir. AIR for TV uygulamalarındaki bu görüntü biçimleriyle ilgili olarak aşağıdakileri göz önünde bulundurun:

- AIR for TV genellikle JPEG dosyalarının kodunu çözmek için donanım hızlandırmayı kullanır.
- AIR for TV genellikle PNG dosyalarının kodunu çözmek için yazılım kullanır. PNG dosyalarının kodunu yazılımda çözme işlemi hızlıdır.
- PNG saydamlığı (bir alfa kanalı) destekleyen platformlar arası tek bitmap biçimidir.

Bu nedenle, bu görüntü biçimlerini uygulamalarınızda aşağıdaki gibi kullanın:

- Donanım hızlandırmalı kod çözme işleminden faydalanmak üzere fotoğraflar için JPEG dosyalarını kullanın.
- Kullanıcı arabirimi öğeleri için PNG görüntüsünü kullanın. Kullanıcı arabirimi öğeleri alfa ayarına sahip olabilir ve yazılım kod çözme işlemi kullanıcı arabirimi öğeleri için yeterli derecede hızlı performans sağlar.

AIR for TV uygulamalarında sahne alanı

Bir AIR for TV uygulaması için geliştirme yaparken ve Stage sınıfı ile çalışırken aşağıdakileri dikkate alın:

- Ekran çözünürlüğü
- Güvenli görüntüleme alanı
- Sahne alanı ölçek modu
- Sahne alanı hizalaması
- Sahne alanının görüntüleme durumu

- Birden fazla ekran boyutu için tasarlama
- Sahne alanının kalite ayarı

Ekran çözünürlüğü

Geçerli olarak, TV aygıtlarında genellikle şu ekran çözünürlüklerinden biri bulunur: 540p, 720p ve 1080p. Bu ekran çözünürlükleri ActionScript Capabilities sınıfında aşağıdaki değerlere neden olur:

Ekran çözünürlüğü	`Capabilities.screenResolutionX`	`Capabilities.screenResolutionY`
540p	960	540
720p	1280	720
1080p	1920	1080

Belirli bir aygıta yönelik tam ekranlı AIR for TV uygulaması yazmak için `Stage.stageWidth` ve `Stage.stageHeight` öğelerini doğrudan aygıtın ekran çözünürlüğüne yazın. Ancak, birden çok aygıtta çalışan tam ekran bir uygulama yazmak için Sahne Alanı boyutlarını ayarlamak üzere `Capabilities.screenResolutionX` ve `Capabilities.screenResolutionY` özelliklerini kullanın.

Örneğin:

```
stage.stageWidth = Capabilities.screenResolutionX;
stage.stageHeight = Capabilities.screenResolutionY;
```

Güvenli görüntüleme alanı

Bir televizyondaki *güvenli görüntüleme alanı*, ekranın kenarlarından içeri doğru girmiş alanıdır. Bu alan, son kullanıcının TV'nin kenarındaki kısım alanın herhangi bir bölümünü engellemeden tüm alanı görebilmesini sağlayacak kadar içeri girer. Ekranın etrafındaki fiziksel çerçeve olan kenarlık üreticiden üreticiye değiştiği için içeri girme oranı çeşitlilik gösterir. Güvenli görüntüleme alanı ekranın görünür olan alanını garantilemeye çalışır. Güvenli görüntüleme alanı *güvenli başlık alanı* olarak da bilinir.

Taşma alanı ekranın kenarının arkasında kaldığı için görünmeyen alanıdır.

Adobe, ekranın her kenarında % 7,5 içe doğru giriş önerir. Örneğin:

1920 X 1080 ekran çözünürlüğü için güvenli görüntüleme alanı

Tam ekranlı bir AIR for TV uygulaması tasarlarken her zaman güvenli görüntüleme alanını göz önünde bulundurun:

- Arka plan görüntüleri veya arka plan renkleri gibi arka planlar için tüm ekranı kullanın.
- Metin, grafik, video ve kullanıcı arabirimi öğeleri gibi önemli uygulama öğeleri için yalnızca güvenli görüntüleme alanını kullanın.

Aşağıdaki tablo, % 7,5 girinti kullanarak tipik ekran çözünürlükleri için güvenli görüntüleme alanının boyutlarını gösterir.

Ekran çözünürlüğü	Güvenli görüntüleme alanının genişliği ve yüksekliği	Sol ve sağ girinti genişliği	Üst ve alt girinti yüksekliği
960 x 540	816 x 460	72	40
1280 x 720	1088 x 612	96	54
1920 x 1080	1632 x 918	144	81

Ancak, güvenli görüntüleme alanını her zaman dinamik olarak hesaplamak en iyi uygulamadır. Örneğin:

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

Sahne alanının ölçek modu

`Stage.scaleMode` öğesini `StageScaleMode.NO_SCALE` olarak ayarlayın ve sahne alanı resize olaylarını dinleyin.

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

Bu ayar, sahne alanı koordinatları ile piksel koordinatlarını aynı duruma getirir. `FULL_SCREEN_INTERACTIVE` görüntüleme durumu ve `TOP_LEFT` sahne alanı hizalaması ile birlikte, bu ayar güvenli görüntüleme alanını etkin bir şekilde kullanmanıza izin verir.

Özellikle tam ekran uygulamalarda, bu ölçek modu Stage sınıfının `stageWidth` ve `stageHeight` özelliklerinin Capabilities sınıfının `screenResolutionX` ve `screenResolutionY` özelliklerine karşılık geldiği anlamına gelir.

Ayrıca, uygulama penceresinin boyutu değiştiğinde, sahne alanının içeriği tanımlı boyutunu korur. Çalışma zamanı otomatik mizanpaj veya ölçekleme gerçekleştirmez. Ayrıca, pencere boyut değiştirdiğinde çalışma zamanı Stage sınıfının `resize` olayını gönderir. Bu nedenle, uygulama başladığında ve uygulama penceresi yeniden boyutlandırıldığında uygulamanın içeriğinin nasıl ayarlanacağı konusunda tam kontrole sahipsinizdir.

Not: *`NO_SCALE` davranışı herhangi bir AIR uygulaması ile aynıdır. Ancak, AIR for TV uygulamalarında bu ayarı kullanmak güvenli görüntüleme alanını kullanmak için önemlidir.*

Sahne alanını hizalama

`Stage.align` öğesini `StageAlign.TOP_LEFT` olarak ayarlayın:

```
stage.align = StageAlign.TOP_LEFT;
```

Hizalama, `0,0` koordinatını ActionScript kullanarak içerik yerleştirme işlemi için uygun bir alan olan ekranın sol üst köşesine yerleştirir.

`NO_SCALE` ölçek modu ve `FULL_SCREEN_INTERACTIVE` görüntüleme durumu ile birlikte bu ayar güvenli görüntüleme alanını etkin bir şekilde kullanmanıza izin verir.

Sahne alanı görüntüleme durumu

Tam ekran bir AIR for TV uygulamasında `Stage.displayState` öğesini`StageDisplayState.FULL_SCREEN_INTERACTIVE` olarak ayarlayın:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Bu değer AIR uygulamasını kullanıcı girdisine izin vererek sahne alanını tüm ekrana genişletecek şekilde ayarlar.

Adobe `FULL_SCREEN_INTERACTIVE` ayarını kullanmanızı önerir. `NO_SCALE` ölçek modu ve `TOP_LEFT` sahne hizalaması ile birlikte bu ayar güvenli görüntüleme alanını etkin bir şekilde kullanmanıza izin verir.

Bu nedenle, tam ekran uygulamalar için ana belge sınıfındaki `ADDED_TO_STAGE` olayı için bir işleyicide aşağıdakileri gerçekleştirin:

```
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;
}
```

Daha sonra `RESIZE` olayı işleyicisinde:

- Ekran çözünürlüğü boyutlarını sahne alanı genişliği ve yüksekliği ile karşılaştırın. Bunlar aynıysa, `RESIZE` olayı sahne alanı görüntüleme durumu `FULL_SCREEN_INTERACTIVE` olarak değiştiği için ortaya çıkmıştır.
- Güvenli görüntüleme alanı ve karşılık gelen girintilerin boyutlarını hesaplayın ve kaydedin.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {

        // Calculate and save safe viewing area dimensions.
    }
}
```

Sahne alanı boyutları `Capabilities.screenResolutionX` ve `screenResolutionY` ile eşit olduğunda, AIR for TV donanımın video ve grafikleriniz için mümkün olan en iyi aslına uygunluğu sunmasına neden olur.

Not: *Grafiklerin ve videonun TV ekranındaki aslına uygunluğu `Capabilities.screenResolutionX` ve `screenResolutionY` değerlerinden farklı olabilir. Bu AIR for TV'yi çalıştıran cihaza bağlıdır. Örneğin, AIR for TV'yi çalıştıran bir kod çözücü 1280 x 720 çözünürlüğe sahipken bağlı TV 1920 x 1080 çözünürlüğüne sahip olabilir. Ancak, AIR for TV donanımın en iyi aslına uygunluğu sunmasına neden olur. Bu nedenle, bu örnekte donanım 1920 x 1080 ekran çözünürlüğünü kullanarak 1080p bir video görüntüler.*

Birden fazla ekran boyutu için tasarlama

Birden fazla AIR for TV aygıtında çalışması ve düzgün görünmesi için aynı tam ekran AIR for TV uygulamasını geliştirebilirsiniz. Aşağıdakileri yapın:

1. `scaleMode`, `align` ve `displayState` sahne alanı özelliklerini önerilen değerlere ayarlayın: sırasıyla `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` ve `StageDisplayState.FULL_SCREEN_INTERACTIVE`.
2. `Capabilities.screenResolutionX` ve `Capabilities.screenResolutionY` öğelerine dayalı güvenli görüntüleme alanını ayarlayın.
3. İçeriğinizin boyutunu ve mizanpajını güvenli görüntüleme alanının genişliğine ve yüksekliğine göre ayarlayın.

   Özellikle mobil aygıt uygulamalarıyla karşılaştırıldığında içeriğinizin nesneleri büyük olsa da, dinamik mizanpaj, göreceli konumlandırma ve uyarlanabilir içerik gibi kavramlar aynıdır. Bu kavramları desteklemek üzere ActionScript'le ilgili daha fazla bilgi almak için bkz. Birden fazla ekran boyutu için mobil Flash içeriği geliştirme.

Sahne alanının kalitesi

Bir TV için AIR uygulamasının `Stage.quality` özelliği her zaman `StageQuality.High` şeklindedir. Bunu değiştiremezsiniz.

Bu özellik, tüm Stage nesnelerinin oluşturulma kalitesini belirtir.

Uzaktan kumanda girdisini işleme

Kullanıcılar genellikle AIR for TV uygulamalarınızla uzaktan kumanda ile etkileşime geçerler. Ancak, tuş girdisini bir masaüstü uygulamasındaki klavyeden gelen tuş girdisiyle aynı şekilde işleyin. Özellikle `KeyboardEvent.KEY_DOWN` olayını işleyin. Daha fazla bilgi için *ActionScript 3.0 Geliştirici Kılavuzu* içindeki Klavye girdisini yakalama bölümüne bakın.

Uzaktan kumanda üzerindeki tuşlar ActionScript sabitleri ile eşleşir. Örneğin, bir uzaktan kumandadaki yön tuş takımının tuşları şu şekilde eşleşir:

Uzaktan kumandanın yön tuş takımı tuşu	ActionScript 3.0 sabiti
Yukarı	`Keyboard.UP`
Aşağı	`Keyboard.DOWN`
Sol	`Keyboard.LEFT`
Sağ	`Keyboard.RIGHT`
Tamam veya Seç	`Keyboard.ENTER`

AIR 2.5 uzaktan kumanda girdisini desteklemek için diğer birçok Keyboard sabitini eklemiştir. Tam bir liste için *Adobe Flash Platform için ActionScript 3.0 Başvurusu* içindeki Keyboard sınıfı bölümüne bakın.

Uygulamanızın mümkün olduğu kadar çok aygıtta çalıştığından emin olmak için Adobe şunları önerir:

- Mümkünse yalnızca yön tuş takımı tuşlarını kullanın.

  Farklı uzaktan kumanda aygıtları farklı tuş takımlarına sahiptir. Ancak, bunlar tipik olarak her zaman yön tuş takımı tuşlarına sahiptir.

  Örneğin, bir Blu-ray oynatıcısının uzaktan kumandası genelde "üstteki kanal" ve "alttaki kanal" tuşlarına sahip değildir. Oynat, duraklat, durdur tuşları bile her uzaktan kumanda da olmayabilir.

- Uygulama yön tuş takımı tuşlarından daha fazlasına ihtiyaç duyuyorsa Menu (Menü) ve Info (Bilgi) tuşlarını kullanır.

  Menu (Menü) ve Info (Bilgi) tuşları uzaktan kumandalardaki ikinci en yaygın tuşlardır.

- Evrensel uzaktan kumandaların sık sık kullanıldığını unutmayın.

  Belirli bir aygıt için bir uygulama oluşturuyor olsanız bile, çok sayıda kullanıcının aygıtla birlikte gelen uzaktan kumandayı kullanmadığını unutmayın. Bunun yerine, evrensel uzaktan kumanda kullanmaktadırlar. Ayrıca, kullanıcılar her zaman evrensel uzaktan kumandalarını aygıtın uzaktan kumandasıyla eşleşecek şekilde programlamazlar. Bu nedenle, yalnızca en sık kullanılan tuşların kullanılması tavsiye edilir.

- Kullanıcının her zaman yön tuş takımı tuşlarından birini kullanarak belirli bir durumdan çıkabileceğinden emin olun.

  Bazı durumlarda uygulamanız uzaktan kumandalarda çok sık kullanılmayan tuşlardan birini kullanmak için iyi bir nedene sahip olabilir. Yön tuş takımı tuşlarından biriyle kaçış yolu sağlamak uygulamanızın tüm aygıtlarda düzgün bir şekilde çalışmasını sağlar.

- Hedef TV için AIR aygıtının işaretçi girdisi özelliğine sahip olduğundan emin değilseniz işaretçi girdisini gerekli kılmayın.

  Birçok masaüstü uygulaması fare girdisi beklese de, çoğu televizyon işaretçi girdisini desteklemez. Bu nedenle, televizyonlarda çalıştırmak için masaüstü uygulamalarını dönüştürüyorsanız, uygulamayı fare girdisi beklemeyecek şekilde değiştirdiğinizden emin olun. Bu değişiklikler olay işleme değişikliklerini ve kullanıcı talimatları değişikliklerini içerir. Örneğin, bir uygulamanın başlangıç ekranı görüntülendiğinde "Başlamak için tıklatın" yazan metni kullanmayın.

Odak yönetme

Bir kullanıcı arabirimi öğesi bir masaüstü uygulamasında odağa sahip olduğunda, klavye ve fare olayları gibi kullanıcı girdisi olaylarının hedefidir. Ayrıca, bir uygulama kullanıcı arabirimi öğesini odak ile vurgular. Bir AIR for TV uygulamasında odak yönetmek bir masaüstü uygulamasında odak yönetmekten farklıdır. Bunun nedeni şudur:

- Masaüstü uygulamaları genellikle odağı bir sonraki kullanıcı arabirimi öğesine değiştirmek için tab tuşunu kullanır. Tab tuşunu kullanmak AIR for TV uygulamaları için geçerli değildir. Uzaktan kumanda aygıtları genellikle bir tab tuşuna sahip değildir. Bu nedenle, masaüstündeki gibi bir DisplayObject öğesinin `tabEnabled` özelliği ile odak yönetme işlemi geçerli değildir.
- Masaüstü uygulamaları genellikle bir kullanıcı arabirimi öğesine odak vermek için kullanıcının fareyi kullanmasını beklerler.

Bu nedenle, uygulamanızda aşağıdakileri yapın:

- `KeyboardEvent.KEY_DOWN` gibi Keyboard olaylarını dinleyen Sahne Alanı'na bir olay dinleyicisi ekleyin.
- Son kullanıcıya hangi kullanıcı arabiriminin vurgulanacağını belirlemek için uygulama mantığı sağlayın. Uygulama başladığında bir kullanıcı arabirimi öğesi vurguladığınızdan emin olun.
- Uygulama mantığınıza göre Sahne Alanı'nın aldığı Keyboard olayını uygun kullanıcı arabirimi öğesi nesnesine gönderin.

  Ayrıca bir kullanıcı arabirimi öğesine odak atamak için `Stage.focus` veya `Stage.assignFocus()` öğesini de kullanabilirsiniz. Ardından keyboard olayları almasını sağlamak için o DisplayObject nesnesine bir olay dinleyicisi ekleyebilirsiniz.

Kullanıcı arabirimi tasarımı

Şunlarla ilgili önerileri dahil ederek AIR for TV uygulamasının kullanıcı arabiriminin televizyonlarda düzgün bir biçimde çalışmasını sağlayın:

- uygulamanın yanıt verme özelliği
- uygulamanın kullanılabilirliği
- kullanıcının kişiliği ve beklentileri

Yanıt verme

Bir AIR for TV uygulamasını mümkün olduğu kadar yanıt verir duruma getirmek için aşağıdaki ipuçlarını kullanın.

- Uygulamanın ilk SWF dosyasını mümkün olduğunca küçük yapın.

  İlk SWF dosyasında yalnızca uygulamayı başlatmak için gerekli olan kaynakları yükleyin. Örneğin, yalnızca uygulamanın başlangıç ekranı görüntüsü yükleyin.

  Bu öneri masaüstü AIR uygulamaları için geçerli olsa da, AIR for TV aygıtları için daha önemlidir. Örneğin, AIR for TV aygıtları masaüstü bilgisayarların sahip olduğu işleme gücüne sahip değildir. Ayrıca, bunlar uygulamayı flash belleğinde saklar. Bunlara erişim masaüstü bilgisayarlardaki sabit disklere olduğu kadar hızlı değildir.

- Uygulamanın en az saniye başına 20 kare hızında çalışmasını sağlayın.

  Bu amaca ulaşmak için grafiklerinizi tasarlayın. Grafiklerinizin karmaşıklığı saniye başına kare hızını etkileyebilir. Görüntü oluşturma performansını iyileştirme hakkında ipuçları için bkz: Adobe Flash Platform için Performansı En İyi Duruma Getirme

***Not:** AIR for TV aygıtlarındaki grafik donanımı genellikle ekranı 60 Hz veya 120 Hz (saniyede 60 veya 120 kez) hızıyla günceller. Örneğin, donanım 60 HZ veya 120 HZ ekrandaki görüntü için sahne alanını saniye başına 30 kare veya saniye başına 60 kare hızıyla tarar. Ancak, kullanıcının bu yüksek kare hızlarını tecrübe edip etmemesi uygulamanın grafiklerinin karmaşıklığına bağlıdır.*

- Ekranı kullanıcı girdisinin 100 - 200 milisaniyesi içinde güncelleyin.

  Güncelleme uzun sürerse kullanıcı sabırsızlanır ve bu genellikle birden fazla tuş basımına neden olur.

Kullanılabilirlik

AIR for TV uygulamalarının kullanıcıları "oturma odası" ortamındadırlar. TV'den 3 metre ilerde, odanın karşısında otururlar. Bazen oda karanlıktır. Girdi için genellikle uzaktan kumanda cihazı kullanırlar. Bazen birlikte, bazen sırayla uygulamayı birden fazla kişi kullanıyor olabilir.

Bu nedenle, kullanıcı arabiriminizi bir TV'de kullanılabilmek üzere tasarlamak için aşağıdakileri göz önünde bulundurun:

- Kullanıcı arabirimi öğelerini büyük yapın.

  Metin, düğmeler veya diğer kullanıcı arabirimi öğelerini tasarlarken, kullanıcının odanın karşısında oturduğunu unutmayın. Her şeyi, örneğin 3 metre ileriden kolaylıkla görülecek ve okunacak şekilde ayarlayın. Ekran büyük olduğu için ekranın içeriğini kalabalık tutmaya çalışmayın.

- İçeriğin odanın karşısından kolaylıkla görülmesini ve okunmasını sağlamak için iyi bir kontrast kullanın.
- Öğeyi parlak hale getirerek hangi kullanıcı arabirimi öğesinin kullanımda olduğunu belli edin.
- Hareketi yalnızca gerektiğinde kullanın. Örneğin, süreklilik için bir ekrandan diğerine kayarak geçme işlemi uygun olabilir. Ancak, hareket kullanıcının gezinmesine yardımcı olmuyorsa veya uygulamaya yerleşik değilse dikkat dağıtıcı olabilir.
- Her zaman kullanıcıya kullanıcı arabirimi aracılığıyla geri gitmek için belirgin bir yol sağlayın.

Uzaktan kumanda kullanımıyla ilgili daha fazla bilgi için bkz. "Uzaktan kumanda girdisini işleme" sayfa 129.

Kullanıcının kişiliği ve beklentileri

AIR for TV uygulamalarının kullanıcılarının genellikle eğlenceli ve rahat bir ortamda TV kalitesi eğlencesi aradıklarını göz önünde bulundurun. Bilgisayarlar veya teknoloji hakkında mutlaka bilgi sahibi olmak zorunda değildirler.

Bu nedenle, AIR for TV uygulamalarını aşağıdaki özelliklerle tasarlayın:

- Teknik terimler kullanmayın.
- Kalıcı iletişim kutularından kaçının.
- Çalışma ortamı veya teknik ortam için değil, oturma odası ortamı için uygun olan samimi, resmi olmayan talimatlar kullanın.
- TV izleyicilerinin beklediği yüksek üretim kalitesine sahip grafikler kullanın.
- Uzaktan kumanda cihazıyla kolaylıkla çalışan bir kullanıcı arabirimi oluşturun. Masaüstü veya mobil uygulamalar için daha uygun olan kullanıcı arabirimi veya tasarım öğeleri kullanmayın. Örneğin, masaüstü ve mobil cihazlardaki kullanıcı arabirimleri genellikle işaret etmeyi ve fare veya parmak ile düğmeleri tıklatmayı içerir.

Fontlar ve metin

AIR for TV uygulamanızda aygıt fontlarını veya gömülü fontları kullanabilirsiniz.

Aygıt fontları, bir aygıtta yüklü olan fontlardır. Tüm AIR for TV aygıtlarında aşağıdaki aygıt fontları vardır:

Font adı	Açıklama
`_sans`	`_sans` aygıt fontu sans-serif yazı biçimidir. Tüm AIR for TV aygıtlarında yüklü olan `_sans` aygıt fontu Myriad Pro'dur. Genellikle, görüş mesafesi nedeniyle sans-serif yazı biçimi TV'de serif yazı biçimlerinden daha iyi görünür.
`_serif`	`_serif` aygıt fontu serif yazı biçimidir. Tüm AIR for TV aygıtlarında yüklü olan `_serif` aygıt fontu Minion Pro'dur.
`_typewriter`	`_typewriter` aygıt fontu tek aralıklı bir fonttur. Tüm AIR for TV aygıtlarında yüklü olan `_typewriter` aygıt fontu Courier Std fontudur.

Tüm AIR for TV aygıtları ayrıca aşağıdaki Asya aygıt fontlarına da sahiptir:

Font adı	Dil	Yazı biçimi kategorisi	yerel ayar kodu
RyoGothicPlusN-Regular	Japonca	sans	ja
RyoTextPlusN-Regular	Japonca	serif	ja
AdobeGothicStd-Light	Korece	sans	ko
AdobeHeitiStd-Regular	Basitleştirilmiş Çince	sans	zh_CN
AdobeSongStd-Light	Basitleştirilmiş Çince	serif	zh_CN
AdobeMingStd-Light	Geleneksel Çince	serif	zh_TW ve zh_HK

Bu AIR for TV aygıt fontları:

- Adobe® Yazım Kitaplığı'ndandır
- Televizyonlarda iyi görünür
- Video ek bilgileri için tasarlanmıştır
- Bitmap fontları değil, font özetleridirler

Not: *Aygıt üreticileri genellikle aygıta diğer aygıt fontlarını da dahil ederler. Bu üretici tarafından sağlanan aygıt fontları, AIR for TV aygıt fontlarına ek olarak yüklenir.*

Adobe, aygıtta tüm aygıt fontlarını görüntüleyen FontMaster Deluxe adlı bir uygulama sağlar. Uygulama TV için Flash Platform'da bulunabilir.

Ayrıca, AIR for TV uygulamanıza font gömebilirsiniz. Gömülü fontlarla ilgili bilgi için *ActionScript 3.0 Geliştirici Kılavuzu* içindeki Gelişmiş metin oluşturma bölümüne bakın.

Adobe, TLF metin alanlarının kullanımıyla ilgili aşağıdakileri önerir:

- Uygulamanın çalıştığı yerel ayardan faydalanmak üzere Asya dili metinleri için TLF metin alanlarını kullanın. TLFTextField nesnesiyle ilişkili TextLayoutFormat nesnesinin `locale` özelliğini ayarlayın. Geçerli yerel ayarı belirlemek için *ActionScript 3.0 Geliştirici Kılavuzu* içindeki Bir yerel ayar seçme bölümüne bakın.
- Font AIR for TV aygıt fontlarından biri değilse TextLayoutFormat nesnesinde `fontFamily` özelliğinde font adını belirtin. AIR for TV, aygıtta mevcutsa fontu kullanır. İstediğiniz font aygıtta değilse, `locale` ayarına bağlı olarak, AIR for TV uygun AIR for TV aygıt fontunu yerine kullanır.
- AIR for TV'nin doğru AIR for TV aygıt fontunu seçmesini sağlamak için `locale` özelliğini ayarlayarak `fontFamily` özelliği için `_sans`, `_serif` veya `_typewriter` öğesini belirtin. Yerel ayarlara bağlı olarak, AIR for TV Asya aygıt fontları kümesinden veya Asya aygıt fontu olmayan kümeden seçim yapar. Bu ayarlar dört büyük Asya yerel ayarı ve İngilizce için otomatik olarak doğru fontu kullanmanızı sağlamak üzere kolay bir yol sağlar.

Not: *Asya dili metni için klasik metin alanlarını kullanırsanız, düzgün oluşturmayı sağlamak için bir AIR for TV aygıt fontunun adını belirtin. Hedef aygıtınızda farklı bir fontun yüklü olduğunu biliyorsanız, onu da belirtebilirsiniz.*

Uygulama performansı ile ilgili olarak aşağıdakileri göz önünde bulundurun:

- Klasik metin alanları TLF metin alanlarından daha hızlı performans sağlar.
- Bitmap fontları kullanan bir klasik metin alanı en hızlı performansı sağlar.

  Bitmap fontları, her karakter hakkında yalnızca özet veri sağlayan özet fontlarının aksine her karakter için bitmap sağlar. Aygıt fontları ve gömülü fontlar bitmap fontları olabilir.
- Aygıt fontu belirtirseniz, aygıt fontunun hedef aygıtınızda yüklü olduğundan emin olun. Aygıtınızda yüklü değilse, AIR for TV aygıtınızda yüklü olan başka bir fontu bulur ve kullanır. Ancak, bu davranış uygulamanın performansını yavaşlatır.
- Herhangi bir görüntüleme nesnesinde olduğu gibi TextField nesnesi genellikle değişmiyorsa, nesnenin `cacheAsBitmap` özelliğini `true` olarak ayarlayın. Bu ayar, soldurma, kayma ve alfa karıştırma gibi geçişler için performansı artırır. Ölçekleme ve çeviri için `cacheAsBitmapMatrix` öğesini kullanın. Daha fazla bilgi için bkz. “Grafik donanım hızlandırması” sayfa 123.

Dosya sistemi güvenliği

AIR for TV uygulamaları AIR uygulamalarıdır ve bu nedenle, aygıtın dosya sistemine erişebilir. Ancak, bir “oturma odası” aygıtında bir uygulamanın aygıt sistemi dosyalarına veya diğer uygulamaların dosyalarına erişemiyor olması çok önemlidir. TV ve ilişkili aygıtların kullanıcıları yalnızca TV izliyor olduklarından, aygıtta sorun oluşmasını beklemez ve bu sorunları hoş karşılamazlar.

Bu nedenle, AIR for TV uygulaması aygıtın dosya sisteminin sınırlı bir görünümüne sahiptir. ActionScript 3.0'ı kullanarak uygulamanız yalnızca belirli dizinlere (ve bunların alt dizinlerine) erişebilir. Ayrıca, ActionScript'te kullandığınız dizin adları aygıttaki gerçek dizin adları değildir. Bu ekstra katman AIR for TV uygulamalarını, uygulamaya ait olmayan yerel dosyalara kötü niyetle veya yanlışlıkla erişmekten korur.

Ayrıntılı bilgi için bkz. AIR for TV uygulamaları için dizin görünümü.

AIR uygulamasının sanal alanı

AIR for TV uygulamaları AIR uygulaması sanal alanı bölümünde açıklanan AIR uygulaması sanal alanında çalışır.

AIR for TV uygulamalarının tek farkı “Dosya sistemi güvenliği” sayfa 134 bölümünde açıklandığı gibi dosya sistemine sınırlı erişimleri olmasıdır.

Uygulama yaşam döngüsü

Bir masaüstü ortamının aksine, son kullanıcı AIR for TV uygulamanızın çalıştığı pencereyi kapatamaz. Bu nedenle, uygulamadan çıkmak için bir kullanıcı arabirimi mekanizması sağlayın.

Genellikle, bir aygıt son kullanıcının uzaktan kumandadaki çıkış tuşuyla bir uygulamadan koşulsuz olarak çıkmasına izin verir. Ancak, AIR for TV uygulamaya `flash.events.Event.EXITING` olayını göndermez. Bu nedenle, uygulamanın bir sonraki başlangıcında kendini uygun bir duruma geri yükleyebilmesini sağlamak için uygulama durumunu sık sık kaydedin.

HTTP tanımlama bilgileri

AIR for TV, HTTP kalıcı tanımlama bilgilerini ve oturum tanımlama bilgilerini destekler. AIR for TV, her AIR uygulamasının tanımlama bilgilerini uygulamaya özgü bir dizinde depolar.

`/app-storage/<app id>/Local Store`

Tanımlama bilgisi dosyası `cookies` olarak adlandırılır.

Not: *Masaüstü aygıtlar gibi diğer aygıtlarda bulunan AIR uygulaması, tanımlama bilgilerini her uygulama için ayrı ayrı depolamaz. Uygulamaya özgü tanımlama bilgisi depolaması uygulamayı ve AIR for TV uygulamasının sistem güvenlik modelini destekler.*

`URLRequest.manageCookies` ActionScript özelliğini şu şekilde kullanın:

- `manageCookies` öğesini `true` olarak ayarlayın. Bu değer varsayılan değerdir. Bu AIR for TV'nin HTTP isteklerine otomatik olarak tanımlama bilgisi eklediği ve HTTP yanıtındaki tanımlama bilgilerini hatırladığı anlamına gelir.

  Not: *`ManageCookies`true olsa bile, uygulama `URLRequest.requestHeaders` öğesini kullanarak bir HTTP isteğine manuel olarak tanımlama bilgisi ekleyebilir. Bu tanımlama bilgisi AIR for TV'nin yönettiği tanımlama bilgisiyle aynı ada sahipse, istek aynı adlı iki tanımlama bilgisini içerir. İki tanımlama bilgisinin değerleri farklı olabilir.*

- `manageCookies` öğesini `false` olarak ayarlayın. Bu değer, uygulamanın HTTP istekleri için tanımlama bilgisi gönderme ve HTTP yanıtındaki tanımlama bilgilerini hatırlama sorumluluğuna sahip olduğu anlamına gelir.

Daha fazla bilgi için bkz. URLRequest.

AIR for TV uygulaması oluşturma iş akışı

Aşağıdaki Adobe Flash Platform geliştirme araçlarıyla AIR for TV uygulamaları geliştirebilirsiniz:

- Adobe Flash Professional

  Adobe Flash Professional CS5.5, AIR for TV uygulamalarını destekleyen ilk AIR sürümü olan AIR 2.5 for TV'yi destekler.

- Adobe Flash® Builder®

  Flash Builder 4.5, AIR 2.5 for TV uygulamasını destekler.

- AIR SDK

  AIR 2.5'ten itibaren, AIR SDK ile sağlanan komut satırı araçlarını kullanarak uygulamalarınızı geliştirebilirsiniz. AIR SDK'yi indirmek için bkz. http://www.adobe.com/products/air/sdk/.

Flash Professional'ı Kullanma

AIR for TV uygulamalarını geliştirmek, test etmek ve yayınlamak için Flash Professional aracının kullanılması, aracın AIR masaüstü uygulamaları için kullanılmasıyla aynıdır.

Ancak, ActionScript 3.0 kodunu yazarken yalnızca `tv` ve `extendedTV` AIR profillerinin desteklediği sınıfları ve yöntemleri kullanın. Ayrıntılar için bkz. “Aygıt profilleri” sayfa 242.

Proje ayarları

Bir AIR for TV uygulaması için projenizi ayarlamak üzere aşağıdakileri yapın:

- Yayınlama Ayarları iletişim kutusunun Flash sekmesinde Oynatıcı değerini en az AIR 2.5 olarak ayarlayın.
- Adobe AIR Ayarları iletişim kutusunun Genel sekmesinde (Uygulama ve Yükleyici Ayarları), profili `TV` veya `extended` TV olarak ayarlayın.

Hata ayıklama

Flash Professional'da AIR Hata Ayıklama Başlatıcısı'nı kullanarak uygulamanızı çalıştırabilirsiniz. Aşağıdakileri yapın:

- Uygulamayı hata ayıklama modunda çalıştırmak için şunu seçin:

  Hata Ayıkla > Film Hatalarını Ayıkla > AIR Hata Ayıklama Başlatıcısı'nda (Masaüstü)

  Bu seçimi yaptıktan sonra, sonraki hata ayıklamaları için şunu seçebilirsiniz:

  Hata Ayıkla > Film Hatalarını Ayıkla > Hata Ayıkla

- Uygulamayı hata ayıklama modu özellikleri olmadan çalıştırmak için şunu seçin:

  Kontrol Et > Filmi Test Et > AIR Hata Ayıklama Başlatıcısı'nda (Masaüstü)

  Bu seçimi yaptıktan sonra bir sonrakiler için Kontrol Et > Filmi Test Et > Test Et seçeneğini belirleyebilirsiniz.

AIR profilini TV veya genişletilmiş TV olarak ayarladığınızdan, AIR Hata Ayıklama Başlatıcısı Uzaktan Kumanda Düğmeleri adlı bir menü sağlar. Uzaktan kumanda aygıtında tuşlara basma işleminin benzetimini yapmak için bu menüyü kullanabilirsiniz.

Daha fazla bilgi için "Flash Professional ile uzaktan hata ayıklama" sayfa 144.

Yerel uzantıları kullanma

Uygulamanız yerel bir uzantı kullanıyorsa, "Yerel bir uzantı kullanmaya yönelik görev listesi" sayfa 149 bölümündeki talimatları uygulayın.

Ancak, bir uygulama yerel uzantıları kullandığında:

- Flash Professional kullanarak uygulamayı yayınlayamazsınız. Uygulamayı yayınlamak için ADT kullanın. Bkz. "ADT ile paketleme" sayfa 141.
- Flash Professional kullanarak uygulamayı çalıştıramaz veya bu uygulamada hata ayıklayamazsınız. Geliştirme makinesinde uygulamada hata ayıklamak için ADL kullanın. Bkz. "ADL kullanarak aygıt benzetimi" sayfa 142

Flash Builder'ı Kullanma

Flash Builder 4.5'ten itibaren, Flash Builder, AIR for TV gelişimini destekler. AIR for TV uygulamalarını geliştirmek, test etmek ve yayınlamak için Flash Builder'ı kullanma, AIR masaüstü uygulamaları için aracı kullanma işlemine benzerdir.

Uygulamayı ayarlama

Uygulamanızın şunları gerçekleştirdiğinden emin olun:

- Bir MXML dosyası kullanıyorsanız, MXML dosyasında container sınıfı olarak `Application` öğesi kullandığından:

```
<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>.
```

  Önemli: *AIR for TV uygulamaları* `WindowedApplication` *öğesini desteklemez.*

  Not: *Hiç bir MXML dosyası kullanmanız gerekmez. Bunun yerine ActionScript 3.0 projesi oluşturabilirsiniz.*

- Yalnızca `tv` ve `extendedTV` AIR profillerinin desteklediği ActionScript 3.0 sınıflarını kullanın. Ayrıntılar için bkz. “Aygıt profilleri” sayfa 242.

Ayrıca, uygulamanızın XML dosyasında şunlardan emin olun:

- `application` öğesinin `xmlns` niteliğinin AIR 2.5 olarak ayarlı olduğundan:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

- `supportedProfiles` öğesinin `tv` veya `extendedTV` içerdiğinden:

```
<supportedProfiles>tv</supportedProfiles>
```

Uygulamada hata ayıklama

Uygulamanızı Flash Builder içindeki AIR Hata Ayıklama Başlatıcısı'nı kullanarak çalıştırabilirsiniz. Aşağıdakileri yapın:

1. Run (Çalıştır) > Debug Configurations (Hata Ayıklama Yapılandırmaları) öğesini seçin.
2. Profile (Profil) alanının Desktop (Masaüstü) olarak ayarlandığından emin olun.
3. Hata ayıklama modunda çalıştırmak için Run (Çalıştır) > Debug (Hata Ayıkla) öğesini veya hata ayıklama modu özellikleri olmadan çalıştırmak için Run (Çalıştır) > Run (Çalıştır) öğesini seçin.

`supportedProfiles` öğesini TV veya genişletilmiş TV olarak ayarladığınızdan, AIR Hata Ayıklama Başlatıcısı Uzaktan Kumanda Düğmeleri adlı bir menü sağlar. Uzaktan kumanda aygıtında tuşlara basma işleminin benzetimini yapmak için bu menüyü kullanabilirsiniz.

Daha fazla bilgi için bkz. “Flash Builder ile uzaktan hata ayıklama” sayfa 144.

Yerel uzantıları kullanma

Uygulamanız yerel bir uzantı kullanıyorsa, “Yerel bir uzantı kullanmaya yönelik görev listesi” sayfa 149 bölümündeki talimatları uygulayın.

Ancak, bir uygulama yerel uzantıları kullandığında:

- Uygulamayı Flash Builder'ı kullanarak yayınlayamazsınız. Uygulamayı yayınlamak için ADT kullanın. Bkz. “ADT ile paketleme” sayfa 141.
- Flash Builder kullanarak uygulamayı çalıştıramaz veya bu uygulamada hata ayıklayamazsınız. Geliştirme makinesinde uygulamada hata ayıklamak için ADL kullanın. Bkz. “ADL kullanarak aygıt benzetimi” sayfa 142

AIR for TV uygulaması tanımlayıcısı özellikleri

Diğer AIR uygulamalarında olduğu gibi, temel uygulama özelliklerini uygulama tanımlayıcı dosyasında ayarlarsınız. TV profili uygulamaları pencere boyutu ve saydamlık gibi bazı masaüstüne özgü özellikleri yok sayar. `extendedTV` profilindeki aygıtları hedefleyen uygulamalar yerel uzantıları kullanabilir. Bu uygulamalar bir `extensions` öğesinde kullanılan yerel uzantıları tanımlar.

Ortak ayarlar

Çeşitli uygulama tanımlayıcısı ayarları tüm TV profili uygulamaları için önemlidir.

Gerekli AIR çalışma zamanı sürümü

Uygulama tanımlayıcı dosyasının ad alanını kullanarak uygulamanızın gerektirdiği AIR çalışma zamanı sürümünü belirtin.

`application` öğesinde atanan ad alanı büyük bir bölümünde uygulamanızın hangi özellikleri kullandığını belirler. Örneğin, AIR 2.5 ad alanını kullanan bir uygulama olduğunu, ancak kullanıcının daha sonraki bir sürüme sahip olduğunu düşünün. Bu durumda, uygulama davranış sonraki AIR sürümünde farklı olsa bile AIR 2.5 davranışını görür. Yalnızca ad alanını değiştirdiğinizde ve bir güncelleme yayınladığınızda uygulamanız yeni davranışa ve özelliklere erişebilir. Güvenlik düzeltmeleri bu kuraldaki önemli istisnalardır.

Ad alanını kök `application` öğesinin `xmlns` niteliğini kullanarak belirtin:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5, TV uygulamalarını destekleyen ilk AIR sürümüdür.

Uygulama kimliği

Yayınladığınız her uygulama için birkaç ayar benzersiz olmalıdır. Bu ayarlar `id`, `name` ve `filename` öğelerini içerir.

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

Uygulama sürümü

Uygulama sürümünü `versionNumber` öğesinde belirtin. `versionNumber` için değer belirtirken noktalarla ayrılmış, üç numaraya kadar sayı içerebilen şunun gibi bir sıralama kullanabilirsiniz: “0.1.2”. Sürüm numarasının her bir parçası en fazla üç basamak içerebilir. (Başka bir deyişle, "999.999.999" izin verilen en büyük sürüm sayısıdır.) Sayıda tüm parçaları içermeniz gerekmez; “1” ve “1.0” örnekleri de geçerli sayılardır.

Ayrıca `versionLabel` öğesini kullanan sürüm için de bir etiket belirtebilirsiniz. Sürüm etiketi eklediğinizde sürüm numarasının yerine görüntülenir.

```
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

Ana uygulama SWF'si

`initialWindow` öğesinin `versionLabel` alt öğesinde ana uygulama SWF dosyasını belirtin. TV profilindeki aygıtları hedef aldığınızda, bir SWF dosyası kullanmanız gerekir (HTML tabanlı uygulamalar desteklenmez).

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

Dosyayı AIR paketine dahil etmeniz gerekir (ADT veya IDE'nizi kullanarak) Uygulama tanımlayıcısında ada başvurmak dosyanın otomatik olarak pakete dahil olmasını sağlamaz.

Ana ekran özellikleri

Ana uygulama ekranının ilk görünümünü ve davranışını `initialWindow` öğesinin çeşitli alt öğeleri denetler. Bu özelliklerin çoğu TV profillerindeki aygıtlarda yok sayılsa da `fullScreen` öğesini kullanabilirsiniz:

- **`fullScreen`** — Uygulamanın tam aygıt ekranına mı geçmesi gerektiğini yoksa ekranı normal işletim sistemi kromuyla paylaşması mı gerektiğini belirtir.

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

Visible öğesi

`visible` öğesi `initialWindow` öğesinin bir alt öğesidir. Uygulamanızın içeriği TV için AIR aygıtlarında her zaman görünür olduğundan, TV için AIR `visible` öğesini yoksayar.

Ancak uygulamanız aynı zamanda masaüstü aygıtlarını da hedefliyorsa `visible` öğesini `true` olarak ayarlayın.

Masaüstü aygıtlarda bu öğenin değeri varsayılan olarak `false` şeklindedir. Bu nedenle, `visible` öğesini dahil etmezseniz uygulamanın içeriği masaüstü aygıtlarda görünür olmaz. Masaüstü aygıtlarında içeriği görünür kılmak için ActionScript sınıfı olan NativeWindow sınıfını kullanabilseniz de, TV aygıt profilleri NativeWindow sınıfını desteklemez. TV için AIR aygıtında çalışan bir uygulamada NativeWindow sınıfını kullanmayı denerseniz uygulama yüklenemez. NativeWindow sınıfının bir yöntemini çağırmış olup olmamanız durumu değiştirmez; bu sınıfı kullanan bir uygulama TV için AIR aygıtında yüklenmez.

Desteklenen profiller

Uygulamanız yalnızca televizyon aygıtında anlamlıysa, diğer bilgi işlem aygıtlarında yüklenmesini engelleyebilirsiniz. Diğer profilleri `supportedProfiles` öğesindeki desteklenenler listesinin dışında bırakın:

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Bir uygulama yerel uzantı kullanıyorsa, desteklenen profil listesine yalnızca `extendedTV` profilini dahil edin:

```
<supportedProfiles>extendedTV</supportedProfiles>
```

`supportedProfiles` öğesini yok sayarsanız uygulamanın tüm profilleri desteklediği varsayılır.

`supportedProfiles` listesine *yalnızca* `tv` profilini dahil etmeyin. Bazı TV aygıtları, TV için AIR'yi her zaman `extendedTV` profiline karşılık gelen bir modda çalıştırır. Bu davranış sayesinde TV için AIR yerel uzantılar kullanan uygulamaları çalıştırabilir. `supportedProfiles` öğeniz yalnızca `tv` öğesini belirtiyorsa, bu içeriğinizin `extendedTV` profiline yönelik TV için AIR modu ile uyumlu olmadığını gösterir. Bu nedenle, bazı TV aygıtları yalnızca `tv` profilini belirten bir uygulamayı yüklemez.

`tv` ve `extendedTV` profillerinde desteklenen ActionScript sınıflarının bir listesi için bkz. "Farklı profillerin yetenekleri" sayfa 243.

Gerekli yerel uzantılar

`extendedTV` profilini destekleyen uygulamalar yerel uzantıları kullanabilir.

`extensions` ve `extensionID` öğelerini kullanarak uygulama tanımlayıcısında AIR uygulamasının kullandığı tüm yerel uzantıları bildirin. Aşağıdaki örnek, gerekli iki yerel uzantıyı belirtmeye yönelik sözdizimini gösterir:

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

Bir uzantı listelenmemişse uygulama bunu kullanamaz.

`extensionID` öğesi uzantı tanımlayıcı dosyasındaki `id` öğesiyle aynı değere sahiptir. Uzantı tanımlayıcı dosyası extension.xml adlı bir XML dosyasıdır. Aygıt üreticisinden aldığınız ANE dosyasında paketlidir.

Bir uzantıyı `extensions` öğesinde listelerseniz, ancak uzantı TV için AIR aygıtında yüklü değilse uygulama çalışmaz. TV için AIR uygulamanızla birlikte paketlediğiniz ANE dosyası uzantının bir saplama sürümüne sahipse kuralda istisnai bir durum oluşur. Bu durumda uygulama çalışabilir ve uzantının saplama sürümünü kullanır. Saplama sürümü ActionScript koduna sahiptir, ancak yerel koda sahip değildir.

Uygulama simgeleri

Televizyon aygıtlarındaki uygulama simgesi gereksinimleri aygıta bağlıdır. Örneğin, şunları aygıt üreticisi belirtir:

- Gerekli simgeleri ve simge boyutlarını.
- Gerekli dosya türlerini ve adlandırma kurallarını.
- Uygulamanız için simgelerin nasıl sağlanacağını. Örneğin, uygulamanızla birlikte simgelerin paketlenip paketlenmeyeceğini belirtir.
- Simgelerin uygulama tanımlayıcı dosyasında `icon` öğesi içinde belirtilip belirtilmeyeceği.
- Uygulama simgeleri sağlamazsa sergilenecek davranışı.

Ayrıntılar için aygıt üreticisine başvurun.

Yoksayılan ayarlar

Televizyon aygıtlarındaki uygulamalar mobil, yerel pencere veya masaüstü işletim sistemi özellikleri için geçerli olan uygulama ayarlarını yok sayar. Yoksayılan ayarlar şunlardır:

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

Bir AIR for TV uygulaması paketleme

ADT ile paketleme

Bir AIR for TV uygulamasını paketlemek için AIR ADT komut satırı aracını kullanabilirsiniz. AIR SDK sürümü 2.5'ten itibaren, ADT, TV aygıtları için paketlemeyi destekler. Paketlemeden önce tüm ActionScript ve MXML kodunuzu derleyin. Ayrıca, bir kod imzalama sertifikanızın olması gerekir. ADT -certificate komutunu kullanarak bir sertifika oluşturabilirsiniz.

ADT komutları ve seçenekleriyle ilgili ayrıntılı bir başvuru için bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163.

AIR paketi oluşturma

AIR paketi oluşturmak için ADT package komutunu kullanın:

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

Örnek şunları varsayar:

- ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu. (Bkz. "Path ortam değişkenleri" sayfa 300.)
- Sertifika codesign.p12 öğesinin ADT komutunu çalıştırdığınız üst dizinde olduğunu.

Komutu uygulama dosyalarını içeren dizinden çalıştırın. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf ve bir simgeler dizinidir.

Komutu gösterildiği şekilde çalıştırdığınızda ADT sizden anahtar deposu şifresini ister. Tüm kabuk programları yazdığınız şifre karakterlerini görüntülemez; yazmayı bitirdiğinizde Enter tuşuna basın. Alternatif olarak, şifreyi ADT komutuna dahil etmek için `storepass` parametresini kullanabilirsiniz.

Bir AIRN paketi oluşturma

AIR for TV uygulamanız yerel bir uzantı kullanıyorsa, bir AIR paketi yerine bir AIRN paketi oluşturun. Bir AIRN paketi oluşturmak için hedef türünü `airn` olarak ayarlayarak ADT package komutunu kullanın.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml
myApp.swf icons -extdir C:\extensions
```

Örnek şunları varsayar:

- ADT aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu. (Bkz. "Path ortam değişkenleri" sayfa 300.)
- Sertifika codesign.p12 öğesinin ADT komutunu çalıştırdığınız üst dizinde olduğunu.
- `-extdir` parametresi, uygulamanın kullandığı ANE dosyalarını içeren bir dizini adlandırır.

  Bu ANE dosyaları uzantının yalnızca ActionScript saplamasını veya benzetici sürümünü içerir. Yerel kodu içeren uzantısının sürümü AIR for TV aygıtına yüklenir.

Komutu uygulama dosyalarını içeren dizinden çalıştırın. Örnekteki uygulama dosyaları myApp-app.xml (uygulama tanımlayıcı dosyası), myApp.swf ve bir simgeler dizinidir.

Komutu gösterildiği şekilde çalıştırdığınızda ADT sizden anahtar deposu şifresini ister. Tüm kabuk programları yazdığınız şifre karakterlerini görüntülemez; yazmayı bitirdiğinizde Enter tuşuna basın. Alternatif olarak, şifreyi komuta dahil etmek için `storepass` parametresini kullanabilirsiniz.

Yerel uzantıları kullanan AIR for TV uygulaması için bir AIRI dosyası da oluşturabilirsiniz. Bir AIRI dosyası imzalanmamış olması dışında AIRN dosyası gibidir. Örneğin:

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Uygulamayı imzalamaya hazır olduğunuzda, AIRI dosyasından AIRN dosyası oluşturabilirsiniz:

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Daha fazla bilgi için bkz. Adobe AIR için Yerel Uzantılar Geliştirme.

Flash Builder veya Flash Professional ile paketleme

Flash Professional ve Flash Builder ADT'yi kendiniz çalıştırmanıza gerek kalmadan AIR paketlerini yayınlamanıza veya dışa aktarmanıza izin verir. Bir AIR uygulaması için AIR paketi oluşturma yordamı bu programların belgelerinde ele alınmıştır.

Ancak şu anda yalnızca ADT, yerel uzantıları kullanan AIR for TV uygulamaları için uygulama paketleri niteliğindeki AIRN paketlerini oluşturabilir.

Daha fazla bilgi için, şu konulara bakın:

- AIR uygulamalarını Flash Builder ile paketleme
- Flash Professional'ı kullanarak Adobe AIR için Yayınlama

AIR for TV uygulamalarında hata ayıklama

ADL kullanarak aygıt benzetimi

Birçok uygulama özelliğini test etmenin ve bunlarda hata ayıklamanın en hızlı ve kolay yolu, Adobe Hata Ayıklama Başlatıcısı (ADL) yardımcı programını kullanarak uygulamanızı geliştirme bilgisayarında çalıştırmaktır.

ADL kullanılacak profili seçmek için uygulama tanımlayıcısında `supportedProfiles` öğesini kullanır. Özel olarak belirtmek gerekirse:

- Birden fazla profil listelenmişse ADL listedeki ilk profili kullanır.
- `supportedProfiles` listesindeki diğer profillerden birini seçmek için ADL'nin `-profile` parametresini kullanabilirsiniz.
- Uygulama tanımlayıcısına bir `supportedProfiles` öğesi dahil etmezseniz, `-profile` argümanı için herhangi bir profil belirtilebilir.

Örneğin, `tv` profilini benzetmek üzere bir uygulama başlatmak için aşağıdaki komutu kullanın:

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

ADL ile masaüstünde `tv` veya `extendedTV` profilini benzetirken, uygulama hedef aygıtla daha yakından eşleşen bir ortamda çalışır. Örneğin:

- `-profile` argümanında profilin bir parçası olmayan ActionScript API'leri kullanılamaz.
- ADL, menü komutları aracılığıyla uzaktan kumanda gibi aygıt girdi kumandalarının girdilerine izin verir.
- `-profile` argümanında `tv` veya `extendedTV` öğesini belirtmek ADL'nin masaüstünde StageVideo sınıfını benzetmesine izin verir.
- `-profile` argümanında `extendedTV` öğesinin belirtilmesi, uygulamanın uygulama AIRN dosyasıyla birlikte paketlenen yerel uzantı saplamalarını veya benzeticilerini kullanmasına olanak verir.

Ancak, ADL uygulamayı masaüstünde çalıştırdığından AIR for TV uygulamalarını test etme işleminin sınırlamaları vardır:

- Aygıtta uygulama performansını yansıtmaz. Performans testlerini hedef aygıtta gerçekleştirin.
- StageVideo sınıfının sınırlamalarını benzetmez. Genellikle AIR for TV aygıtlarını hedef alırken bir video oynatmak için Video sınıfını değil StageVideo sınıfını kullanırsınız. StageVideo sınıfı, aygıtın donanımının performans avantajlarından yararlanır, ancak görüntü sınırlamaları vardır. ADL videoyu masaüstüne bu sınırlamalar olmadan oynatır. Bu nedenle, video oynatma işlemini hedef aygıtta test edin.
- Bu öğe yerel bir uzantının yerel kodunun benzetimini yapamaz. Ancak ADL `-profile` argümanındaki yerel uzantıları destekleyen `extendedTV` profilini belirtebilirsiniz. ADL, ANE paketinde bulunan uzantısının yalnızca ActionScript saplaması veya benzetici sürümüyle test yapmanıza izin verir. Ancak, genellikle aygıtta yüklü olan karşılık gelen uzantısı yerel kodu da içerir. Kendi yerel koduyla uzantısını kullanarak test etmek için uygulamayı hedef aygıtta çalıştırın.

Daha fazla bilgi için bkz. "AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157.

Yerel Uzantıları kullanma

Uygulamanız yerel uzantıları kullanıyorsa, ADL komutu aşağıdaki örnek gibi görünür:

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

Örnek şunları varsayar:

- ADL aracının yolunun komut satırı kabuğunuzun yol tanımında olduğunu. (Bkz. "Path ortam değişkenleri" sayfa 300.)
- Geçerli dizin uygulama dosyalarını içerir. Bu dosyalar SWF dosyalarını ve bu örnekte myApp-app.xml olan uygulama tanımlayıcı dosyasını içerir.
- `-extdir` parametresi uygulamanın kullandığı her yerel uzantı için bir dizin içeren bir dizini adlandırır. Bu dizinlerin her biri yerel bir uzantının *paketlenmemiş* ANE dosyasını içerir. Örneğin:

```
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
```

Bu paketlenmemiş ANE dosyaları uzantının yalnızca ActionScript saplamasını veya benzetici sürümünü içerir. Yerel kodu içeren uzantısının sürümü AIR for TV aygıtına yüklenir.

Daha fazla bilgi için bkz. Adobe AIR için Yerel Uzantılar Geliştirme.

Kumanda girdisi
ADL, bir TV aygıtındaki uzaktan kumanda düğmelerinin benzetimini yapar. ADL, TV profillerinden biri kullanılarak başlatıldığında görüntülenen menüyü kullanarak bu düğme girdilerini benzetilen aygıta gönderebilirsiniz.

Ekran boyutu
ADL `-screensize` parametresini ayarlayarak uygulamanızı farklı boyuttaki ekranlarda test edebilirsiniz. Normal ve ekranı kaplayan ekranların genişlik ve yüksekliklerini temsil eden dört değeri içeren bir dize belirtebilirsiniz.

Dikey mizanpaj için piksel boyutlarını her zaman belirtin. Bu, genişlik değerinin yükseklik değerinden daha küçük olarak belirtilmesi anlamına gelir. Örneğin:

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

İzleme ifadeleri

TV uygulamanızı masaüstünde çalıştırdığınızda, izleme çıktısı hata ayıklayıcısına veya ADL'yi başlatmak için kullanılan terminal penceresine yazdırılır.

Flash Professional ile uzaktan hata ayıklama

Hedef aygıtta çalışırken uzaktan AIR for TV uygulamanızda hata ayıklamak için Flash Professional 'i kullanabilirsiniz. Ancak, uzaktan hata ayıklama kurma adımları aygıta bağlıdır. Örneğin, Adobe® AIR® for TV MAX 2010 Donanım Geliştirme Kiti bu aygıt için ayrıntılı adımların belgelerini içerir.

Ancak hedef aygıttan bağımsız olarak uzaktan hata ayıklamaya hazırlanmak için aşağıdaki adımları uygulayın:

1. Yayınlama Ayarları iletişim kutusunun Flash sekmesinde Hata Ayıklamaya İzin Ver'i seçin.

   Bu seçenek, Flash Professional'ın FLA dosyanızdan oluşturduğu tüm SWF dosyalarına hata ayıklama bilgilerini dahil etmesine neden olur.

2. Adobe AIR Ayarları iletişim kutusunun (Uygulama ve Yükleyici Ayarları) İmza sekmesinde AIR Intermediate (AIRI) dosyası hazırlama seçeneğini belirleyin.

   Uygulamanızı geliştirmeye devam ederken dijital imza gerektirmeyen AIRI dosyasını kullanmanız yeterlidir.

3. AIRI dosyasını oluşturarak uygulamanızı yayınlayın.

Son adımlar uygulamayı hedef aygıta yüklemek ve burada çalıştırmaktır. Ancak, bu adımlar aygıta bağlıdır.

Flash Builder ile uzaktan hata ayıklama

Hedef aygıtta çalışırken uzaktan AIR for TV uygulamanızda hata ayıklamak için Flash Builder'ı da kullanabilirsiniz. Ancak, uzaktan hata ayıklama gerçekleştirme adımları aygıta bağlıdır.

Ancak hedef aygıttan bağımsız olarak uzaktan hata ayıklamaya hazırlanmak için aşağıdaki adımları uygulayın:

1. Project (Proje) > Export Release Build (Sürüm Yapısını Dışa Aktar) öğesini seçin. AIR Intermediate (AIRI) dosyası hazırlama seçeneğini belirtin.

   Uygulamanızı geliştirmeye devam ederken dijital imza gerektirmeyen AIRI dosyasını kullanmanız yeterlidir.

2. AIRI dosyasını oluşturarak uygulamanızı yayınlayın.
3. Uygulamanın AIRI paketini hata ayıklama bilgilerini içeren SWF dosyalarını içerecek şekilde değiştirin.

Hata ayıklama bilgilerini içeren SWF dosyaları, bin-debug adlı bir dizinde uygulamanın Flash Builder proje dizininde bulunur. AIRI paketindeki SWF dosyalarını bin-debug dizinindeki SWF dosyalarıyla değiştirin.

Bir Windows geliştirme makinesinde bu değişikliği aşağıdakileri yaparak gerçekleştirebilirsiniz:

1. .airi yerine .zip dosya adı uzantısına sahip olmasını sağlamak için AIR paketi dosyasını yeniden adlandırın.
2. ZIP dosyası içeriklerini çıkarın.
3. Çıkarılan dizin yapısındaki SWF dosyalarını bin-debug dizinindekilerle değiştirin.
4. Çıkarılan dizindeki dosyaları yeniden sıkıştırın.
5. Sıkıştırılmış dosyayı bir kez daha .airi dosya adı uzantısına sahip olacak şekilde değiştirin.

Mac geliştirme makinesi kullanıyorsanız, bu değiştirme adımları aygıta bağlıdır. Ancak, bunlar genellikle şunları içerir:

1. AIRI paketini hedef aygıta yükleyin.
2. Hedef aygıtta uygulamanın yükleme dizinindeki SWF dosyalarını bin-debug dizinindeki SWF dosyalarıyla değiştirin.

   Örneğin, Adobe AIR for TV MAX 2010 Donanım Geliştirme Kiti ile gelen aygıtı düşünün. AIRI paketini kitin belgelerinde açıklandığı gibi yükleyin. Daha sonra hedef aygıta erişmek için Mac geliştirme makinenizin komut satırında telnet'i kullanın. /opt/adobe/stagecraft/apps/<*uygulama adı*>/ yolunda bulunan uygulama yükleme dizinindeki SWF dosyalarını bin-debug dizinindeki SWF dosyalarıyla değiştirin.

Aşağıdaki adımlar Flash Builder ile uzaktan hata ayıklamak ve Adobe AIR for TV MAX 2010 Donanım Geliştirme Kiti ile birlikte gelen aygıt içindir.

1. Flash Builder'ın çalıştığı geliştirme bilgisayarınızda, MAX 2010 Donanım Geliştirme Kiti ile birlikte gelen AIR for TV Aygıt Bağlayıcısı'nı çalıştırın. Bu geliştirme bilgisayarınızın IP adresini gösterir.
2. Donanım kiti aygıtında, geliştirme kitiyle birlikte gelen DevMaster uygulamasını başlatın.
3. DevMaster uygulamasında, AIR for TV Aygıt Bağlayıcısı'nda gösterildiği gibi geliştirme bilgisayarınızın IP adresini girin.
4. DevMaster uygulamasında, Enable Remote Debugging (Uzaktan Hata Ayıklamayı Etkinleştir) öğesinin seçili olduğundan emin olun.
5. DevMaster uygulamasından çıkın.
6. Geliştirme bilgisayarında, AIR for TV Bağlayıcısı'nda Start (Başlat) öğesini seçin.
7. Donanım kiti aygıtında başka bir uygulama başlatın. İzleme bilgilerinin AIR for TV Aygıt Bağlayıcısı'nda görüntülendiğini doğrulayın.

   İzleme bilgileri görüntülenmiyorsa, geliştirme bilgisayarı ve donanım kiti aygıtı bağlı değildir. Geliştirme bilgisayarında izleme bilgileri için kullanılan bağlantı noktasının kullanılabilir olduğundan emin olun. AIR for TV Aygıt Bağlayıcısı'nda farklı bir bağlantı noktası seçebilirsiniz. Ayrıca, güvenlik duvarınızın seçilen bağlantı noktasına erişim sağladığından emin olun.

Ardından, Flash Builder'da hata ayıklayıcıyı başlatın. Aşağıdakileri yapın:

1. Flash Builder'da Run (Çalıştır) > Debug Configurations (Hata Ayıklama Yapılandırmaları) öğesini seçin.
2. Yerel hata ayıklamaya yönelik varolan hata ayıklama yapılandırmasından projenin adını kopyalayın.
3. Debug Configurations (Hata Ayıklama Yapılandırmaları) iletişim kutusunda Web Application (Web Uygulaması) öğesini seçin. Ardından New Launch Configuration (Yeni Başlangıç Yapılandırması) simgesini seçin.
4. Proje adını Project (Proje) alanına yapıştırın.

5 URL Or Path To Launch (Başlatmak için URL veya Yol) bölümünde Use Default (Varsayılanı Kullan) öğesindeki onay işaretini kaldırın. Ayrıca, metin alanına `about:blank` yazın.

6 Değişikliklerinizi kaydetmek için Apply (Uygula) öğesini seçin.

7 Flash Builder hata ayıklayıcısını başlatmak için Debug (Hata Ayıkla) öğesini seçin.

8 Uygulamanızı donanım kiti aygıtında başlatın.

Artık Flash Builder hata ayıklayıcısını, örneğin, kesme noktası ayarlamak ve değişkenleri incelemek için kullanabilirsiniz.

Bölüm 9: Adobe AIR için yerel uzantıları kullanma

Adobe AIR için yerel uzantılar, yerel kodda programlanan aygıta özgü işlevselliğe erişmenizi olanak veren ActionScript API'leri sağlar. Yerel uzantı geliştiricileri bazen aygıt üreticileriyle çalışır, bazen de üçüncü taraf geliştiricilerdir.

Yerel bir uzantı geliştiriyorsanız bkz. Adobe AIR için Yerel Uzantılar Geliştirme.

Yerel bir uzantı şunların bileşimidir:

- ActionScript sınıfları.
- Yerel kod.

Ancak yerel bir uzantı kullanan bir AIR uygulama geliştiricisi olarak yalnızca ActionScript sınıflarıyla çalışırsınız.

Yerel uzantılar aşağıdaki durumlarda kullanışlıdır:

- Yerel kod uygulaması platforma özgü özelliklere erişim sağlar. Platforma özgü bu özellikler yerleşik ActionScript sınıflarında mevcut değildir ve uygulamaya özgü ActionScript sınıflarında uygulanamaz. Yerel kod uygulaması aygıta özgü donanım ve yazılıma yönelik erişime sahip olduğundan dolayı bu tür bir işlevsellik sağlayabilir.
- Yerel kod uygulaması bazen yalnızca ActionScript kullanan bir uygulamadan daha hızlı olabilir.
- Yerel kod uygulaması eski yerel koda yönelik ActionScript erişimi sağlayabilir.

Bazı yerel uzantı örnekleri Adobe Geliştirici Merkezi'nde mevcuttur. Örneğin, bir yerel uzantı Android'in titreşim özelliğine yönelik AIR uygulamaları erişimi sağlar. Bkz. Adobe AIR için yerel uzantılar.

AIR Yerel Uzantı (ANE) dosyaları

Yerel uzantı geliştiricileri bir ANE dosyasına yerel uzantı paketlerler. ANE dosyası yerel uzantı için gerekli kütüphaneleri ve kaynakları içeren bir arşiv dosyasıdır.

Bazı aygıtlar için ANE dosyasının, yerel uzantının kullandığı yerel kod kütüphanesini içerdiğini unutmayın. Ancak diğer aygıtlar için yerel kod kütüphanesi aygıta yüklüdür. Bazı durumlarda yerel uzantıda belirli bir aygıt için hiçbir yerel kod yoktur. Yerel uzantı yalnızca ActionScript ile uygulanır.

AIR uygulama geliştiricisi olarak ANE dosyasını aşağıdaki gibi kullanırsınız:

- Bir SWC dosyasını kütüphane yoluna eklediğiniz şekilde ANE dosyasını uygulamanın kütüphane yoluna dahil edin. Bu uygulama uzantının ActionScript sınıflarıyla ilişki kurmasına olanak verir.

  Not: *Uygulamanızı derlerken, ANE için dinamik bağlantı kullandığınızdan emin olun. Flash Builder kullanıyorsanız, ActionScript Builder Yolu Özellikleri panelinde Harici seçeneğini belirtin; komut satırını kullanıyorsanız, -external-library-path öğesini belirtin.*

- ANE dosyasını AIR uygulamasıyla paketleyin.

Yerel uzantıları ve NativeProcess ActionScript sınıfını karşılaştırma

ActionScript 3.0 bir NativeProcess sınıfı sağlar. Bu sınıf, bir AIR uygulamasının ana bilgisayar işletim sisteminde yerel işlemleri çalıştırmasını sağlar. Bu özellik platforma özgü özelliklere ve kütüphanelere erişim sağlayan yerel uzantılara benzer. Yerel uzantı kullanmak yerine NativeProcess sınıfını kullanmaya karar verirken aşağıdakileri göz önünde bulundurun:

- Yalnızca `extendedDesktop` AIR profili NativeProcess sınıfını destekler. Bu nedenle, `mobileDevice` ve `extendedMobileDevice` AIR profillerine sahip uygulamalar için yerel uzantılar tek seçenektir.
- Yerel uzantı geliştiricileri çoğunlukla çeşitli platformlar için yerel uygulamalar sağlar, ancak sağladıkları ActionScript API platformlar arasında genelde aynıdır. NativeProcess sınıfını kullanırken yerel işlemi başlatacak ActionScript kodu farklı platformlar arasında değişebilir.
- NativeProcess sınıfı ayrı bir işlem başlatırken yerel bir uzantı AIR uygulamasıyla aynı işlemde çalışır. Bu nedenle, kod çökmesi konusunda kaygılanıyorsanız NativeProcess sınıfını kullanmak daha güvenlidir. Ancak ayrı işlem, büyük olasılıkla uygulamanız gereken işlemler arası iletişim işlemesine sahip olduğunuz anlamına gelmektedir.

Yerel uzantıları ve ActionScript sınıfı kütüphanelerini (SWC dosyaları) karşılaştırma

Bir SWC dosyası arşiv biçimindeki bir ActionScript sınıfı kütüphanesidir. SWC dosyası bir SWF dosyasını ve diğer kaynak dosyaları içerir. SWC dosyası bireysel ActionScript kodunu ve kaynak dosyaları paylaşma yerine ActionScript sınıflarını paylaşmaya yönelik uygun bir yöntemdir.

Yerel bir uzantı paketi bir ANE dosyasıdır. Bir SWC dosyası gibi bir ANE dosyası da SWF dosyasını ve arşiv biçimindeki diğer dosyaları içeren bir ActionScript sınıfı kütüphanesidir. Ancak, bir ANE dosyası ve bir SWC dosyası arasındaki en önemli fark yalnızca bir ANE dosyasının yerel bir kod kütüphanesi içerebileceğidir.

***Not:** Uygulamanızı derlerken, ANE dosyası için dinamik bağlantıyı kullandığınızdan emin olun. Flash Builder kullanıyorsanız, ActionScript Builder Yolu Özellikleri panelinde Harici seçeneğini belirtin; komut satırını kullanıyorsanız, -external-library-path öğesini belirtin.*

Daha fazla Yardım konusu

SWC dosyaları hakkında

Desteklenen aygıtlar

AIR 3'ten itibaren, uygulamalarda aşağıdaki aygıtlar için yerel uzantılar kullanabilirsiniz:

- Android 2.2'den itibaren Android aygıtları
- iOS 4.0'dan itibaren iOS aygıtları
- AIR 3.3'ten itibaren iOS Simulator
- Blackberry PlayBook
- AIR 3.0 uygulamasını destekleyen Windows masaüstü aygıtları

- AIR 3.0 uygulamasını destekleyen Mac OS X masaüstü aygıtları

Çoğunlukla aynı yerel uzantı birden çok platformu hedefler. Uzantının ANE dosyası desteklenen her platform için ActionScript ve yerel kütüphaneleri içerir. Genelde ActionScript kütüphanelerinde tüm platformlar için ortak arabirimler bulunur. Yerel kütüphaneler mutlaka farklıdır.

Bazen yerel bir uzantı varsayılan bir platformu destekler. Varsayılan platform uygulamasında yalnızca ActionScript kodu bulunur, ancak yerel kod bulunmaz. Uzantının özel olarak desteklemediği bir platform için bir uygulama paketlerseniz, uygulama yürütüldüğünde varsayılan uygulamayı kullanır. Örneğin, yalnızca mobil aygıtlara uygulanan bir özellik sağlayan bir uzantı düşünün. Uzantı, bir masaüstü uygulamasının özelliğin benzetimini yapmak için kullanabileceği varsayılan bir uygulama da sağlayabilir.

Desteklenen aygıt profilleri

Aşağıdaki AIR profilleri yerel uzantıları destekler:

- AIR 3.0'dan itibaren `extendedDesktop`
- AIR 3.0'dan itibaren `mobileDevice`
- AIR 3.0'dan itibaren `extendedMobileDevice`

Daha fazla Yardım konusu

AIR profili desteği

Yerel bir uzantı kullanmaya yönelik görev listesi

Uygulamanızda yerel bir uzantı kullanmak için aşağıdaki görevleri uygulayın:

1. Uygulama tanımlayıcı dosyanızdaki uzantıyı bildirin.
2. ANE dosyasını uygulamanızın kütüphane yoluna ekleyin.
3. Uygulamayı paketleyin.

Uygulama tanımlayıcı dosyanızdaki uzantıyı bildirme

Tüm AIR uygulamalarında bir uygulama tanımlayıcı dosyası bulunur. Uygulama yerel bir uzantı kullandığında, uygulama tanımlayıcı dosyası bir `<extensions>` öğesi içerir. Örneğin:

```
<extensions>
    <extensionID>com.example.Extension1</extensionID>
    <extensionID>com.example.Extension2</extensionID>
</extensions>
```

`extensionID` öğesi uzantı tanımlayıcı dosyasındaki `id` öğesiyle aynı değere sahiptir. Uzantı tanımlayıcı dosyası extension.xml adlı bir XML dosyasıdır. Bu dosya ANE dosyasında paketlenir. Uzantı extension.xml dosyasına bakmak için bir arşiv çıkarma aracı kullanabilirsiniz.

ANE dosyasını uygulamanızın kütüphane yoluna ekleme

Yerel bir uzantı kullanan bir uygulamayı derlemek için ANE dosyasını kütüphane yolunuza dahil edin.

ANE dosyasını Flash Builder ile kullanma

Uygulamanız yerel bir uzantı kullanıyorsa, kütüphane yolunuzda ActionScript uzantısı için ANE dosyasını dahil edin. Daha sonra ActionScript kodunuzu derlemek için Flash Builder'ı kullanabilirsiniz.

Flash Builder 4.5.1'in kullanıldığı aşağıdaki adımları uygulayın:

1. ANE dosyasının dosya adı uzantısını .ane'den .swc'ye değiştirin. Bu adım Flash Builder'ın dosyayı bulabilmesi için gereklidir.
2. Flash Builder projenizde Proje > Özellikler seçeneğini belirleyin.
3. Özellikler iletişim kutusunda Flex Build Path öğesini seçin.
4. Kütüphane Yolu sekmesinde SWC Ekle... öğesini seçin.
5. SWC dosyasını bulun ve Aç'ı seçin.
6. SWC Ekle... iletişim kutusunda Tamam'ı seçin.

   ANE dosyası artık Özellikler iletişim kutusundaki Kütüphane Yolu sekmesinde görünür.
7. SWC dosyası girişini genişletin. Library Path Item Options (Kütüphane Yolu Öğe Seçenekleri) iletişim kutusunu açmak için Link Type (Bağlantı Türü) öğesini çift tıklatın.
8. Library Path Item Options (Kütüphane Yolu Öğe Seçenekleri) iletişim kutusunda, Link Type (Bağlantı Türü) öğesini External (Harici) olarak değiştirin.

Artık uygulamanızı örneğin Project (Proje) > Build Project (Proje Oluştur) öğesini kullanarak derleyebilirsiniz.

ANE dosyasını Flash Professional ile kullanma

Uygulamanız yerel bir uzantı kullanıyorsa, kütüphane yolunuzda ActionScript uzantısı için ANE dosyasını dahil edin. Daha sonra ActionScript kodunuzu derlemek için Flash Professional CS5.5'i kullanabilirsiniz. Aşağıdakileri yapın:

1. ANE dosyasının dosya adı uzantısını .ane'den .swc'ye değiştirin. Bu adım Flash Professional'ın dosyayı bulabilmesi için gereklidir.
2. FLA dosyanızda Dosya > ActionScript Ayarları öğesini seçin.
3. Gelişmiş ActionScript 3.0 Ayarları iletişim kutusunda Kütüphane Yolu sekmesini seçin.
4. SWC Dosyasına Git düğmesini seçin.
5. SWC dosyasını bulun ve Aç'ı seçin.

   SWC dosyası artık Gelişmiş ActionScript 3.0 Ayarları iletişim kutusunda Kütüphane Yolu sekmesinde görünür
6. SWC dosyası seçiliyken, Bir Kütüphaneye Yönelik Bağlantı Seçeneklerini Ayarla düğmesini seçin.
7. Kütüphane Yolu Öğe Seçenekleri iletişim kutusunda, Bağlantı Türü öğesini Harici olarak değiştirin.

Yerel uzantılar kullanan bir uygulamayı paketleme

Yerel uzantılar kullanan bir uygulamayı paketlemek için ADT kullanın. Flash Professional CS5.5 veya Flash Builder 4.5.1 kullanarak uygulamayı paketleyemezsiniz.

ADT kullanma hakkında ayrıntılar AIR Geliştirici Aracı (ADT) sayfasında yer almaktadır.

Örneğin, aşağıdaki ADT komutu yerel uzantılar kullanan bir uygulama için bir DMG dosyası (Mac OS X için yerel bir yükleyici dosyası) oluşturur:

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

Aşağıdaki komut bir Android aygıtı için bir APK paketi oluşturur:

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

Aşağıdaki komut bir iPhone uygulaması için bir iOS paketi oluşturur:

```
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
```

Aşağıdakileri unutmayın:

- Yerel bir yükleyici paket türü kullanın.
- Uzantı dizinini belirtin.
- ANE dosyasının uygulamanın hedef aygıtını desteklediğinden emin olun.

Yerel bir yükleyici paket türü kullanın

Uygulama paketi yerel bir yükleyici olmalıdır. Yerel bir uzantı kullanan bir uygulama için platformlar arası bir AIR paketi (a .air paketi)oluşturamazsınız. Çünkü yerel uzantılar genellikle yerel kod içerir. Ancak, genellikle yerel bir uzantı aynı ActionScript API'lerine sahip birden çok yerel platformu destekler. Bu durumlarda aynı ANE dosyasını farklı yerel yükleyici paketlerinde kullanabilirsiniz.

Aşağıdaki tablo ADT komutunun `-target` seçeneği için kullanılacak değeri özetler:

Uygulamanın hedef platformu	**-target**
Mac OS X veya Windows masaüstü aygıtları	-target native -target bundle
Android	-target apk veya diğer Android paketi hedefleri.
iOS	-target ipa-ad-hoc veya diğer iOS paketi hedefleri
iOS Simulator	-target ipa-test-interpreter-simulator -target ipa-debug-interpreter-simulator

Uzantı dizinini belirtin

Yerel uzantılar (ANE dosyaları) içeren dizini ADT'ye bildirmek için ADT seçeneği `-extdir` parametresini kullanın.

Bu seçenek hakkında ayrıntılar için bkz. “Dosya ve yol seçenekleri” sayfa 179.

ANE dosyasının uygulamanın hedef aygıtını desteklediğinden emin olun

Bir ANE dosyası sağlarken, yerel uzantı geliştiricisi uzantının hangi platformları desteklediği konusunda sizi bilgilendirir. ANE dosyasının içeriklerine bakmak için bir arşiv çıkarma aracı da kullanabilirsiniz. Çıkarılan dosyalar desteklenen her platform için bir dizin içerir.

Uzantının hangi platformları desteklediğini bilmek ANE dosyasını kullanan uygulamanın paketlenmesi sırasında önemlidir. Aşağıdaki kuralları göz önünde bulundurun:

- Bir Android uygulama paketi oluşturmak için ANE dosyası `Android-ARM` platformunu içermelidir. Alternatif olarak, ANE dosyası varsayılan platformu ve en az bir adet başka platformu içermelidir.
- Bir iOS uygulama paketi oluşturmak için ANE dosyası `iPhone-ARM` platformunu içermelidir. Alternatif olarak, ANE dosyası varsayılan platformu ve en az bir adet başka platformu içermelidir.
- Bir iOS Simulator uygulama paketi oluşturmak için ANE dosyası `iPhone-x86` platformunu içermelidir.
- Bir Mac OS X uygulama paketi oluşturmak için ANE dosyası `MacOS-x86` platformunu içermelidir. Alternatif olarak, ANE dosyası varsayılan platformu ve en az bir adet başka platformu içermelidir.
- Bir Windows uygulama paketi oluşturmak için ANE dosyası `Windows-x86` platformunu içermelidir. Alternatif olarak, ANE dosyası varsayılan platformu ve en az bir adet başka platformu içermelidir.

Bölüm 10: ActionScript derleyicileri

ActionScript ve MXML kodu bir AIR uygulamasına dahil edilmeden önce derlenmelidir. Adobe Flash Builder veya Adobe Flash Professional gibi bir Tümleşik Yazılım Geliştirme Ortamı (IDE) kullanıyorsanız, IDE derleme işlemini arka planda gerçekleştirir. Ancak, bir IDE kullanmadığınızda veya bir komut dosyası oluştururken SWF dosyalarınızı oluşturmak için komut satırından da ActionScript derleyicilerini başlatabilirsiniz.

Flex SDK'deki AIR komut satırı araçları hakkında

Bir Adobe AIR uygulaması oluşturmak için kullandığınız komut satırı araçlarının her biri, uygulamaları oluşturmak için kullanılan karşılık gelen aracı çağırır.

- amxmlc, uygulama sınıflarını derlemek için mxmlc öğesini çağırır.
- acompc, kütüphane ve bileşen sınıflarını derlemek için compc öğesini çağırır.
- aasdoc, kaynak kodu yorumlarından belge dosyaları oluşturmak için asdoc öğesini çağırır.

Yardımcı programların Flex ve AIR sürümleri arasındaki tek fark AIR sürümlerinin yapılandırma seçeneklerini flex-config.xml yerine air-config.xml dosyasından yüklemesidir.

Flex SDK araçları ve komut satırı seçenekleri, Flex belgeleri içinde tam olarak açıklanmıştır. Burada, Flex SDK araçları başlamanıza yardımcı olmak ve Flex uygulamaları ve AIR uygulamaları oluşturma arasındaki farkları belirtmek amacıyla temel bir düzeyde açıklanmaktadır.

Daha fazla Yardım konusu

"Flex SDK ile ilk masaüstü AIR uygulamanızı oluşturma" sayfa 35

Derleyici kurulumu

Genelde derleme seçeneklerini hem komut satırında hem de bir veya daha fazla yapılandırma dosyasıyla belirlersiniz. Global Flex SDK yapılandırma dosyası derleyicilerin çalıştığı her yerde kullanılan varsayılan değerler içerir. Bu dosyayı kendi geliştirme ortamınıza uyacak şekilde düzenleyebilirsiniz. Flex SDK yüklemenizin çerçeveler dizininde bulunan iki global Flex yapılandırma dosyası vardır. air-config.xml dosyası, amxmlc derleyicisini çalıştırdığınızda kullanılır. Bu dosya AIR kütüphanelerini de dahil ederek AIR için olan derleyiciyi yapılandırır. mxmlc dosyasını çalıştırdığınızda flex-config.xml dosyası kullanılır.

Varsayılan yapılandırma değerleri, Flex ve AIR'nin nasıl çalıştığını keşfetmek için uygundur ancak geniş çaplı bir projeye giriştiğinizde mevcut seçenekleri daha yakından inceleyin. Belirli bir proje için global değerlerin önüne geçen bir yerel yapılandırma dosyasında derleme seçenekleri için projeye özel değerler sağlayabilirsiniz.

Not: *Özellikle AIR uygulamaları için kullanılan derleme seçenekleri yoktur ancak bir AIR uygulamasını derlerken AIR kütüphanelerine başvurmanız gerekir. Genelde, bu kütüphanelere proje düzeyi yapılandırma dosyasında, Ant gibi bir oluşturma aracı veya doğrudan komut listesinde başvurulur.*

MXML ve ActionScript dosyalarını AIR için derleme

Komut satırı MXML derleyicisiyle (amxmlc) AIR uygulamanızın Adobe® ActionScript® 3.0 ve MXML varlıklarını derleyebilirsiniz. (HTML tabanlı uygulamaları derlemeniz gerekmez. Flash Professional'da bir SWF derlemek için, sadece filmi bir SWF dosyasına yayınlayın.)

amxmlc'yi kullanmaya ilişkin temel komut satırı deseni şu şekildedir:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

burada *[compiler options]*, AIR uygulamanızı derlemek için kullanılan komut satırı seçeneklerini belirtir.

amxmlc komutu standart Flex mxmlc derleyicisini ek bir parametre olan `+configname=air` parametresiyle çağırır. Bu parametre derleyiciye flex-config.xml dosyası yerine air-config.xml dosyasını kullanması talimatını verir. amxmlc kullanmak diğer bakımlardan mxmlc kullanmakla aynıdır.

Derleyici, bir AIR uygulamasını derlemek için genellikle gerekli olan AIR ve Flex kütüphanelerini belirterek air-config.xml yapılandırma dosyasını yükler. Ayrıca global yapılandırmayı geçersiz kılmak veya bu yapılandırmaya ek seçenekler eklemek için yerel, proje düzeyinde bir yapılandırma dosyası da kullanabilirsiniz. Genellikle yerel bir yapılandırma dosyası oluşturmanın en kolay yolu, global sürümünün bir kopyasını düzenlemektir. Yerel dosyayı `-load-config` seçeneğini kullanarak yükleyebilirsiniz:

-load-config=project-config.xml Global seçenekleri geçersiz kılar.

-load-config+=project-config.xml `-library-path` seçeneği gibi, değerden daha fazlasını alan global seçeneklere ek değerler ekler. Yalnızca tek bir değer alan global seçenekler geçersiz kılınır.

Yerel yapılandırma dosyası için özel bir adlandırma kuralı kullanırsanız, amxmlc derleyicisi yerel dosyayı otomatik olarak yükler. Örneğin ana MXML dosyası `RunningMan.mxml` dosyasıysa, yerel yapılandırma dosyasını şöyle adlandırın: `RunningMan-config.xml`. Şimdi uygulamayı derlemek için tek yapmanız gereken şunu yazmaktır:

```
amxmlc RunningMan.mxml
```

Dosya adı derlenen MXML dosyasının adıyla eşleştiği için, `RunningMan-config.xml` otomatik olarak yüklenir.

amxmlc örnekleri

Aşağıdaki örnekler amxmlc derleyicisinin kullanımı gösterir. (Yalnızca uygulamanızın ActionScript ve MXML varlıkları derlenmelidir.)

AIR MXML dosyasını derlemek için:

```
amxmlc myApp.mxml
```

Çıktı adını derleyin ve ayarlayın:

```
amxmlc –output anApp.swf -- myApp.mxml
```

Bir AIR ActionScript dosyası derleyin:

```
amxmlc myApp.as
```

Derleyici yapılandırma dosyası belirtin:

```
amxmlc –load-config config.xml -- myApp.mxml
```

Başka bir yapılandırma dosyasından ek seçenekler ekleyin:

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

Komut satırına kütüphane ekleyin (yapılandırma dosyasında zaten varolan kütüphanelere ek olarak):

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

Bir AIR MXML dosyasını yapılandırma dosyası kullanmadan derleyin (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
```

Bir AIR MXML dosyasını yapılandırma dosyası kullanmadan derleyin (Mac OS X veya 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
```

Bir AIR MXML dosyasını çalışma zamanı paylaşılan kütüphanesi kullanmak üzere derleyin:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --
myApp.mxml
```

Bir ANE kullanmak için (ANE için `-external-library-path` kullandığınızdan emin olun) bir AIR MXML dosyası derleyin:

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

Java'dan derleme (sınıf yolu `mxmlc.jar` dosyasını içerecek şekilde ayarlanmış olarak):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional
compiler options] -- myApp.mxml
```

flexlib seçeneği, derleyicinin flex_config.xml dosyasını bulmasını sağlayarak Flex SDK çerçeveleri dizininizin konumunu tanımlar.

Java'dan derleme (sınıf yolu ayarlanmadan):

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air
[additional compiler options] -- myApp.mxml
```

Örnek, derleyiciyi Apache Ant kullanarak çağırmak için mxmlc.jar dosyasını çalıştırmak üzere bir Java görevi kullanır):

```
<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>
```

Bir AIR bileşeni veya kod kütüphanesi derleme (Flex)

AIR kütüphanelerini ve bağımsız bileşenleri derlemek için, acompc bileşen derleyicisini kullanın. acompc bileşen derleyicisi, aşağıdaki durumlar haricinde amxmlc derleyicisi gibi davranır:

- Kod tabanındaki hangi sınıfların kitaplığa veya bileşene dahil edileceğini belirtmelisiniz.

- acompc otomatik olarak yerel yapılandırma dosyası aramaz. Bir proje konfigürasyon dosyası kullanmak için, –load-config seçeneğini kullanmalısınız.

acompc komutu standart Flex compc bileşen derleyicisini çağırır, ancak yapılandırma seçeneklerini `flex-config.xml` dosyası yerine `air-config.xml` dosyasından yükler.

Bileşen derleyici yapılandırma dosyası

Kaynak yolunu ve sınıf adlarını komut satırına yazma işleminden (ve yanlış yazma olasılığından) kaçınmak için yerel yapılandırma dosyası kullanın. Yerel yapılandırma dosyasını yüklemek için acompc komut satırına -load-config seçeneğini ekleyin.

Aşağıdaki örnekte, her ikisi de pakette bulunan ParticleManager ve Particle adlı iki sınıfa sahip bir kütüphane oluşturmaya yönelik bir konfigürasyon gösterilmektedir: `com.adobe.samples.particles`. Sınıf dosyaları `source/com/adobe/samples/particles` klasöründe bulunmaktadır.

```
<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>
```

Kitaplığı `ParticleLib-config.xml` adlı yapılandırma dosyasını kullanarak derlemek için, şunu yazın:

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Aynı komutu komut satırının tamamında çalıştırmak için, şunu yazın:

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Komutun tamamını tek bir satıra yazın veya komut kabuğunuz için satır devam karakterini kullanın.)

acompc örnekleri

Bu örnekler `myLib-config.xml` adlı bir yapılandırma dosyası kullandığınızı varsayar.

Bir AIR bileşeni veya kitaplığı derleyin:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Bir çalışma zamanı paylaşılan kütüphanesi derleyin:

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Komutu çalıştırmadan önce lib klasörünün boş bir şekilde mevcut olması gerektiğini unutmayın.)

Bölüm 11: AIR Hata Ayıklama Başlatıcısı (ADL)

AIR Hata Ayıklama Başlatıcısı'nı (ADL) geliştirme sırasında hem SWF tabanlı hem de HTML tabanlı uygulamaları çalıştırmak için kullanın. ADL kullanarak bir uygulamayı paketlemeden ve yüklemeden çalıştırabilirsiniz. Varsayılan olarak ADL, SDK'ye dahil olan bir çalışma zamanı kullanır, yani ADL'yi kullanmak için çalışma zamanını ayrı olarak yüklemeniz gerekmez.

ADL, izleme ifadelerini ve çalışma zamanı hatalarını standart çıktıya yazdırır, ancak kesme noktalarını veya diğer hata ayıklama özelliklerini desteklemez. Karmaşık hata ayıklama sorunları için Flash Hata Ayıklayıcı'yı (veya Flash Builder gibi bir entegre geliştirme ortamını) kullanabilirsiniz.

Not: *`trace()` ifadeleriniz konsolda görüntülenmiyorsa, mm.cfg dosyasında `ErrorReportingEnable` veya `TraceOutputFileEnable` öğesini belirtmediğinizden emin olun. Bu dosyanın platforma özgü konumuyla ilgili daha fazla bilgi için bkz. mm.cfg dosyasını düzenleme.*

AIR doğrudan hata ayıklamayı destekler, bu yüzden çalışma zamanının bir hata ayıklama sürümüne gerek olmaz (Ancak bu sürüme Adobe® Flash® Player'da gerek olurdu). Komut satırı hata ayıklamasını yönetmek için, Flash Hata Ayıklayıcısı ve AIR Hata Ayıklama Başlatıcısı'nı (ADL) kullanın.

Flash Hata Ayıklayıcısı Flex SDK dizininde dağıtılır. Windows'taki fdb.exe gibi yerel sürümler bin alt dizinindedir. Java sürümü lib alt dizinindedir. AIR Hata Ayıklama Başlatıcısı, adl.exe, Flex SDK yüklemenizin bin dizinindedir. (Ayrı bir Java sürümü yoktur).

Not: *Fdb onu Flash Player ile açmaya çalışacağından, bir AIR uygulamasını doğrudan fdb ile başlatamazsınız. Onun yerine, AIR uygulamasının çalışan bir fdb oturumuna bağlanmasına izin verin.*

ADL kullanımı

ADL ile bir uygulama çalıştırmak için aşağıdaki deseni kullanın:

```
adl application.xml
```

Uygulamanın uygulama tanımlayıcı dosyası *application.xml* olduğunda.

ADL için tam söz dizimi şudur:

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(Köşeli parantez [] içindeki öğeler isteğe bağlıdır.)

-runtime *runtime-directory* Kullanılacak çalışma zamanını içeren dizini belirtir. Belirtilmemişse, ADL programıyla aynı SDK'de bulunan çalışma zamanı dizini kullanılır. ADL'i SDK klasörünün dışına taşırsanız, çalışma zamanı dizinini belirtin. Windows ve Linux'ta, `Adobe AIR` dizinini içeren dizini belirtin. Mac OS X'te, `Adobe AIR.framework` içeren dizini belirtin.

***-pubid publisher-id* Bu çalıştırma için AIR uygulamasının yayıncı kimliği olarak belirtilen değeri atar.** Geçici bir yayıncı belirlemek, yerel bağlantı üzerinden iletişim kurma gibi, bir uygulamanın benzersiz olarak tanımlanmasına yardımcı olan yayıncı kimliğini kullanan AIR uygulaması özelliklerini test etmenizi sağlar. AIR 1.5.3'te olduğu gibi, uygulama açıklayıcı dosyasındaki yayıncı kimliğini de belirleyebilirsiniz (ve bu parametreyi kullanmamalısınız).

***Not:** Bir Yayıncı Kimliği, AIR 1.5.3'ten itibaren artık otomatik olarak tahmin edilmez ve bir AIR uygulamasına atanmaz. Mevcut bir AIR uygulaması için bir güncelleme oluştururken bir yayıncı kimliği belirtebilirsiniz, ancak yeni uygulamalar yayıncı kimliği gerektirmez ve belirtmemelidir.*

-nodebug Hata ayıklama desteğini kapatır. Kullanılırsa, uygulama işlemi Flash hata ayıklayıcıya bağlanamaz ve işlenmeyen istisnaların iletişim kutuları bastırılır. (Ancak, izleme ifadeleri konsol penceresine yazdırmaya devam eder.) Hata ayıklamanın kapatılması, uygulamanızın biraz daha hızlı çalışmasını sağlar ve ayrıca yüklenmiş bir uygulamanın çalışma modunu daha iyi taklit eder.

-atlogin Oturum açma sırasında uygulamanın başlatılmasını benzetir. Bu bayrak, yalnızca bir uygulama kullanıcı oturum açtığında başlatılmak üzere ayarlandığında yürütülen uygulama mantığını test etmenizi sağlar. `-atlogin` kullanıldığında, uygulamaya gönderilen InvokeEvent nesnesinin `reason` özelliği `standard` yerine `login` olur (uygulama zaten çalışmıyorsa).

***-profile profileName* ADL belirtilen profili kullanarak uygulamanın hatalarını ayıklar.** *`profileName`* aşağıdaki değerlerden biri olabilir:

- desktop
- extendedDesktop
- mobileDevice

Uygulama tanımlayıcısı bir `supportedProfiles` öğesi içeriyorsa, `-profile` ile belirttiğiniz profil desteklenen listenin bir üyesi olmalıdır. `-profile` bayrağı kullanılmıyorsa uygulama tanımlayıcısındaki ilk profil etkin profil olarak kullanılır. Uygulama tanımlayıcısı `supportedProfiles` öğesini içermiyorsa ve `-profile` bayrağını kullanmıyorsanız, *desktop* profili kullanılır.

Daha fazla bilgi için bkz. "supportedProfiles" sayfa 236 ve "Aygıt profilleri" sayfa 242.

-screensize *value* Masaüstünde mobileDevice profilinde uygulama çalıştırırken kullanılacak taklit edilen ekran boyutu. Ekran boyutunu önceden tanımlanmış ekran türü olarak veya dikey mizanpaj için normal genişlik ve yükseklik artı tam ekran genişliği ve yüksekliğinin piksel boyutları olarak belirtin. Değeri türe göre belirtmek için aşağıdaki önceden belirlenmiş ekran türlerinden birini kullanın:

Ekran türü	Normal genişlik x yükseklik	Tam ekran genişliği x yüksekliği
480	720 x 480	720 x 480
720	1280 x 720	1280 x 720
1080	1920 x 1080	1920 x 1080
Droid	480 x 816	480 x 854
FWQVGA	240 x 432	240 x 432
FWVGA	480 x 854	480 x 854
HVGA	320 x 480	320 x 480

Ekran türü	Normal genişlik x yükseklik	Tam ekran genişliği x yüksekliği
iPad	768 x 1004	768 x 1024
iPadRetina	1536 x 2008	1536 x 2048
iPhone	320 x 460	320 x 480
iPhoneRetina	640 x 920	640 x 960
iPhone5Retina	640 x 1096	640 x 1136
iPhone6	750 x 1294	750 x 1334
iPhone6Plus	1242 x 2148	1242 x 2208
iPod	320 x 460	320 x 480
iPodRetina	640 x 920	640 x 960
iPod5Retina	640 x 1096	640 x 1136
NexusOne	480 x 762	480 x 800
QVGA	240 x 320	240 x 320
SamsungGalaxyS	480 x 762	480 x 800
SamsungGalaxyTab	600 x 986	600 x 1024
WQVGA	240 x 400	240 x 400
WVGA	480 x 800	480 x 800

Piksel boyutlarını doğrudan belirtmek için aşağıdaki biçimi kullanın:

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Dikey mizanpaj için piksel boyutlarını her zaman belirtin. Bu, genişlik değerinin yükseklik değerinden daha küçük olarak belirtilmesi anlamına gelir. Örneğin, NexusOne ekranı şununla belirtilebilir:

```
-screensize 480x762:480x800
```

-extdir *extension-directory* Çalışma zamanının yerel uzantıları araması gerektiği dizin. Dizin uygulamanın kullandığı her yerel uzantı için bir alt dizin içerir. Bu dizinlerin her biri, bir uzantının *paketlenmemiş* ANE dosyasını içerir. Örneğin:

```
C:\extensionDirs\
    extension1.ane\
        META-INF\
            ANE\
                Android-ARM\
                    library.swf
                    extension1.jar
                extension.xml
            signatures.xml
        catalog.xml
        library.swf
        mimetype
    extension2.ane\
        META-INF\
            ANE\
                Android-ARM\
                    library.swf
                    extension2.jar
                extension.xml
            signatures.xml
        catalog.xml
        library.swf
        mimetype
```

-extdir parametresini kullanırken aşağıdakileri göz önünde bulundurun:

- ADL komutu, belirtilen her bir dizinin .ane dosya adı uzantısına sahip olmasını gerektirir. Ancak, dosya adının ".ane" sonekinden önceki bölümü geçerli herhangi bir dosya adı olabilir. Bu bölümün uygulama tanımlayıcı dosyasının `extensionID` uzantısının değeriyle eşleşmesi *gerekmez*.
- `-extdir` parametresini bir kereden fazla belirtebilirsiniz.
- `-extdir` parametresinin kullanımı ADT aracı ve ADL aracı için farklıdır. ADT'de, parametre ANE dosyalarını içeren bir dizini belirtir.
- Uzantı dizinlerini belirtmek için ortam değişkeni `AIR_EXTENSION_PATH` öğesini de kullanabilirsiniz. Bkz. "ADT ortam değişkenleri" sayfa 185.

application.xml Uygulama tanımlayıcı dosyası. Bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201. Uygulama açıklayıcısı ADL tarafından gerekli görülen tek parametredir ve çoğu zaman ihtiyaç duyulan tek parametredir.

root-directory Uygulamanın çalıştırılacak kök dizinini belirtir. Belirtilmemişse, uygulama tanımlayıcı dosyasını içeren dizin kullanılır.

-- *arguments* "--" karakterlerinden sonra gelen tüm karakter dizeleri uygulamaya komut satırı argümanları olarak iletilir.

Not: *Zaten çalışmakta olan bir AIR uygulamasını başlattığınızda, bu uygulamanın yeni bir örneği başlatılmaz. Bunun yerine çalışan örneğe bir `invoke` olayı gönderilir.*

ADL Örnekleri

Geçerli dizinde bir uygulama çalıştırın:

```
adl myApp-app.xml
```

Geçerli dizinin alt dizininde bir uygulama çalıştırın:

```
adl source/myApp-app.xml release
```

Bir uygulama çalıştırın ve "tick" ve "tock" olmak üzere iki komut satırı argümanını iletin:

```
adl myApp-app.xml -- tick tock
```

Belirli bir çalışma zamanını kullanarak bir uygulama çalıştırın:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Hata ayıklama desteği olmadan bir uygulama çalıştırın:

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

Mobil aygıt profilinde bir uygulama çalıştırın ve Nexus One ekran boyutunun benzetimini yapın:

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Uygulamayı çalıştırmak için Apache Ant kullanarak bir uygulama çalıştırın (örnekte gösterilen yollar Windows içindir):

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADL" value="${SDK_HOME}/bin/adl.exe"/>
<property name="APP_ROOT" value="c:/dev/MyApp/bin-debug"/>
<property name="APP_DESCRIPTOR" value="${APP_ROOT}/myApp-app.xml"/>

<target name="test">
    <exec executable="${ADL}">
        <arg value="${APP_DESCRIPTOR}"/>
        <arg value="${APP_ROOT}"/>
    </exec>
</target>
```

ADL çıkış ve hata kodları

Aşağıdaki tablo, ADL tarafından yazdırılan çıkış kodlarını açıklamaktadır:

Çıkış kodu	Açıklama
0	Başarılı başlatma. ADL, AIR uygulaması çıktıktan sonra çıkar.
1	Zaten çalışmakta olan AIR uygulamasını başarılı bir şekilde başlatma. ADL hemen çıkar.
2	Kullanım hatası. ADL'ye verilen argümanlar yanlış.
3	Çalışma zamanı bulunamadı.
4	Çalışma zamanı başlatılamadı. Bunun nedeni genellikle uygulamada belirtilen sürümün, çalışma zamanının sürümüyle eşleşmemesidir.
5	Bilinmeyen nedenlerden kaynaklanan bir hata oluştu.
6	Uygulama tanımlayıcı dosyası bulunamadı.
7	Uygulama tanımlayıcının içeriği geçersiz. Bu hata genellikle XML'in doğru biçimlendirilmediğini belirtir.
8	Ana uygulama içerik dosyası (uygulama tanımlayıcı dosyasının <content> öğesinde belirtilen) bulunamadı.

Çıkış kodu	Açıklama
9	Ana uygulama içerik dosyası geçerli bir SWF veya HTML dosyası değil.
10	Uygulama, -profil seçeneğiyle belirtilen profili desteklemez.
11	-screensize argümanı geçerli dosyada desteklenmez.

Bölüm 12: AIR Geliştirici Aracı (ADT)

AIR Geliştirici Aracı (ADT), AIR uygulamaları geliştirmek için kullanılan çok amaçlı komut satırı aracıdır. Aşağıdaki görevleri gerçekleştirmek için ADT'yi kullanabilirsiniz:

- Bir AIR uygulamasını .air yükleme dosyası olarak paketleme
- Bir AIR uygulamasını yerel yükleyici olarak paketleme. Örneğin, Windows'ta .exe yükleyici dosyası, iOS'de .ipa ya da Android'de .apk olarak paketleme.
- AIR Yerel Uzantı (ANE) dosyası olarak yerel bir uzantıyı paketleyin.
- Bir AIR uygulamasını dijital sertifikayla imzalama
- Uygulama güncellemeleri için kullanılan dijital imzayı değiştirme (geçiş yapma)
- Bir bilgisayara bağlı aygıtları belirleme
- Kendinden imzalı dijital kod imzalama sertifikası oluşturma
- Mobil bir aygıtta uygulamayı uzaktan yüklemeyi başlatma ve kaldırma
- Mobil bir aygıtta AIR çalışma zamanını uzaktan yükleme ve kaldırma

ADT, AIR SDK'ye dahil olan bir Java programıdır. Kullanmak için Java 1.5 veya daha yüksek bir sürümüne sahip olmanız gerekir. SDK, ADT'yi başlatmak için bir komut dosyası içerir. Bu komut dosyasını kullanmak için Java programının konumu path ortam değişkeninde bulunmalıdır. Path ortam değişkeninizde AIR SDK `bin` dizini de listeleniyorsa, ADT'yi başlatmak için uygun argümanlarla birlikte komut satırına `adt` yazabilirsiniz. (Path ortam değişkenini nasıl ayarlayacağınızı bilmiyorsanız, işletim sistemi belgelerinize bakın. Daha fazla yardım için, birçok bilgisayar sisteminde yolu ayarlama yöntemleri “Path ortam değişkenleri” sayfa 300 içinde açıklanır.)

ADT'yi kullanmak için en az 2 GB bilgisayar belleği gereklidir. Bundan daha az belleğe sahipseniz, özellikle iOS için uygulama paketlerken bellek yetersiz kalabilir.

Path değişkeninde hem Java hem de AIR SDK bin dizininin bulunduğunu varsayarak, aşağıdaki temel sözdizimini kullanarak ADT'yi çalıştırabilirsiniz:

```
adt -command options
```

Not: *Adobe Flash Builder ve Adobe Flash Professional dahil birçok entegre geliştirme ortamı sizin için AIR uygulamalarını paketleyebilir ve imzalayabilir. Zaten böyle bir geliştirme ortamı kullanıyorsanız, genelde bu ortak görevler için ADT kullanmanız gerekmez. Ancak, entegre geliştirme ortamınız tarafından desteklenmeyen işlevler için komut satırı aracı olarak ADT'yi kullanmanız gerekebilir. Ek olarak, ADT'yi otomatik derleme sürecinin bir parçası şeklinde komut satırı aracı olarak kullanabilirsiniz.*

ADT komutları

ADT'ye iletilen ilk argüman aşağıdaki komutlardan birini belirtir.

- -package — bir AIR uygulamasını veya AIR Yerel Uzantısı'nı (ANE) paketler.
- -prepare — bir AIR uygulamasını ara dosya (AIRI) olarak paketler ancak imzalamaz. AIRI dosyaları yüklenemez.
- -sign — -prepare komutuyla oluşturulmuş bir AIRI paketini imzalar. -prepare ve -sign komutları paketleme ve imzalama işlemlerinin farklı zamanlarda gerçekleştirilmesine izin verir. Ayrıca bir ANE paketini imzalamak veya yeniden imzalamak için de -sign komutunu kullanabilirsiniz.

- -migrate — imzalı bir AIR paketine geçiş imzası uygular. Bu yeni veya yenilenmiş bir kod imzalama sertifikası kullanmanıza izin verir.
- -certificate — kendinden imzalı dijital kod imzalama sertifikası oluşturur.
- -checkstore — bir anahtar deposundaki dijital sertifikaya erişilebildiğini doğrular.
- -installApp — bir aygıtta veya aygıt taklitçisinde AIR uygulaması yükler.
- -launchApp — bir aygıtta veya aygıt taklitçisinde bir AIR uygulaması başlatır.
- -appVersion — bir aygıtta veya aygıt taklitçisinde geçerli olarak yüklü durumda olan AIR uygulamasının sürümünü bildirir.
- -uninstallApp — bir aygıttan veya aygıt taklitçisinden AIR uygulamasını kaldırır.
- -installRuntime — bir aygıta veya aygıt taklitçisine AIR çalışma zamanını yükler.
- -runtimeVersion — bir aygıtta veya aygıt taklitçisinde geçerli olarak yüklü olan AIR çalışma zamanının sürümünü bildirir.
- -uninstallRuntime — bir aygıtta veya aygıt taklitçisinde geçerli olarak yüklü olan AIR çalışma zamanını kaldırır.
- -version — ADT sürüm numarasını bildirir.
- -devices — bağlı mobil aygıtlar veya taklitçilere ilişkin aygıt bilgilerini raporlar.
- -help — komutların ve seçeneklerin listesini görüntüler.

Birçok ADT komutu ilişkili seçenek bayrağı ve parametreleri kümelerini paylaşır. Bu seçenek kümeleri ayrı ayrı ayrıntılı olarak açıklanmaktadır:

- "ADT kod imzalama seçenekleri" sayfa 177
- "Dosya ve yol seçenekleri" sayfa 179
- "Hata ayıklayıcı bağlantısı seçenekleri" sayfa 180
- "Yerel uzantı seçenekleri" sayfa 181

ADT package komutu

`-package` komutu ana uygulama dizininden çalıştırılmalıdır. Komut aşağıdaki sözdizimlerini kullanır:

Bileşen uygulama dosyalarından bir AIR paketi oluşturma:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Bileşen uygulama dosyalarından yerel bir paket oluşturma:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Bileşen uygulama dosyalarından yerel uzantı içeren bir yerel paket oluşturun:

```
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
```

Bir AIR veya AIRI dosyasından yerel paket oluşturma:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Bileşen yerel uzantı dosyalarından yerel bir uzantı paketi oluşturun.

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Not: *ANE dosyasını imzalamanız gerekmez, bu nedenle* `AIR_SIGNING_OPTIONS` *parametreleri bu örnekte isteğe bağlıdır.*

AIR_SIGNING_OPTIONS AIR imzalama seçenekleri, AIR yükleme dosyasını imzalamak için kullanılmış sertifikayı tanımlar. İmzalama seçenekleri tam olarak "ADT kod imzalama seçenekleri" sayfa 177 bölümünde açıklanır.

-migrate Bu bayrak, uygulamanın `AIR_SIGNING_OPTIONS` parametrelerinde belirtilen sertifikanın yanı sıra bir geçiş sertifikasıyla imzalandığını belirtir. Bu bayrak, yalnızca bir masaüstü uygulamasını yerel yükleyici olarak paketliyorsanız ve uygulama yerel bir uzantı kullanıyorsa geçerlidir. Diğer durumlarda bir hata oluşur. Geçiş sertifikasına yönelik imzalama seçenekleri **MIGRATION_SIGNING_OPTIONS** parametreleri olarak belirtilir. Bu imzalama seçenekleri "ADT kod imzalama seçenekleri" sayfa 177 bölümünde tam olarak açıklanır. `-migrate` bayrağını kullanmak, yerel uzantı kullanan bir masaüstü yerel yükleyici uygulaması için güncelleme oluşturmanıza ve uygulamanın kod imzalama sertifikasını değiştirmenize (örn. orijinal sertifikanın süresi dolduğunda) olanak tanır. Daha fazla bilgi için bkz. "Bir AIR uygulamasının güncellenmiş sürümünü imzalama" sayfa 195.

`-package` komutunun `-migrate` bayrağı AIR 3.6 ve üst sürümlerinde kullanılabilir.

-target Oluşturulacak paket türü. Desteklenen paket türleri şunlardır:

- air — bir AIR paketi. "air" varsayılan değerdir ve AIR veya AIRI dosyaları oluştururken -target bayrağının belirtilmesi gerekmez.
- airn — genişletilmiş televizyon profilindeki aygıtlar için yerel bir uygulama paketi.
- ane —bir AIR yerel uzantı paketi
- Android paketi şunları hedef alır:
  - apk — bir Android paketi. Bu hedefle üretilen bir paket yalnızca bir Android aygıtına yüklenebilir. Taklitçiye yüklenemez.
  - apk-captive-runtime — Uygulamayı ve AIR çalışma zamanının sabit bir sürümünü içeren bir Android paketi. Bu hedefle üretilen bir paket yalnızca bir Android aygıtına yüklenebilir. Taklitçiye yüklenemez.
  - apk-debug — ekstra hata ayıklama bilgilerine sahip bir Android paketi. (Uygulamadaki SWF dosyaları da hata ayıklama desteğiyle derlenmelidir.)
  - apk-emulator — hata ayıklama desteği olmayan bir taklitçide kullanmak için Android paketi. (Hem taklitçilerde hem aygıtlarda hata ayıklamaya izin vermek için apk-debug hedefini kullanın.)
  - apk-profile — uygulama performansını ve bellek profilini destekleyen bir Android paketi.
- iOS paketi şunları hedef alır:
  - ipa-ad-hoc — geçici dağıtım için bir iOS paketi.
  - ipa-app-store — Apple App store dağıtımı için bir iOS paketi.
  - ipa-debug — ekstra hata ayıklama bilgilerine sahip bir iOS paketi. (Uygulamadaki SWF dosyaları da hata ayıklama desteğiyle derlenmelidir.)
  - ipa-test — en iyileştirme veya hata ayıklama bilgisi olmadan derlenen bir iOS paketi.
  - ipa-debug-interpreter — işlevsel olarak hata ayıklama paketine eşittir ancak daha hızlı derler. Ancak, ActionScript bayt kodu yorumlanır ve makine koduna dönüştürülmez. Sonuç olarak, kod yürütme işlemi yorumlayıcı paketinden daha yavaştır.
  - ipa-debug-interpreter-simulator — işlevsel olarak ipa-debug-interpreter seçeneğine eşittir ancak iOS simulator için paketlenir. Yalnızca Macintosh. Bu seçeneği kullanırsanız iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini de dahil etmeniz gerekir.
  - ipa-test-interpreter — işlevsel olarak test paketine eşittir ancak daha hızlı derler. Ancak, ActionScript bayt kodu yorumlanır ve makine koduna dönüştürülmez. Sonuç olarak, kod yürütme işlemi yorumlayıcı paketinden daha yavaştır.
  - ipa-test-interpreter-simulator — işlevsel olarak ipa-test-interpreter seçeneğine eşittir ancak iOS simulator için paketlenir. Yalnızca Macintosh. Bu seçeneği kullanırsanız iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini de dahil etmeniz gerekir.
- native — yerel bir masaüstü yükleyicisi. Üretilen dosya türü, komutun çalıştırıldığı işletim sisteminin yerel yükleme biçimidir:
  - EXE — Windows
  - DMG — Mac
  - DEB — Ubuntu Linux (AIR 2.6 veya öncesi)
  - RPM — Fedora veya OpenSuse Linux (AIR 2.6 veya öncesi)

  Daha fazla bilgi için bkz. "Masaüstü yerel yükleyicisini paketleme" sayfa 54.

-sampler (yalnızca iOS, AIR 3.4 ve sonraki sürümleri) iOS uygulamalarındaki telemetri tabanlı ActionScript örnekleyicisini etkinleştirir. Bu bayrağın kullanılması, uygulamanın profilini Adobe Scout olarak çıkarmanıza olanak tanır. Scout size ActionScript işlev zamanlaması, DisplayList, Stage3D görüntü oluşturma ve daha fazla olanağa yönelik derin bir bakış sunarken, herhangi bir Flash platformu içeriğinin profilini çıkarabilir. Bu bayrağın kullanılmasının performans üzerinde hafif bir etkiye sahip olabileceğini unutmayın ve bu nedenle, üretim uygulamaları için kullanmayın.

-hideAneLibSymbols (yalnızca iOS, AIR 3.4 ve sonraki sürümleri) Uygulama geliştiricileri, birden çok kaynağa ait birden çok yerel uzantıyı kullanabilir ve ANE'ler ortak bir sembol adını paylaşıyorsa, ADT "nesne dosyasında çoğaltılmış sembol" hatasını verir. Bazı durumlarda bu hata, çalışma zamanında yaşanan çökmeyle görülebilir. ANE kütüphanesinin sembollerinin yalnızca kütüphanenin kaynakları için mi (yes) yoksa global olarak (no) mı görünür yapılıp yapılmayacağını belirtmek üzere `hideAneLibSymbols` seçeneğini kullanabilirsiniz.

- **yes** — İstenmeyen sembol çakışması sorunlarını çözecek şekilde ANE sembollerini gizler.
- **no** — (Varsayılan) ANE sembollerini gizlemez. Bu, AIR 3.4 öncesinde geçerli bir davranıştır.

-embedBitcode (yalnızca iOS, AIR 25 ve üzeri) Uygulama geliştiricileri evet veya hayır seçeneğini belirterek iOS uygulamalarına bitcode katıştırıp katıştırmayacağını belirlemek üzere `embedBitcode` seçeneğini kullanabilir. Belirtilmemişse, bu anahtarın varsayılan değeri hayır şeklindedir.

DEBUGGER_CONNECTION_OPTIONS Hata ayıklayıcı bağlantı seçenekleri, bir hata ayıklama paketinin başka bir bilgisayarda çalışan uzak hata ayıklayıcıya bağlanmaya çalışması mı gerektiğini yoksa uzak hata ayıklayıcıdan bir bağlantı dinlemesi mi gerektiğini belirtir. Bu seçenekler kümesi, yalnızca mobil hata ayıklama paketleri (apk-debug ve ipa-debug öğelerini hedef alır) için desteklenir. Bu seçenekler "Hata ayıklayıcı bağlantısı seçenekleri" sayfa 180 açıklanmıştır.

- airDownloadURL Android aygıtlarında AIR çalışma zamanını indirmek ve yüklemek için alternatif bir URL belirtir. Belirtilmemişse ve AIR uygulaması çalışma zamanı zaten yüklü değilse, kullanıcıyı Android Market'taki AIR çalışma zamanına yönlendirir.

Uygulamanız alternatif bir pazar ile dağıtılıyorsa (Google tarafından yönetilen Android Market dışında bir pazar), AIR çalışma zamanını o pazardan indirmek için URL belirtmeniz gerekebilir. Bazı alternatif pazarlar, uygulamaların pazar dışından indirme gerektirmesine izin vermez. Bu seçenek yalnızca Android paketleri için desteklenir.

NATIVE_SIGNING_OPTIONS Yerel imzalama seçenekleri, yerel bir paket dosyasını imzalamak için kullanılmış sertifikayı tanımlar. Bu imzalama seçenekleri, AIR çalışma zamanı tarafından değil, yerel işletim sistemi tarafından imza uygulamak için kullanılır. Seçenekler bunun dışında AIR_SIGNING_OPTIONS ile aynıdır ve "ADT kod imzalama seçenekleri" sayfa 177 bölümünde tam olarak açıklanmıştır.

Yerel imzalar Windows ve Android'de desteklenir. Windows'ta, hem AIR imzalama seçenekleri hem de yerel imzalama seçenekleri belirtilmelidir. Android'de yalnızca yerel imzalama seçenekleri belirtilebilir.

Çoğu durumda, hem AIR imzası hem de yerel imza uygulamak için aynı kod imzalama sertifikasını kullanabilirsiniz. Ancak, bu her durumda geçerli değildir. Örneğin, Google'ın Android Market'a gönderilmiş uygulamalar için geçerli olan ilkesi, tüm uygulamaların en azından 2033 yılına kadar geçerli olan bir sertifika ile imzalanması gerektiğini bildirir. Bu, tanınmış bir sertifika yetkilisi (AIR imzası uygularken önerilir) tarafından verilen sertifikanın bir Android uygulamasını imzalamak için kullanılmaması gerektiği anlamına gelir. (Hiçbir sertifika yetkilisi bu kadar uzun bir geçerlilik süresi olan kod imzalama sertifikası sağlamaz.)

output Oluşturulacak paket dosyasının adı. Dosya uzantısının belirtilmesi isteğe bağlıdır. Belirtilmezse, -target değerine uygun bir uzantı ve geçerli işletim sistemi eklenir.

app_descriptor Uygulama tanımlayıcı dosyasının yolu. Yol, geçerli dizine göre veya mutlak yol olarak belirtilebilir. (Uygulama tanımlayıcı dosyası AIR dosyasında *application.xml* olarak yeniden adlandırılır.)

-platformsdk Hedef aygıt için platform SDK'sinin yolu:

- Android - AIR 2.6+ SDK, ilgili ADT komutlarını uygulamak için gerekli Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)
- iOS - AIR SDK, sabit bir iOS SDK ile paketlenir. -platformsdk seçeneği uygulamaları harici bir SDK ile paketlemenize olanak tanır böylelikle kullanımınız sabit iOS SDK ile sınırlı kalmaz. Örneğin, en son iOS SDK ile bir uzantı oluşturduysanız uygulamanızı paketlerken bu SDK'yi belirtebilirsiniz. Ayrıca iOS Simulator ile ADT'yi kullanırken her zaman iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini dahil etmeniz gerekir.

-arch Uygulama geliştiriciler bu argümanı kullanarak x86 platformları için APK oluşturabilir; aşağıdaki değerleri alır:

- armv7 - Android armv7 platformu için ADT paketleri APK'sı.
- x86 - Android x86 platformu için ADT paketleri APK'sı.

Bir değer belirtilmediğinde armv7 varsayılan değerdir

FILE_OPTIONS Pakete dahil edilecek uygulama dosyalarını tanımlar. Dosya seçenekleri tam olarak "Dosya ve yol seçenekleri" sayfa 179 içinde açıklanmıştır. Bir AIR veya AIRI dosyasından yerel paket oluştururken dosya seçeneklerini belirtmeyin.

input_airi Bir AIRI dosyasından yerel paket oluştururken belirtin. Hedef *air* ise (veya hedef belirtilmemişse) AIR_SIGNING_OPTIONS öğesi gerekir.

input_air Bir AIR dosyasından yerel paket oluştururken belirtin. AIR_SIGNING_OPTIONS öğesini belirtmeyin.

ANE_OPTIONS Yerel bir uzantı paketi oluşturmak için seçenekleri ve dosyaları tanımlar. Uzantı paketi seçenekleri "Yerel uzantı seçenekleri" sayfa 181 bölümünde tamamen açıklanmıştır.

ADT -paket komutu örnekleri

SWF tabanlı bir AIR uygulamasının geçerli dizinindeki belirli uygulama dosyalarını paketleyin:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

HTML tabanlı bir AIR uygulamasının geçerli dizinindeki belirli uygulama dosyalarını paketleyin:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js
image.gif
```

Geçerli çalışma dizinindeki tüm dosyaları ve alt dizinleri paketleyin:

```
adt –package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Not: *Anahtar deposu dosyası, uygulamanızı imzalamak için kullanılan özel anahtarı içerir. İmzalayıcı sertifikasını hiçbir zaman AIR paketine dahil etmeyin. ADT komutunda joker karakterler kullanırsanız, anahtar deposu dosyasını farklı bir konuma yerleştirin, böylece pakete dahil olmaz. Bu örnekte anahtar deposu dosyası cert.p12, üst dizinde bulunur.*

Yalnızca ana dosyaları ve bir görüntüler alt dizinini paketleyin:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

HTML tabanlı bir uygulamayı ve HTML, komut dosyaları ve görüntüler alt dizinlerinde bulunan tüm dosyaları paketleyin:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js
html scripts images
```

application.xml dosyasını ve bir çalışma dizininde (release/bin) bulunan ana SWF dosyasını paketleyin:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml –C
release/bin myApp.swf
```

Derleme dosya sisteminizdeki birden fazla yerden alınan varlıkları paketleyin. Bu örnekte uygulama varlıkları paketlemeden önce aşağıdaki klasörlerde konumlandırılmıştır:

```
/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
```

Aşağıdaki ADT komutunu /devRoot/myApp dizininden çalıştırma:

```
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
```

Şu paket yapısıyla sonuçlanır:

```
/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
```

ADT'yi basit bir SWF tabanlı uygulama için Java programı olarak çalıştırın (sınıf yolunu ayarlamadan):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

ADT'yi basit bir HTML tabanlı uygulama için Java programı olarak çalıştırın (sınıf yolunu ayarlamadan):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

ADT'yi bir Java programı olarak çalıştırın (Java sınıf yolu ADT.jar paketini içerecek şekilde ayarlanmış olarak):

```
java -com.adobe.air.ADT –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

ADT'yi Apache Ant'de bir Java görevi olarak çalıştırın (ancak, ADT komutunun doğrudan Ant komut dosyalarında kullanılması genellikle en iyi yöntemdir). Örnekte gösterilen yollar Windows içindir:

```
<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>
```

Not: *Bazı bilgisayar sistemlerinde, dosya sistemi yollarındaki çift baytlık karakterler yanlış yorumlanabilir. Bu durum oluşursa, ADT'yi çalıştırmak için kullanılan JRE'yi UTF-8 karakter setini kullanacak şekilde ayarlamayı deneyin. Bu, ADT'yi Mac ve Linux'ta başlatmak için kullanılan komut dosyasında varsayılan olarak gerçekleştirilir. Windows adt.bat dosyasında veya ADT'yi doğrudan Java'dan çalıştırdığınızda, Java komut satırında* `-Dfile.encoding=UTF-8` *seçeneğini belirtin.*

ADT prepare komutu

-prepare komutu imzalanmamış bir AIRI paketi oluşturur. Bir AIRI paketi tek başına kullanılamaz. Bir AIRI dosyasını imzalı bir AIR paketine dönüştürmek için -sing komutunu veya AIRI dosyasını bir yerel pakete dönüştürmek için package komutunu kullanın.

-prepare komutu aşağıdaki sözdizimini kullanır:

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output Oluşturulan AIRI dosyasının adı.

app_descriptor Uygulama tanımlayıcı dosyasının yolu. Yol, geçerli dizine göre veya mutlak yol olarak belirtilebilir. (Uygulama tanımlayıcı dosyası AIR dosyasında *application.xml* olarak yeniden adlandırılır.)

FILE_OPTIONS Pakete dahil edilecek uygulama dosyalarını tanımlar. Dosya seçenekleri tam olarak "Dosya ve yol seçenekleri" sayfa 179 içinde açıklanmıştır.

ADT sign komutu

-sign komutu AIRI ve ANE dosyalarını imzalar.

-sign komutu aşağıdaki sözdizimini kullanır:

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS AIR imzalama seçenekleri, bir paket dosyasını imzalamak için kullanılan sertifikayı tanımlar. İmzalama seçenekleri tam olarak "ADT kod imzalama seçenekleri" sayfa 177 bölümünde açıklanır.

input İmzalanacak AIRI veya ANE dosyasının adı.

output Oluşturulacak imzalı paketin adı.

Bir ANE dosyası zaten imzalanmışsa eski imza atılır. (AIR dosyaları yeniden imzalanamaz. Uygulama güncellemesi için yeni bir imza kullanmak üzere `-migrate` komutunu kullanın.)

ADT migrate komutu

-migrate komutu bir AIR dosyasına geçiş imzası uygular. Bir geçiş imzası, dijital sertifikanızı yenilediğinizde veya değiştirdiğinizde ve eski sertifika ile imzalanmış uygulamaları güncellemeniz gerektiğinde kullanılmalıdır.

AIR uygulamalarının geçiş imzası ile paketlenmesi hakkında daha fazla bilgi için bkz. "Bir AIR uygulamasının güncellenmiş sürümünü imzalama" sayfa 195.

Not: *Geçiş sertifikası sertifikanın sona erme süresinden sonra 365 gün içinde uygulanmalıdır. Bu yetkisiz kullanım süresi geçtikten sonra, uygulama güncellemeleriniz geçiş imzası ile imzalanamaz. Kullanıcılar önce uygulamanızın geçiş imzası ile imzalanmış sürümüne güncelleyebilir ve ardından en son güncellemeyi yükleyebilir veya orijinal uygulamayı kaldırabilir ve yeni AIR paketini yükleyebilirler.*

Bir geçiş imzası kullanmak için önce yeni veya yenilenmiş sertifikayı kullanarak AIR uygulamanızı imzalayın (-package veya -sing komutlarını kullanarak), ardından eski sertifikayı ve -migrate komutunu kullanarak geçiş imzasını uygulayın.

`-migrate` komutu aşağıdaki sözdizimini kullanır:

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS AIR uygulamasının mevcut sürümlerini imzalamak için kullanılmış orijinal sertifikayı tanımlayan AIR imzalama seçenekleri. İmzalama seçenekleri tam olarak "ADT kod imzalama seçenekleri" sayfa 177 bölümünde açıklanır.

input YENİ uygulama sertifikasıyla zaten imzalanmış olan AIR dosyası.

output Hem yeni hem de eski sertifikaların imzalarını taşıyan son paketin adı.

Girdi ve çıktı AIR dosyaları için kullanılan dosya adları farklı olmalıdır.

Not: *ADT migrate komutu, yerel uzantıların bulunduğu AIR masaüstü uygulamalarında kullanılamaz. Bunun nedeni, bu uygulamaların .air dosyaları olarak değil yerel yükleyiciler olarak paketlenmesidir. Yerel uzantı içeren bir AIR masaüstü uygulamasının sertifikalarını değiştirmek için uygulamayı "ADT package komutu" sayfa 164 kullanarak -migrate bayrağı ile paketleyin.*

ADT checkstore komutu

-checkstore komutu anahtar deposunun geçerliliğini kontrol etmenize izin verir. Komut aşağıdaki sözdizimini kullanır:

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS Doğrulanacak anahtar deposunu tanımlayan imzalama seçenekleri. İmzalama seçenekleri tam olarak "ADT kod imzalama seçenekleri" sayfa 177 bölümünde açıklanır.

ADT certificate komutu

-certificate komutu kendinden imzalı dijital kod imzalama sertifikası oluşturmanıza izin verir. Komut aşağıdaki sözdizimini kullanır:

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type
output password
```

-cn Yeni sertifikanın ortak adı olarak atanan dize.

-ou Sertifikayı yayımlayan birim olarak atanan dize. (İsteğe bağlı.)

-o Sertifikayı yayımlayan kurum olarak atanan dize. (İsteğe bağlı.)

-c İki harfli ISO-3166 ülke kodu. Geçersiz bir kod verilirse, sertifika oluşturulmaz. (İsteğe bağlı.)

-validityPeriod Sertifikanın geçerli olacağı yıl sayısı. Belirtilmezse beş yıllık bir geçerlilik süresi atanır. (İsteğe bağlı.)

key_type Sertifika için kullanılacak anahtar türü *2048-RSA*'dır.

output Oluşturulacak sertifika dosyası için yol veya dosya adı.

password Yeni sertifikaya erişim için şifre. AIR dosyaları bu sertifikayla imzalanırken şifre gereklidir.

ADT installApp komutu

-installApp komutu bir aygıtta veya taklitçide uygulama yükler.

Bu komutla yeniden yüklemeden önce varolan bir uygulamayı kaldırmanız gerekir.

Komut aşağıdaki sözdizimini kullanır:

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package
fileName
```

-platform Aygıtın platformunun adı. *ios* veya *android* seçeneğini belirleyin.

-platformsdk Hedef aygıt için platform SDK'sinin yolu (isteğe bağlı):

- Android - AIR 2.6+ SDK, ilgili ADT komutlarını uygulamak için gerekli Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)
- iOS - AIR SDK, sabit bir iOS SDK ile paketlenir. -platformsdk seçeneği uygulamaları harici bir SDK ile paketlemenize olanak tanır böylelikle kullanımınız sabit iOS SDK ile sınırlı kalmaz. Örneğin, en son iOS SDK ile bir uzantı oluşturduysanız uygulamanızı paketlerken bu SDK'yi belirtebilirsiniz. Ayrıca iOS Simulator ile ADT'yi kullanırken her zaman iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini dahil etmeniz gerekir.

-device*ios_simulator*, seri numarası (Android) veya bağlı aygıtın işleyicisini (iOS) belirtin. iOS üzerinde bu parametre gereklidir; Android üzerinde bu parametrenin yalnızca birden fazla Android aygıtının veya taklitçinin bilgisayarınıza eklenmesi ve çalıştırılması durumunda belirtilmesi gerekir. Belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası (Android) veya Geçersiz aygıt belirtildi (iOS). Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Not: *Bir IPA dosyasının doğrudan bir iOS aygıtına yüklenmesi, AIR 3.4 ve sonraki sürümlerinde gerçekleştirilebilir ve iTunes 10.5.0 veya sonraki sürümlerini gerektirir.*

Bağlı aygıtların işleyicisini veya seri numarasını belirlemek için `adt -devices` komutunu (AIR 3.4 ve üzeri sürümlerinde kullanılabilir) kullanın. iOS üzerinde Aygıt UUID'si yerine işleyiciyi kullanırsınız. Daha fazla bilgi için bkz. “ADT devices komutu” sayfa 176.

Ayrıca Android üzerinde, eklenen aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

-package Yüklenecek paketin dosya adı. iOS üzerinde bunun bir IPA dosyası olması gerekir. Android'de bu bir APK paketi olmalıdır. Belirtilen paket zaten yüklüyse, ADT hata kodu 14'ü döndürür: Aygıt hatası.

ADT appVersion komutu

-appVersion komutu bir aygıt veya taklitçide uygulamanın yüklü sürümünü bildirir. Komut aşağıdaki sözdizimini kullanır:

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Aygıtın platformunun adı. *ios* veya *android* seçeneğini belirleyin.

-platformsdk Hedef aygıt için platform SDK'sinin yolu:

- Android - AIR 2.6+ SDK, ilgili ADT komutlarını uygulamak için gerekli Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)
- iOS - AIR SDK, sabit bir iOS SDK ile paketlenir. -platformsdk seçeneği uygulamaları harici bir SDK ile paketlemenize olanak tanır böylelikle kullanımınız sabit iOS SDK ile sınırlı kalmaz. Örneğin, en son iOS SDK ile bir uzantı oluşturduysanız uygulamanızı paketlerken bu SDK'yi belirtebilirsiniz. Ayrıca iOS Simulator ile ADT'yi kullanırken her zaman iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini dahil etmeniz gerekir.

-device*ios_simulator* veya aygıtın seri numarasını belirtin. Yalnızca bilgisayarınıza birden fazla Android aygıt veya taklitçi eklendiğinde ve çalışırken aygıtın belirtilmesi gerekir. Belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası. Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Android'de, bağlı aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

-appid Yüklü uygulamanın AIR uygulaması kimliği. Aygıtta belirtilen kimliğe sahip herhangi bir uygulama yüklü değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası.

ADT launchApp komutu

-launchApp bir aygıtta veya taklitçide yüklü uygulamayı çalıştırır. Komut aşağıdaki sözdizimini kullanır:

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Aygıtın platformunun adı. *ios* veya *android* seçeneğini belirleyin.

-platformsdk Hedef aygıt için platform SDK'sinin yolu:

- Android - AIR 2.6+ SDK, ilgili ADT komutlarını uygulamak için gerekli Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)
- iOS - AIR SDK, sabit bir iOS SDK ile paketlenir. -platformsdk seçeneği uygulamaları harici bir SDK ile paketlemenize olanak tanır böylelikle kullanımınız sabit iOS SDK ile sınırlı kalmaz. Örneğin, en son iOS SDK ile bir uzantı oluşturduysanız uygulamanızı paketlerken bu SDK'yi belirtebilirsiniz. Ayrıca iOS Simulator ile ADT'yi kullanırken her zaman iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini dahil etmeniz gerekir.

-device*ios_simulator* veya aygıtın seri numarasını belirtin. Yalnızca bilgisayarınıza birden fazla Android aygıt veya taklitçi eklendiğinde ve çalışırken aygıtın belirtilmesi gerekir. Belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası. Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Android'de, bağlı aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

-appid Yüklü uygulamanın AIR uygulaması kimliği. Aygıtta belirtilen kimliğe sahip herhangi bir uygulama yüklü değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası.

ADT uninstallApp komutu

-uninstallApp komutu uzak bir aygıttaki veya taklitçideki yüklü uygulamayı tamamen kaldırır. Komut aşağıdaki sözdizimini kullanır:

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Aygıtın platformunun adı. *ios* veya *android* seçeneğini belirleyin.

-platformsdk Hedef aygıt için platform SDK'sinin yolu:

- Android - AIR 2.6+ SDK, ilgili ADT komutlarını uygulamak için gerekli Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)
- iOS - AIR SDK, sabit bir iOS SDK ile paketlenir. -platformsdk seçeneği uygulamaları harici bir SDK ile paketlemenize olanak tanır böylelikle kullanımınız sabit iOS SDK ile sınırlı kalmaz. Örneğin, en son iOS SDK ile bir uzantı oluşturduysanız uygulamanızı paketlerken bu SDK'yi belirtebilirsiniz. Ayrıca iOS Simulator ile ADT'yi kullanırken her zaman iOS Simulator SDK'nin yolunu belirterek -platformsdk seçeneğini dahil etmeniz gerekir.

-device*ios_simulator* veya aygıtın seri numarasını belirtin. Yalnızca bilgisayarınıza birden fazla Android aygıt veya taklitçi eklendiğinde ve çalışırken aygıtın belirtilmesi gerekir. Belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası. Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Android'de, bağlı aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

-appid Yüklü uygulamanın AIR uygulaması kimliği. Aygıtta belirtilen kimliğe sahip herhangi bir uygulama yüklü değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası.

ADT installRuntime komutu

-installRuntime komutu bir aygıtta AIR çalışma zamanını yükler.

Bu komutla yeniden yüklemeden önce varolan AIR çalışma zamanı sürümünü kaldırmanız gerekir.

Komut aşağıdaki sözdizimini kullanır:

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package
fileName
```

-platform Aygıtın platformunun adı. Şu anda bu komut yalnızca Android platformunda desteklenir. *android* adını kullanın.

-platformsdk Hedef aygıt için platform SDK'sinin yolu. Şu anda desteklenen tek platform SDK'si Android'tir. AIR 2.6+ SDK ilgili ADT komutlarını uygulamak için gerekli olan Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)

-device Aygıtın seri numarası. Yalnızca bilgisayarınıza birden fazla aygıt veya taklitçi eklendiğinde ve çalışırken aygıtın belirtilmesi gerekir. Belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası. Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Android'de, bağlı aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

-package Yüklenecek çalışma zamanının dosya adı. Android'de bu bir APK paketi olmalıdır. Paket belirtilmemişse, AIR SDK'sinde mevcut olanlardan aygıt veya taklitçi için uygun çalışma zamanı seçilir. Çalışma zamanı zaten yüklüyse ADT hata kodu 14'ü döndürür: Aygıt hatası.

ADT runtimeVersion komutu

-runtimeVersion komutu bir aygıttaki veya taklitçideki AIR çalışma zamanının yüklü sürümünü bildirir. Komut aşağıdaki sözdizimini kullanır:

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Aygıtın platformunun adı. Şu anda bu komut yalnızca Android platformunda desteklenir. *android* adını kullanın.

-platformsdk Hedef aygıt için platform SDK'sinin yolu. Şu anda desteklenen tek platform SDK'si Android'tir. AIR 2.6+ SDK ilgili ADT komutlarını uygulamak için gerekli olan Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)

-device Aygıtın seri numarası. Yalnızca bilgisayarınıza birden fazla aygıt veya taklitçi eklendiğinde ve çalışırken aygıtın belirtilmesi gerekir. Çalışma zamanı yüklü değilse veya belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası. Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Android'de, bağlı aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

ADT uninstallRuntime komutu

-uninstallRuntime komutu bir aygıttan veya taklitçiden AIR çalışma zamanını tamamen kaldırır. Komut aşağıdaki sözdizimini kullanır:

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Aygıtın platformunun adı. Şu anda bu komut yalnızca Android platformunda desteklenir. *android* adını kullanın.

-platformsdk Hedef aygıt için platform SDK'sinin yolu. Şu anda desteklenen tek platform SDK'si Android'tir. AIR 2.6+ SDK ilgili ADT komutlarını uygulamak için gerekli olan Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Ayrıca, AIR_ANDROID_SDK_HOME ortam değişkeni zaten ayarlıysa, platform SDK'si yolunun komut satırında sağlanması gerekmez. (Her ikisi de ayarlanmışsa, komut satırında sağlanan yol kullanılır.)

-device Aygıtın seri numarası. Yalnızca bilgisayarınıza birden fazla aygıt veya taklitçi eklendiğinde ve çalışırken aygıtın belirtilmesi gerekir. Belirtilen aygıt bağlı değilse ADT çıkış kodu 14'ü döndürür: Aygıt hatası. Birden fazla aygıt veya taklitçi bağlıysa ve aygıt belirtilmemişse ADT çıkış kodu 2'yi döndürür: Kullanım hatası.

Android'de, bağlı aygıtların ve çalışan taklitçilerin seri numaralarını listelemek için Android ADB aracını kullanın:

```
adb devices
```

ADT devices komutu

ADT -help komutu, o anda bağlı mobil aygıtlarının ve taklitçilerin aygıt kimliklerini gösterir:

```
adt -devices -platform iOS|android
```

-platform Kontrol edilmesi gereken platformun adı. `android` veya `iOS` belirtin.

Not: *iOS'ta bu komut, iTunes 10.5.0'ı veya sonraki sürümlerini gerektirir.*

ADT help komutu

ADT -help komutu, komut satırı seçeneklerinin kısa bir hatırlatmasını görüntüler:

```
adt -help
```

help çıktısı aşağıdaki sembolik kuralları kullanır:

- <> — açılı ayraçlar arasındaki öğeler sağlamanız gereken bilgileri gösterir.
- () — parantez içindeki öğeler help komutu çıktısında grup olarak görülen seçenekleri gösterir.
- ALL_CAPS — büyük harfle yazılan öğeler ayrı ayrı açıklanan seçenekler kümesini gösterir.
- | — VEYA. Örneğin, (A | B), A veya B öğesi anlamına gelir.
- ? — 0 veya 1. Bir öğeyi izleyen soru işareti, öğenin isteğe bağlı olduğunu ve kullanılırsa yalnızca tek bir örneğinin oluşacağını gösterir.
- * — 0 veya daha fazla. Bir öğeyi izleyen yıldız işareti, öğenin isteğe bağlı olduğunu ve herhangi bir sayıda örneğin oluşabileceğini gösterir.
- + — 1 veya daha fazla. Bir öğeyi izleyen artı işareti, öğenin gerekli olduğunu ve birden fazla örneğin oluşabileceğini gösterir.
- sembol yok — Bir öğede sonek olarak sembol yoksa, bu öğe gereklidir ve yalnızca tek bir örneği oluşabilir.

ADT seçenek kümeleri

ADT komutlarının birkaç tanesi ortak seçenek kümelerini paylaşır.

ADT kod imzalama seçenekleri

ADT, AIR uygulamalarını imzalamak için özel anahtarlara ve sertifikalara erişmek amacıyla Java Cryptography Architecture (JCA) kullanır. İmzalama seçenekleri anahtar deposunu ve bu anahtar deposu içindeki özel anahtarı ve sertifikayı tanımlar.

Anahtar deposu hem özel anahtarı hem de ilişkilendirilmiş sertifika zincirini içermelidir. İmzalama sertifikası bilgisayarda güvenilen bir sertifikaya eklenirse, sertifikanın ortak ad alanı içeriği AIR yükleme iletişim kutusunda yayıncı adı olarak görüntülenir.

ADT, sertifikanın x509v3 standardına uymasını (RFC3280) ve kod imzalama için uygun değerlere sahip Genişletilmiş Anahtar Kullanımı uzantısını içermesini gerektirir. Sertifikada tanımlanan kısıtlamalar korunur ve AIR uygulamalarını imzalamaya yönelik bazı sertifikaların kullanımını engelleyebilir.

Not: *ADT, uygun olduğunda, sertifika geri alma listelerini denetlemek ve zaman damgalarını almak üzere İnternet kaynaklarına bağlanmak için Java çalışma zamanı ortamı proxy ayarlarını kullanır. ADT kullanırken bu İnternet kaynaklarına bağlanma konusunda bir sorunla karşılaşırsanız ve ağınız belirli proxy ayarları isterse, JRE proxy ayarlarını yapılandırmanız gerekebilir.*

AIR imzalama seçenekleri sözdizimi

İmzalama seçenekleri aşağıdaki söz dizimini kullanır (tek bir komut satırında):

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias Anahtar deposundaki anahtarın başka adı. Anahtar deposu yalnızca tek bir sertifika içerdiğinde başka ad belirtmek gerekmez. Başka ad belirtilmemişse, ADT, anahtar deposundaki ilk anahtarı kullanır.

Tüm anahtar deposu yönetim uygulamaları, başka bir adın sertifikalara atanmasına izin vermez. Örneğin, Windows sistemi anahtar deposunu kullanırken, sertifikanın ayırt edilen adını başka ad olarak kullanın. Kullanılabilir sertifikaları listelemek için Java Keytool yardımcı programını kullanabilirsiniz, böylece başka adı belirleyebilirsiniz. Örneğin şu komutu çalıştırmak:

```
keytool -list -storetype Windows-MY
```

sertifika için aşağıdakine benzer bir çıktı oluşturur:

```
CN=TestingCert,OU=QE,O=Adobe,C=US, PrivateKeyEntry,
Certificate fingerprint (MD5): 73:D5:21:E9:8A:28:0A:AB:FD:1D:11:EA:BB:A7:55:88
```

ADT komut satırında bu sertifikaya başvurmak için, başka adı şu şekilde ayarlayın:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

Mac OS X'te Keychain'deki bir sertifikanın başka adı, Keychain Access uygulamasında görüntülenen addır.

-storetype Anahtar deposu uygulaması tarafından belirlenen anahtar deposu türü. Java'nın çoğu yüklemesinde bulunan varsayılan anahtar deposu uygulaması `JKS` ve `PKCS12` türlerini destekler. Java 5.0, `PKCS11` türü için donanım belirteçlerindeki anahtar depolarına erişim konusunda ve `Keychain` türü için de Mac OS X anahtarlığa erişim konusunda destek içerir. Java 6.0, `MSCAPI` türü için destek içerir (Windows'ta). Diğer JCA sağlayıcıları yüklenmişse ve yapılandırılmışsa, ek anahtar deposu türleri kullanılabilirdir. Anahtar deposu türü belirtilmemişse, varsayılan JCA sağlayıcısı için varsayılan tür kullanılır.

Saklama türü	Anahtar deposu biçimi	Minimum Java sürümü
JKS	Java anahtar deposu dosyası (.keystore)	1.2
PKCS12	PKCS12 dosyası (.p12 veya .pfx)	1.4
PKCS11	Donanım belirteci	1.5
KeychainStore	Mac OS X Keychain	1.5
Windows-MY veya Windows-ROOT	MSCAPI	1.6

-keystore Dosya tabanlı saklama türleri için anahtar deposu dosyasının yolu.

-storepass Anahtar deposuna erişmek için gereken şifre. Belirtilmemişse, ADT şifreyi ister.

-keypass AIR uygulamasını imzalamak için kullanılan özel anahtara erişim için gereken şifre. Belirtilmemişse, ADT şifreyi ister.

Not: *ADT komutunun bir parçası olarak şifre girerseniz, şifre karakterleri komut satırı geçmişinde kaydedilir. Bu nedenle, sertifikanın güvenliğinin önemli olduğu durumlarda -keypass veya -storepass seçeneklerini kullanmanız önerilmez. Ayrıca, şifre seçeneklerini atladığınızda şifre istemlerinde yazdığınız karakterlerin görüntülenmediğini unutmayın (aynı güvenlik nedenlerinden dolayı). Yalnızca şifreyi yazın ve Enter tuşuna basın.*

-providerName Belirtilen anahtar deposu türü için JCA sağlayıcısı. Belirtilmemişse, ADT, bu anahtar deposu türü için varsayılan sağlayıcıyı kullanır.

-tsa Dijital imzaya zaman damgası uygulamak için RFC3161 uyumlu zaman damgası sunucusun URL'sini belirtir. URL belirtilmemişse, Geotrust tarafından sağlanan varsayılan bir zaman damgası sunucusu kullanılır. Bir AIR uygulamasının imzası zaman damgalı olduğunda, imzalayıcı sertifikasının süresi dolduktan sonra da uygulama yüklenebilir, bunun nedeni zaman damgasının, sertifikanın imzalama sırasında geçerli olduğunu doğrulamasıdır.

ADT, zaman damgası sunucusuna bağlanamazsa, imzalama iptal edilir ve hiç paket üretilmez. Zaman damgalamayı devre dışı bırakmak için `-tsa none` öğesini belirtin. Ancak, zaman damgası olmadan paketlenmiş bir AIR uygulaması, imzalayıcı sertifikasının süresi dolduktan sonra artık yüklenebilir olmaz.

Not: *İmzalama seçeneklerinin çoğu Java Keytool yardımcı programının aynı seçeneğine eşittir. Keytool yardımcı programını, Windows'ta anahtar depolarını incelemek ve yönetmek için kullanabilirsiniz. Apple® güvenlik yardımcı programı da Mac OS X'te bu amaçla kullanılabilir.*

-provisioning-profile Apple iOS ön hazırlık dosyası. (Yalnızca iOS uygulamalarını paketlemek için gereklidir.)

İmzalama seçeneği örnekleri

.p12 dosyasıyla imzalama:

```
-storetype pkcs12 -keystore cert.p12
```

Varsayılan Java anahtar deposuyla imzalama:

```
-alias AIRcert -storetype jks
```

Belirli bir Java anahtar deposuyla imzalama:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Mac OS X anahtarlık ile imzalama:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Windows sistemi anahtar deposuyla imzalama:

```
-alias cn=AIRCert -storeype Windows-MY
```

Donanım belirteciyle imzalama (Java'yı belirteç kullanmak üzere yapılandırma konusundaki talimatlar ve doğru `providerName` değeri için belirteç üreticisinin talimatlarına başvurun):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Zaman damgası gömmeden imzalama:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Dosya ve yol seçenekleri

Dosya ve yol seçenekleri pakete dahil olan tüm dosyaları belirtir. Dosya ve yol seçenekleri aşağıdaki sözdizimini kullanır:

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs AIR dosyasında paketlenecek dosyalar ve dizinler. İstenilen sayıda dosya ve dizin boşluklarla ayrılarak belirtilebilir. Bir dizini listelerseniz, gizli dosyalar haricinde bu dizinin içinde bulunan tüm dosyalar ve alt dizinler pakete eklenir. (Buna ek olarak, uygulama tanımlayıcı dosyası doğrudan veya joker karakter ya da dizin genişletme yoluyla belirtilmişse, yoksayılır ve pakete ikinci defa eklenmez.) Belirtilen dosyalar ve dizinler geçerli dizinde veya geçerli dizinin alt dizinlerinden birinde olmalıdır. Geçerli dizini değiştirmek için `-C` seçeneğini kullanın.

Önemli: *Joker karakterler `C` seçeneğinin ardından `file_or_dir` argümanlarında kullanılamaz. (Komut kabukları joker karakterleri, argümanları ADT'ye iletmeden önce genişletir, bu da ADT'nin dosyaları yanlış konumda aramasına neden olur.) Ancak yine de geçerli dizinin yerine geçmesi için nokta karakterini, ".", kullanabilirsiniz. Örnek: `-C assets .` alt dizinler de dahil olmak üzere varlıklar dizinindeki her şeyi uygulama paketinin kök düzeyine kopyalar.*

`-C dir files_and_dirs` Uygulama paketine eklenen sonraki dosyaları ve dizinleri işlemeden önce çalışma dizinini *dir* değeri ile değiştirir (**`files_and_dirs`** içinde belirtilir). Dosyalar veya dizinler uygulama paketinin köküne eklenir. Dosya sistemindeki birden çok noktadan dosyaları dahil etmek için `–C` seçeneği istenilen sayıda kullanılabilir. `dir` için göreceli bir yol belirtilmişse, yol her zaman orijinal çalışma dizininden çözümlenir.

ADT paketteki dosyaları ve dizinleri işledikçe, geçerli dizin ve hedef dosyalar arasındaki göreceli yollar saklanır. Paket yüklendiğinde, bu yollar uygulama dizini yapısına genişletilir. Bu nedenle, `-C release/bin lib/feature.swf` dosyasını belirtmek, `release/bin/lib/feature.swf` dosyasını kök uygulama klasörünün `lib` alt dizinine yerleştirir.

`-e file_or_dir dir` Dosyayı veya dizini belirtilen paket dizininin içine yerleştirir. Bu seçenek bir ANE dosyasını paketlerken kullanılamaz.

Not: *Uygulama tanımlayıcı dosyasının `<content>` öğesi, ana uygulama dosyasının uygulama paketi dizin ağacındaki son konumunu belirtmelidir.*

-extdir *dir* *`dir`* değeri yerel uzantıları (ANE dosyaları) arayacak dizinin adıdır. Mutlak bir yol veya geçerli dizine bağlı bir yol belirtin. `-extdir` seçeneğini bir kereden fazla belirtebilirsiniz.

Belirtilen dizin uygulamanın kullandığı yerel uzantılar için ANE dosyalarını içerir. Bu dizindeki her bir ANE dosyası .ane dosya uzantısına sahiptir. Ancak, .ane dosya adı uzantısının önündeki dosya adının uygulama tanımlayıcı dosyasının `extensionID` öğesinin değeriyle eşleşmesi *gerekmez*.

Örneğin, `-extdir ./extensions` parametresini kullanırsanız `extensions` dizini aşağıdaki gibi görünebilir:

```
extensions/
    extension1.ane
    extension2.ane
```

Not: *-extdir seçeneğinin kullanımı ADT aracı ve ADL aracı için farklıdır. ADL'de, seçenek her biri paketlenmemiş bir ANE dosyasını içeren alt dizinlere sahip bir dizini belirtir. ADT'de, seçenekler ANE dosyalarını içeren bir dizini belirtir.*

Hata ayıklayıcı bağlantısı seçenekleri

Paketin hedefi apk-debug, ipa-debug veya ipa-debug-interpreter olduğunda uygulamanın uzaktan bir hata ayıklayıcıya bağlanmayı (normalde wifi hata ayıklama için kullanılır) veya uzaktan bir hata ayıklayıcıdan gelen bağlantıyı dinlemeyi (normalde USB hata ayıklama için kullanılır) deneyip denemeyeceğini belirtmek için bağlantı seçenekleri kullanılabilir. Bir hata ayıklayıcıya bağlanmak için `-connect` seçeneğini kullanın. Bir USB bağlantısı üzerinden hata ayıklayıcıdan bağlantı kabul etmek için `-listen` seçeneğini kullanın. Bu seçenekler birbirini geçersiz kılar; başka bir deyişle, bunları bir arada kullanamazsınız.

-connect seçeneği aşağıdaki sözdizimini kullanır:

```
-connect hostString
```

-connect Varsa, uygulama uzak hata ayıklayıcısına bağlanmaya çalışacaktır.

hostString Flash hata ayıklama aracı FDB'yi çalıştıran bilgisayarı tanımlayan bir dize. Belirtilmezse, uygulama paketin oluşturulduğu bilgisayarda çalışan bir hata ayıklayıcıya bağlanmaya çalışacaktır. Ana bilgisayar dizesi *machinename.subgroup.example.com* gibi tam nitelikli bir bilgisayar etki alanı adı veya *192.168.4.122* gibi bir IP adresi olabilir. Belirtilen (veya varsayılan) makine bulunamıyorsa, çalışma zamanı geçerli ana bilgisayar adını talep eden bir iletişim kutusu görüntüler.

`-listen` seçeneği aşağıdaki sözdizimini kullanır:

```
-listen port
```

-listen Varsa, uzak hata ayıklayıcısından bir bağlantı bekler.

port (İsteğe bağlı) Dinlenecek bağlantı noktası. Varsayılan olarak, çalışma zamanı 7936 bağlantı noktasını dinler. `-listen` seçeneğini kullanmayla ilgili daha fazla bilgi için bkz. "USB üzerinden FDB ile uzaktan hata ayıklama" sayfa 104.

Android uygulama profili oluşturma seçenekleri

Paketin hedefi apk-profile olduğunda, profil oluşturucu seçenekleri performans ve bellek profili oluşturma işlemleri için hangi önceden yüklenmiş SWF dosyasının kullanılacağını belirtmek üzere kullanılabilir. Profil oluşturucu seçenekleri aşağıdaki sözdizimini kullanır:

```
-preloadSWFPath directory
```

-preloadSWFPath Varsa, uygulama belirtilen dizinde önyükleme SWF'sini bulmaya çalışır. Belirtilmezse, ADT, AIR SDK'sindeki önyükleme SWF dosyasını dahil eder.

directory Profil oluşturucu önyükleme SWF dosyasını içeren dizin.

Yerel uzantı seçenekleri

Yerel uzantı seçenekleri, yerel bir uzantı için bir ANE dosyasını paketlemeye yönelik seçenekleri ve dosyaları belirtir. Bu seçenekleri `-target` seçeneğinin `ane` olduğu bir ADT paketi komutuyla kullanın.

```
extension-descriptor -swc swcPath
   -platform platformName
  -platformoptions path/platform.xml
   FILE_OPTIONS
```

extension-descriptor Yerel uzantının tanımlayıcı dosyası.

-swc Yerel uzantı için ActionScript kodunu ve kaynakları içeren SWC dosyası.

-platform Bu ANE dosyasının desteklediği platformun adı. Her birinde kendi `FILE_OPTIONS` öğesi bulunan birden çok `-platform` seçeneğini dahil edebilirsiniz.

-platformoptions Bir platform seçenekleri (platform.xml) dosyasına giden yol. Varsayılan olmayan bağlayıcı seçeneklerini, paylaşılan kütüphaneleri ve uzantı tarafından kullanılan üçüncü taraf statik kütüphaneleri belirtmek için bu dosyayı kullanın. Daha fazla bilgi almak ve örnekleri görmek için iOS yerel kütüphanelerine bakın.

FILE_OPTIONS Pakete eklenecek yerel platform dosyalarını (örneğin yerel uzantı paketine eklenecek statik kütüphaneler) belirler. Dosya seçenekleri tam olarak "Dosya ve yol seçenekleri" sayfa 179 içinde açıklanmıştır. (-e seçeneğinin bir ANE dosyası paketlerken kullanılamayacağını unutmayın.)

Daha fazla Yardım konusu

Yerel bir uzantıyı paketleme

ADT hata mesajları

Aşağıdaki tablolar ADT programı tarafından bildirilebilecek olası hataları ve bunların olası nedenlerini listeler:

Uygulama tanımlayıcı doğrulama hataları

Hata kodu	Açıklama	Notlar
100	Uygulama tanımlayıcı ayrıştırılamıyor	Uygulama tanımlayıcı dosyasındaki kapatılmamış etiketler gibi XML sözdizimi hatalarını kontrol edin.
101	Ad alanı eksik	Eksik ad alanını ekleyin.
102	Geçersiz ad alanı	Ad alanı yazımını kontrol edin.
103	Beklenmeyen öğe veya nitelik	Sorunlu öğeleri ve nitelikleri kaldırın. Tanımlayıcı dosyada özel değerler geçerli değil. Öğe ve nitelik adlarının yazımını kontrol edin. Öğelerin doğru üst öğede konumlandırıldığından ve niteliklerin doğru öğelerle kullanıldığından emin olun.
104	Eksik öğe veya nitelik	Gerekli öğe veya niteliği ekleyin.
105	Öğe veya nitelik geçersiz değer içeriyor	Sorunlu değeri düzeltin.

Hata kodu	Açıklama	Notlar
106	Geçersiz pencere nitelik bileşimi	`transparency = true` ve `systemChrome = standard` gibi bazı pencere ayarları birlikte kullanılamaz. Uyumsuz ayarlardan birini değiştirin.
107	Minimum pencere boyutu maksimum pencere boyutundan büyük	Minimum veya maksimum boyut ayarını değiştirin.
108	Nitelik önceki öğede zaten kullanılmış	
109	Yinelenen öğe.	Yinelenen öğeyi kaldırın.
110	Belirtilen türde en az bir öğe gerekir.	Eksik öğeyi ekleyin.
111	Uygulama tanımlayıcısında listelenen profillerden hiçbiri yerel uzantıları desteklemiyor.	supportedProfies listesine yerel uzantıları destekleyen bir profil ekleyin.
112	AIR hedefi yerel uzantıları desteklemez.	Yerel uzantıları destekleyen bir hedef seçin.
113	<nativeLibrary> ve <initializer> birlikte sağlanmalıdır.	Yerel uzantıdaki her yerel kütüphane için bir başlatıcı işlevi belirtilmelidir.
114	<nativeLibrary> öğesi olmadan <finalizer> öğesi bulundu.	Platform yerel kütüphane kullanmıyorsa bir sonlandırıcı belirtmeyin.
115	Varsayılan platform bir yerel uygulama içermemelidir.	Varsayılan platform öğesinde yerel kütüphane belirtmeyin.
116	Tarayıcı başlatma bu hedef için desteklenmiyor.	`<allowBrowserInvocation>` öğesi belirtilen paketleme hedefi için `true` olamaz.
117	Bu hedef, yerel uzantıları paketlemek için en azından n ad alanını gerektirir.	Uygulama tanımlayıcısındaki AIR ad alanını desteklenen bir değer olarak değiştirin.

Ad alanları, öğeler, nitelikler ve bunların geçerli değerleri hakkında bilgi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201.

Uygulama simgesi hataları

Hata kodu	Açıklama	Notlar
200	Simge dosyası açılamıyor	Dosyanın belirtilen yolda mevcut olup olmadığını kontrol edin. Dosyanın açılabildiğinden emin olmak için başka bir uygulama kullanın.
201	Simge yanlış boyutta	Simge boyutu (piksel cinsinden) XML etiketiyle eşleşmeli. Örneğin, uygulama tanımlayıcı öğesi verildiğinde: `<image32x32>icon.png</image32x32>` `icon.png` içindeki görüntü tam olarak 32x32 piksel olmalı.
202	Simge dosyası, desteklenmeyen bir görüntü biçimi içeriyor.	Yalnızca PNG biçimi desteklenir. Uygulamanızı paketlemeden önce diğer biçimlerdeki görüntüleri dönüştürün.

Uygulama dosyası hataları

Hata kodu	Açıklama	Notlar
300	Eksik dosya veya dosya açılamıyor	Komut satırında belirtilen bir dosya bulunamadı veya açılamıyor.
301	Uygulama tanımlayıcı dosyası eksik veya açılamıyor	Uygulama tanımlayıcı dosyası belirtilen yolda bulunamadı veya açılamıyor.
302	Pakette kök içerik dosyası eksik	Uygulama tanımlayıcının `<content>` öğesinde referans verilen SWF veya HTML dosyası, ADT komut satırında listelenen dosyalara dahil edilerek pakete eklenmeli.
303	Pakette simge dosyası eksik	Uygulama tanımlayıcıda belirtilen simge dosyaları, ADT komut satırında listelenen dosyalar arasına dahil edilerek pakete eklenmeli. Simge dosyaları otomatik olarak eklenmez.
304	Başlangıç penceresi içeriği geçersiz	Uygulama tanımlayıcının `<content>` öğesinde referans verilen dosya geçerli bir HTML veya SWF dosyası olarak tanınmadı.
305	Başlangıç penceresi içeriği SWF sürümü, ad alanı sürümünü aşıyor	Uygulama tanımlayıcının `<content>` öğesinde referans verilen dosyanın SWF sürümü, tanımlayıcı ad alanında belirtilen AIR sürümü tarafından desteklenmiyor. Örneğin, bir SWF10 (Flash Player 10) dosyasını bir AIR 1.1 uygulamasının başlangıç içeriği olarak paketlemeye çalışmak bu hatayı yaratacaktır.
306	Profil desteklenmiyor.	Uygulama tanımlayıcısı dosyasının içinde belirttiğiniz profil desteklenmiyor. Bkz. "supportedProfiles" sayfa 236.
307	Ad alanı en az *nnn* olmalıdır.	Uygulamada kullanılan özellikler için uygun ad alanını kullanın (2.0 ad alanı gibi).

Diğer hatalar için çıkış kodları

Çıkış kodu	Açıklama	Notlar
2	Kullanım hatası	Komut satırı argümanlarında hatalar olup olmadığını kontrol eder.
5	Bilinmeyen hata	Bu hata, yaygın hata koşullarıyla açıklanamayan bir durumu gösterir. Olası kök nedenler ADT ve Java Çalışma Zamanı Ortamı arasındaki uyumsuzluğu, bozuk ADT veya JRE yüklemelerini ve ADT içindeki programlama hatalarını içerir.
6	Çıktı dizinine yazılamadı	Belirtilen (veya uygulanan) çıktı dizininin erişilebilir olduğundan ve bunu içeren sürücünün yeterli disk alanına sahip olduğundan emin olun.

Çıkış kodu	Açıklama	Notlar
7	Sertifikaya erişilemedi	Anahtar deposuna giden yolun doğru biçimde belirtildiğinden emin olun. Anahtar deposu içindeki sertifikanın erişilebilir olup olmadığını kontrol edin. Java 1.6 Keytool yardımcı programı, sorun giderme sertifikası erişim sorunlarına yardımcı olmak üzere kullanılabilir.
8	Geçersiz sertifika	Sertifika dosyası yanlış biçimlendirilmiş, değiştirilmiş, süresi dolmuş veya iptal edilmiş.
9	AIR dosyası imzalanamadı	ADT'ye aktarılan imzalama seçeneklerini doğrulayın.
10	Zaman damgası oluşturulamadı	ADT, zaman damgası sunucusuna bağlantı oluşturamadı. İnternet'e bir proxy sunucusu aracılığıyla bağlandığınızda, JRE proxy ayarlarını yapılandırmanız gerekebilir.
11	Sertifika oluşturma hatası	İmza oluşturmak için kullanılan komut satırı argümanlarını doğrulayın.
12	Geçersiz giriş	Komut satırında ADT'ye aktarılan diğer argümanları ve dosya yollarını doğrulayın.
13	Eksik aygıt SDK'si	SDK konfigürasyonunu doğrulayın. ADT belirtilen komutu yürütmek için gerekli olan aygıt SDK'sini bulamadı.
14	Aygıt hatası	ADT aygıt sınırlaması veya sorunu nedeniyle komutu yürütemiyor. Örneğin, bu çıkış kodu gerçekten yüklü olmayan bir uygulama kaldırılmaya çalışıldığında atılır.
15	Aygıt yok	Bir aygıtın takılı ve açık olduğunu ya da taklitçinin çalıştığını doğrulayın.
16	Eksik GPL bileşenleri	Geçerli AIR SDK'si talep işlemini gerçekleştirmek için gereken tüm bileşenleri içermiyor.
17	Aygıt paketleme aracı başarısız oldu.	Beklenen işletim sistemi bileşenleri eksik olduğundan paket oluşturulamadı.

Android hataları

Çıkış kodu	Açıklama	Notlar
400	Geçerli Android sdk sürümü niteliği desteklemiyor.	Nitelik adının doğru yazılıp yazılmadığını ve görüldüğü öğe için geçerli bir nitelik olup olmadığını denetleyin. Nitelik Android 2.2'den sonra çıkmışsa ADT komutunda -platformsdk bayrağını ayarlamanız gerekebilir.
401	Geçerli Android sdk sürümü nitelik değerini desteklemiyor	Nitelik değerinin doğru yazılıp yazılmadığını ve nitelik için geçerli bir değer olup olmadığını denetleyin. Nitelik değeri Android 2.2'den sonra çıkmışsa ADT komutunda -platformsdk bayrağını ayarlamanız gerekebilir.
402	Geçerli Android sdk sürümü XML etiketini desteklemiyor	XML etiketi adının doğru yazılıp yazılmadığını ve geçerli bir Android bildirim belgesi öğesi olup olmadığını denetleyin. Öğe Android 2.2'den sonra çıkmışsa ADT komutunda -platformsdk bayrağını ayarlamanız gerekebilir.
403	Android etiketinin geçersiz kılınmasına izin verilmez	Uygulama AIR kullanımı için ayrılmış bir Android bildirim belgesi öğesini geçersiz kılmaya çalışıyor. Bkz. "Android ayarları" sayfa 73.
404	Android niteliğinin geçersiz kılınmasına izin verilmez	Uygulama AIR kullanımı için ayrılmış bir Android bildirim belgesi niteliğini geçersiz kılmaya çalışıyor. Bkz. "Android ayarları" sayfa 73.
405	Android etiketi %1 manifestAdditions etiketindeki ilk öğe olmalıdır	Belirtilen etiketi gerekli konuma taşıyın.
406	Android etiketi %2 öğesinin %1 niteliği içinde %3 geçersiz değeri bulunuyor.	Nitelik için geçerli bir değer sağlayın.

ADT ortam değişkenleri

ADT aşağıdaki ortam değişkenlerinin değerlerini okur (ayarlanmışlarsa):

AIR_ANDROID_SDK_HOME, Android SDK'sinin kök dizininin (araçlar klasörünü içeren dizin) yolunu belirtir. AIR 2.6+ SDK ilgili ADT komutlarını uygulamak için gerekli olan Android SDK araçlarını içerir. Bu değeri yalnızca Android SDK'nin farklı bir sürümünü kullanmak için ayarlayın. Bu değişken ayarlıysa, -platformsdk seçeneğini gerektiren ADT komutları çalışırken -platformsdk seçeneğinin belirtilmesi gerekmez. Hem bu değişken hem de komut satırı seçeneği ayarlıysa komut satırında belirtilen yol kullanılır.

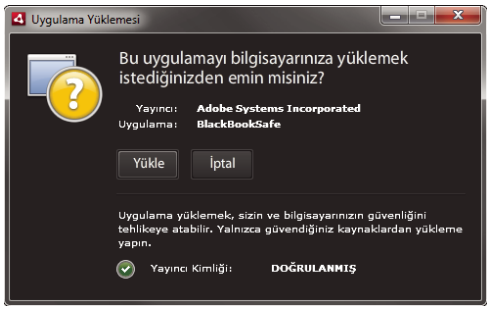
AIR_EXTENSION_PATH, bir uygulamanın gerektirdiği yerel uzantıların aranacağı dizinlerin bir listesini belirtir. Bu dizinler listesi tüm yerel uzantı dizinleri ADT komut satırında belirtildikten sonra sıralı olarak aranır. ADL komutu bu ortam değişkenini de kullanabilir.

Not: *Bazı bilgisayar sistemlerinde, bu ortam değişkenlerinde saklanan dosya sistemi yollarındaki çift baytlık karakterler yanlış yorumlanabilir. Bu durum oluşursa, ADT'yi çalıştırmak için kullanılan JRE'yi UTF-8 karakter setini kullanacak şekilde ayarlamayı deneyin. Bu, ADT'yi Mac ve Linux'ta başlatmak için kullanılan komut dosyasında varsayılan olarak gerçekleştirilir. Windows adt.bat dosyasında veya ADT'yi doğrudan Java'dan çalıştırdığınızda, Java komut satırında* `-Dfile.encoding=UTF-8` *seçeneğini belirtin.*

Bölüm 13: AIR uygulamalarını imzalama

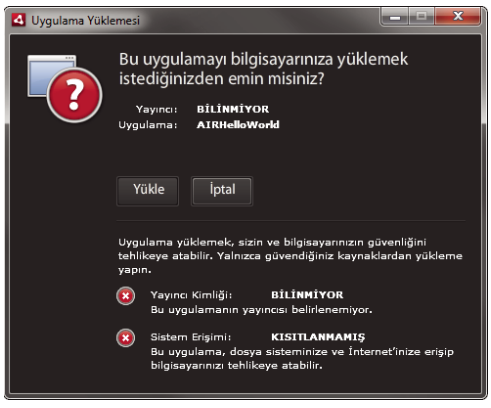
AIR dosyasını dijital olarak imzalama

AIR yükleme dosyalarınızı tanınan bir sertifika yetkilisi (CA) tarafından yayımlanan bir sertifikayla dijital olarak imzalamak, kullanıcılarınıza, yükledikleri uygulamanın yanlışlıkla veya kötü amaçla değiştirilmediğine dair bir güvence verir ve sizi imzalayan (yayıncı) olarak tanımlar. AIR uygulaması güvenilen veya yükleme bilgisayarında güvenilen bir sertifikaya *zincirle* bağlanan bir sertifikayla imzalandığında, AIR, yükleme sırasında yayıncı adını görüntüler.



Güvenilir bir sertifikayla imzalanan uygulama için yükleme onayı iletişim kutusu

Uygulamayı kendinden imzalı bir sertifika (veya güvenilir bir sertifikaya zincirle bağlanmayan bir sertifika) ile imzalarsanız, kullanıcı uygulamanızı yükleyerek daha yüksek bir güvenlik riski almayı kabul etmelidir. Yükleme iletişim kutuları bu ek riski yansıtır:



Kendinden imzalı bir uygulama için yükleme onayı iletişim kutusu

***Önemli:** Kötü amaçlı bir varlık, imzalama anahtar deposu dosyanızı bir şekilde ele geçirirse veya özel anahtarınızı bulursa, AIR dosyasında sizin kimliğinizi kullanarak sahtecilik yapabilir.*

Kod imzalama sertifikaları

Kod imzalayıcı sertifikaların kullanımına ilişkin güvenlik teminatları, sınırlamalar ve yasal zorunluluklar Sertifika Uygulama Bildirimleri'nde (CPS) ve sertifikayı yayımlayan yetkili tarafından yayınlanan abone sözleşmelerinde anlatılır. Şu anda AIR kod imzalayıcı sertifikalarını yayınlayan sertifika yetkililerine ilişkin anlaşmalar hakkında daha fazla bilgi için bkz.:

ChosenSecurity (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

ChosenSecurity CPS (http://www.chosensecurity.com/resource_center/repository.htm)

GlobalSign (http://www.globalsign.com/code-signing/index.html)

GlobalSign CPS (http://www.globalsign.com/repository/index.htm)

Thawte CPS (http://www.thawte.com/cps/index.html)

VeriSign CPS (http://www.verisign.com/repository/CPS/)

VeriSign Subscriber's Agreement (https://www.verisign.com/repository/subscriber/SUBAGR.html)

AIR kod imzalama hakkında

AIR dosyası imzalandığında, yükleme dosyasına bir dijital imza dahil edilir. İmza, AIR dosyasının imzalandıktan sonra değiştirilmediğini doğrulamak için kullanılan bir paket özetini ve yayıncı kimliğini doğrulamak için kullanılan imzalayıcı sertifikayla ilgili bilgileri içerir.

AIR, bir sertifikaya güvenilip güvenilmeyeceğini belirlemek için işletim sisteminin sertifika deposuyla desteklenen ortak anahtar altyapısını (PKI) kullanır. Yayıncı bilgilerinin doğrulanması için, AIR uygulamasının yüklendiği bilgisayar AIR uygulamasını imzalamak için kullanılan sertifikaya doğrudan güvenmeli veya sertifikayı güvenilen bir sertifika yetkilisine bağlayan bir sertifika zincirine güvenmelidir.

AIR dosyası güvenilen kök sertifikalarından birine zincirle bağlı olmayan bir sertifikayla (normalde bu tüm kendinden imzalı sertifikaları da içerir) imzalanırsa, yayıncı bilgileri doğrulanamaz. AIR, AIR paketinin imzalandıktan sonra değiştirilmediğini belirleyebilse de, dosyayı gerçekten kimin oluşturduğunu ve imzaladığını bilmenin bir yolu yoktur.

Not: *Kullanıcı kendinden imzalı bir sertifikaya güvenmeyi seçebilir, bundan sonra sertifikayla imzalanmış olan tüm AIR uygulamaları sertifikada yayıncı adı olarak ortak ad alanı değerini görüntüler. AIR, kullanıcının bir sertifikayı güvenilir olarak belirlemesi için herhangi bir yöntem sağlamaz. Sertifika (özel anahtar içermeyen) kullanıcıya ayrı olarak sağlanmalı ve kullanıcı sertifikayı sistem sertifika deposundaki uygun konuma aktarmak için işletim sistemi tarafından sağlanan mekanizmalardan birini veya uygun bir aracı kullanmalıdır.*

AIR yayıncı kimlikleri hakkında

Önemli: *AIR 1.5.3'ten itibaren, yayıncı kimliği uygun bulunmaz ve artık kod imzalama sertifikasına dayanarak hesaplanmaz. Yeni uygulamalar bir yayıncı kimliği gerektirmez ve kullanmamalıdır. Mevcut uygulamaları güncellerken, uygulama açıklayıcısı dosyasında orijinal yayıncı kimliğini belirtmeniz gerekir.*

AIR 1.5.3'ten önce, AIR uygulama yükleyicisi bir AIR dosyasının yüklenmesi sırasında bir yayıncı kimliği oluştururdu. Bu, AIR dosyasını imzalamada kullanılan sertifika için benzersiz olan bir kimlikti. Aynı sertifikayı birden fazla AIR uygulaması için yeniden kullandıysanız, uygulamalar aynı yayıncı kimliğini almıştır. Bir uygulama güncellemesini farklı bir sertifika ile ve bazen orijinal sertifikanın yenilenmiş bir örneğiyle bile imzalamak yayıncı kimliğini değiştiriyordu.

AIR 1.5.3 ve sonrasında, AIR tarafından bir yayıncı kimliği atanmaz. AIR 1.5.3 ile yayınlanan bir uygulama, uygulama açıklayıcısında bir yayıncı kimliği dizesi belirtebilir. Yayıncı kimliğini yalnızca ilk olarak 1.5.3'ten önceki AIR sürümleri için yayınlanmış olan uygulamalara ilişkin güncellemeler yayınlarken belirtmeniz gerekir. Uygulama açıklayıcısında orijinal kimliği belirtmezseniz, yeni AIR paketi mevcut uygulamanın bir güncellemesi olarak kabul edilmez.

Orijinal yayıncı kimliğini belirlemek için, META-INF/AIR alt dizininde orijinal uygulamanın yüklü olduğu `publisherid` dosyasını bulun. Bu dosyanın içindeki dize yayıncı kimliğidir. Uygulama açıklayıcınız, yayıncı kimliğinin elle belirtmek için uygulama açıklayıcısı dosyasının ad alanı bildiriminde AIR 1.5.3 çalışma zamanını (veya sonrasını) belirtmelidir.

Yayıncı kimliği mevcut olduğunda, aşağıdaki amaçlar için kullanılır:

- Şifrelenmiş yerel veri deposuna ilişkin şifreleme anahtarının bir parçası olarak
- Uygulama depolama dizinine ilişkin yolun bir parçası olarak
- Yerel bağlantılara ilişkin bağlantı dizesinin bir parçası olarak
- Bir uygulamanın çağrılması için AIR tarayıcı içi API'si ile kullanılan kimlik dizesinin bir parçası olarak
- OSID'nin bir parçası olarak (özel yükleme/kaldırma programları oluşturulurken kullanılır)

Bir yayıncı kimliği değiştiğinde, kimliğe bağlı herhangi bir AIR özelliğinin davranışı da değişir. Örneğin, mevcut şifrelenmiş yerel veri deposundaki verilere artık erişilemez ve uygulamaya yerel bağlantı oluşturan Flash veya AIR örnekleri, bağlantı dizesindeki yeni kimliği kullanmalıdır. Yüklü bir uygulamanın yayıncı kimliği AIR 1.5.3 veya üst sürümlerinde değişemez. Bir AIR paketi yayınlarken farklı bir yayıncı kimliği kullanırsanız, yükleyici yeni paketi bir güncelleme yerine farklı bir uygulama olarak kabul eder.

Sertifika biçimleri hakkında

AIR imzalama araçları, Java Şifreleme Mimarisi (JCA) ile erişilebilen tüm anahtar depolarını kabul eder. Bunlar arasında PKCS12 biçimli dosyalar (genellikle .pfx veya .p12 dosya uzantısı kullanan), Java .keystore dosyaları, PKCS11 donanım anahtar depoları ve sistem anahtar depoları gibi dosya tabanlı anahtar depoları bulunmaktadır. ADT'nin erişebildiği anahtar deposu biçimleri, ADT'yi çalıştırmak için kullanılan Java çalışma zamanının sürümüne ve yapılandırmasına bağlıdır. PKCS11 donanım belirteçleri gibi bazı anahtar deposu türlerine erişmek, ek yazılım sürücülerinin ve JCA eklentilerinin yüklenmesini ve yapılandırılmasını gerektirebilir.

AIR dosyalarını imzalamak için en yaygın bulunan kod imzalayıcı sertifikaları kullanabilir veya özellikle AIR uygulamalarını imzalamak için yayınlanan yeni bir sertifika edinebilirsiniz. Örneğin aşağıdaki VeriSign, Thawte, GlobalSign veya ChosenSecurity sertifika türlerinden herhangi biri kullanılabilir:

- ChosenSecurity
  - Adobe AIR için TC Yayıncı Kimliği
- GlobalSign
  - ObjectSign Kod İmzalayıcı Sertifikası
- Thawte:
  - AIR Geliştirici Sertifikası
  - Apple Geliştirici Sertifikası
  - JavaSoft Geliştirici Sertifikası
  - Microsoft Authenticode Sertifikası

- VeriSign:
  - Adobe AIR Dijital Kimliği
  - Microsoft Authenticode Digital ID
  - Sun Java Signing Digital ID

Not: *Sertifika kod imzalama için oluşturulmuş olmalıdır. AIR dosyalarını imzalamak için SSL veya başka türde bir sertifika kullanamazsınız.*

Zaman damgaları

Bir AIR dosyasını imzaladığınızda paketleme aracı, zaman damgası yetkilisinin sunucusunu, bağımsız doğrulanabilir bir imzalama tarihi ve saati almak için sorgular. Alınan zaman damgası AIR dosyasına gömülüdür. İmzalayıcı sertifika imzalama sırasında geçerli olduğu sürece, sertifikanın süresi dolduktan sonra bile AIR dosyası yüklenebilir. Diğer yandan, zaman damgası alınamazsa, sertifikanın süresi dolduğunda veya sertifika iptal edildiğinde AIR dosyası artık yüklenemez hale gelir.

AIR paketleme araçları varsayılan olarak bir zaman damgası alır. Ancak zaman damgası hizmeti kullanılamadığında uygulamaların paketlenmesini sağlamak için, zaman damgalamayı devre dışı bırakabilirsiniz. Adobe, genel olarak dağıtılan tüm AIR dosyalarının bir zaman damgası içermesini önerir.

AIR paketleme araçları tarafından kullanılan varsayılan zaman damgası yetkilisi Geotrust'tır.

Sertifika alma

Bir sertifika almak için normalde sertifika yetkilisinin web sitesini ziyaret edersiniz ve şirketin tedarik işlemini tamamlarsınız. AIR araçlarının ihtiyacı olan anahtar deposu dosyasını oluşturmak için kullanılan araçlar, satın alınan sertifikanın türüne, sertifikanın alıcı bilgisayarda nasıl saklandığına ve bazı durumlarda sertifikayı almak için kullanılan tarayıcının türüne bağlıdır. Örneğin, Thawte'den bir Adobe Geliştirici sertifikasını almak ve dışa aktarmak için Mozilla Firefox kullanmalısınız. Bu şekilde sertifika doğrudan Firefox kullanıcı arabiriminden .p12 veya .pfx dosyası olarak dışa aktarılabilir.

Not: *1.5 ve üstü Java sürümleri, PKCS12 sertifika dosyalarını korumak için kullanılan şifrelerde yüksek ASCII karakterlerini kabul etmez. Java, AIR geliştirici araçları tarafından imzalanmış AIR paketlerini oluşturmak için kullanılır. Sertifikayı .p12 veya .pfx dosyası olarak dışa aktardığınızda, şifrede yalnızca normal ASCII karakterlerini kullanın.*

AIR kurulum dosyalarını paketlemek için, Air Geliştirici Aracı'nı (ADT) kullanarak kendinden imzalı bir sertifika oluşturabilirsiniz. Bazı üçüncü taraf araçlar da kullanılabilir.

Kendinden imzalı bir sertifikanın nasıl oluşturulacağı ve AIR dosyasını imzalama konusunda talimatlar için bkz. "AIR Geliştirici Aracı (ADT)" sayfa 163. Ayrıca AIR dosyalarını Flash Builder, Dreamweaver ve Flash için AIR güncelleme kullanarak da dışa aktarabilir ve imzalayabilirsiniz.

Aşağıdaki örnek, Thawte Sertifika Yetkilisi'nden AIR Geliştirici Sertifikası'nın nasıl alınacağını ve sertifikanın ADT'yle kullanıma nasıl hazırlanacağını anlatır.

Örnek: Thawte'den AIR Geliştirici Sertifikası alma

Not: *Bu örnek kod imzalayıcı sertifikasını alma ve kullanıma hazırlama yollarından yalnızca birini göstermektedir. Her sertifika yetkilisinin kendi politika ve yordamları vardır.*

Thawte web sitesi, AIR Geliştirici Sertifikası'nı satın almak için Mozilla Firefox tarayıcı kullanmanızı gerekli kılar. Sertifikanın özel anahtarı, tarayıcının anahtar deposunda saklanmaktadır. Firefox anahtar deposunun ana şifreyle korunduğundan ve bilgisayarın da fiziksel koşullar bakımından güvencede olduğundan emin olun. (Tedarik işlemi tamamlandıktan sonra, sertifika anahtarını ve özel anahtarı tarayıcı anahtar deposundan dışa aktarabilir ve kaldırabilirsiniz.)

Sertifika kayıt işleminin bir parçası olarak, bir özel/ortak anahtar çifti oluşturulur. Özel anahtar otomatik olarak Firefox anahtar deposunda saklanır. Sertifikayı Thawte'nin web sitesinden istemek ve almak için aynı bilgisayarı ve tarayıcıyı kullanmalısınız.

1. Thawte web sitesini ziyaret edin ve Kod İmzalayıcı Sertifikalar için Ürün sayfası (Product page for Code Signing Certificates) bölümüne gidin.
2. Kod İmzalayıcı Sertifikalar listesinden, Adobe AIR Geliştirici Sertifikası'nı seçin.
3. Üç adımlı kayıt işlemini tamamlayın. Kuruluş ve irtibat bilgileri vermeniz gerekir. Daha sonra Thawte kimlik doğrulama işlemini gerçekleştirir ve ek bilgiler isteyebilir. Doğrulama tamamlandıktan sonra Thawte size sertifikayı almanız için gereken talimatların bulunduğu bir e-posta yollar.

   Not: *Gerekli belgelerin türü hakkında ek bilgi şu bağlarda bulunabilir: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.*
4. Thawte sitesinden yayımlanan sertifikayı alın. Sertifika otomatik olarak Firefox anahtar deposuna kaydedilir.
5. Özel anahtarı ve sertifikayı içeren bir anahtar deposu dosyasını Firefox anahtar deposundan aşağıdaki adımları izleyerek dışa aktarın:

   Not: *Özel anahtarı/sertifikayı Firefox'tan dışarı aktarırken, ADT, Flex, Flash ve Dreamweaver'ın kullanabileceği .p12 (pfx) biçiminde dışa aktarılır.*

   a. Firefox *Sertifika Yöneticisi* iletişim kutusunu açın:
   b. Windows'ta: Araçlar -> Seçenekler -> Gelişmiş -> Şifreleme -> Sertifikaları Görüntüle öğelerini açın
   c. Mac OS'de: Firefox -> Tercihler -> Gelişmiş -> Şifreleme -> Sertifikaları Görüntüle öğelerini açın
   d. Linux'ta: Düzenle -> Tercihler -> Gelişmiş -> Şifreleme -> Sertifikaları Görüntüle öğelerini açın
   e. Sertifika listesinden Adobe AIR Kod İmzalayıcı Sertifikası'nı seçin ve **Yedekle** düğmesini tıklatın.
   f. Anahtar deposu dosyasının dışa aktarılacağı bir dosya adı ve konum girin ve **Kaydet**'i tıklatın.
   g. Firefox ana şifresini kullanıyorsanız dosyayı dışa aktarırken, yazılım güvenlik aygıtı için şifrenizi girmeniz istenir. (Bu şifre yalnızca Firefox tarafından kullanılır.)
   h. *Sertifika Yedekleme Şifresi Seç* iletişim kutusunda, anahtar deposu dosyası için bir şifre oluşturun.

      Önemli: *Bu şifre anahtar deposu dosyasını korur ve dosya, AIR uygulamalarını imzalamada kullanılırken gereklidir. Güvenli bir şifre seçilmelidir.*
   i. Tamam'ı tıklatın. Başarılı bir yedekleme şifresi mesajı almanız gerekir. Özel anahtarı ve sertifikayı içeren anahtar deposu dosyası .p12 dosya uzantısıyla (PKCS12 biçiminde) kaydedilir.
6. Dışa aktarılan anahtar deposu dosyasını ADT, Flash Builder, Flash Professional veya Dreamweaver ile kullanın. Dosya için oluşturulan şifre, bir AIR uygulaması imzalanacağı zaman gereklidir.

Önemli: *Özel anahtar ve sertifika, Firefox anahtar deposunda saklanmaya devam eder. Bu sayede sertifika dosyasının ek bir kopyasını dışa aktarabilirsiniz, ancak sertifikanızın ve özel anahtarınızın güvenliğinin sağlanması için korunması gereken bir başka erişim noktası da sağlanmış olur.*

Sertifikaları değiştirme

Bazı durumlarda, AIR uygulamanızın güncellemelerini imzalamak için kullandığınız sertifikayı değiştirmeniz gerekir. Bu durumlar şunlardır:

- Orijinal imzalama sertifikasını yenileme.
- Kendinden imzalı bir sertifikadan bir sertifika yetkilisi tarafından yayımlanan bir sertifikaya yükseltme yapılırken
- Süresi dolmak üzere olan kendinden imzalı bir sertifika, başka bir sertifikayla değiştirilirken
- Bir ticari sertifika diğeriyle değiştirilirken, örneğin şirketinizin kimliği değiştiğinde

AIR'nin bir AIR dosyasını bir güncelleme olarak tanıması için, ya hem orijinal hem de güncelleme olan AIR dosyalarını aynı sertifika ile imzalamanız ya da güncellemeye bir sertifika geçiş imzası uygulamanız gerekir. Geçiş imzası, güncel AIR paketine orijinal sertifika kullanılarak uygulanan ikinci bir imzadır. Geçiş imzası, imzalayanın uygulamanın orijinal yayıncısı olduğunu belirtmek için orijinal sertifikayı kullanır.

Geçiş imzalı bir AIR dosyası yüklendikten sonra, yeni sertifika birincil sertifika olur. Sonraki güncellemeler geçiş imzası gerektirmez. Ancak, güncellemeleri atlayan kullanıcılarla uyum sağlamak için mümkün olduğu kadar uzun süre boyunca geçiş imzaları uygulamanız gerekir.

Önemli: *Sertifikayı değiştirmeli ve süresi bitmeden önce güncellemeye orijinal sertifikayla bir geçiş imzası uygulamalısınız. Aksi takdirde, kullanıcıların yeni sürümü yüklemeden önce uygulamanızın varolan sürümünü kaldırmaları gerekir. AIR 1.5.3 veya üstü için 365 günlük yetkisiz kullanım süresi dahilinde süresi dolmuş bir sertifikayı kullanarak geçiş imzası uygulayabilirsiniz (süresi dolduktan sonra). Ancak, ana uygulama imzasını uygulamak için süresi sona ermiş sertifikayı kullanamazsınız.*

Sertifikaları değiştirmek için:

1. Uygulamanız için bir güncelleme oluşturun
2. Güncelleme AIR dosyasını **yeni** sertifikayla paketleyin ve imzalayın
3. AIR dosyasını **orijinal** sertifikayla (ADT `-migrate` komutunu kullanarak) yeniden imzalayın

Geçiş imzasına sahip bir AIR dosyası diğer yönlerden normal bir AIR dosyasıdır. Uygulama bir sisteme orijinal sürümü olmadan yüklenirse, AIR yeni sürümü normal bir şekilde yükler.

Not: *AIR 1.5.3'ten önce, bir AIR uygulamasının yenilenmiş bir sertifikayla imzalanması her zaman bir geçiş imzası gerektirmiyordu. AIR 1.5.3'ten itibaren, yenilenen sertifikalar için her zaman bir geçiş imzası gerekir.*

Geçiş imzası uygulamak için "ADT migrate komutu" sayfa 171"Bir AIR uygulamasının güncellenmiş sürümünü imzalama" sayfa 195 bölümünde açıklandığı şekilde kullanın.

Not: *ADT migrate komutu, yerel uzantıların bulunduğu AIR masaüstü uygulamalarında kullanılamaz. Bunun nedeni, bu uygulamaların .air dosyaları olarak değil yerel yükleyiciler olarak paketlenmesidir. Yerel uzantı içeren bir AIR masaüstü uygulamasının sertifikalarını değiştirmek için uygulamayı "ADT package komutu" sayfa 164 kullanarak -migrate bayrağı ile paketleyin.*

Uygulama kimlik değişiklikleri

AIR 1.5.3'ten önce, bir AIR uygulamasının kimliği geçiş imzası ile imzalanan bir güncelleme yüklendiğinde değişiyordu. Bir uygulamanın kimliğini değiştirmenin bazı tepkileri vardır, bunlara aşağıdakiler dahildir:

- Yeni uygulama sürümü, var olan şifreli yerel depodaki verilere erişemez.
- Uygulama depolama dizininin konumu değişir. Eski konumdaki veriler yeni dizine kopyalanmaz. (Ancak yeni uygulama, eski yayıncı kimliğine bağlı olarak orijinal dizini bulabilir.)
- Uygulama artık eski yayıncı kimliğini kullanarak yerel bağlantıları açamaz.

- Web sayfasından bir uygulamaya erişmeye yarayan kimlik dizesi değişir.
- Uygulamanın OSID'si değişir. (OSID, özel yükleme/kaldırma programları yazılırken kullanılır.)

AIR 1.5.3 veya üstü ile bir güncelleme yayınlanırken, uygulama kimliği değişemez. Orijinal uygulama ve yayıncı kimlikleri güncel AIR dosyasının uygulama açıklayıcısında belirtilmelidir. Aksi halde, yeni paket bir güncelleme olarak kabul edilmez.

Not: *AIR 1.5.3 veya sonraki bir sürümü ile yeni bir AIR uygulaması yayınlarken, bir yayıncı kimliği belirtmemelisiniz.*

Terminoloji

Bu bölümde, uygulamanızı genel dağıtım için nasıl imzalamanız gerektiğine yönelik kararlar alırken anlamanız gereken bazı anahtar sözcükleri kapsayan bir terimler sözlüğü bulunmaktadır.

Terim	**Açıklama**
Sertifika Yetkilisi (CA)	Güvenilen bir üçüncü taraf olarak hizmet veren ve sonunda bir ortak anahtar sahibinin kimliğini onaylayan ortak anahtar altyapı ağındaki bir varlık. Normalde bir CA, sertifika sahibinin kimliğini doğruladığını bildirmek için kendi özel anahtarıyla imzalanan dijital sertifikalar yayımlar.
Sertifika Uygulama Bildirimi (CPS)	Sertifika yetkilisinin sertifika yayımlama ve doğrulamaya ilişkin uygulamalarını ve politikalarını ortaya koyar. CPS, CA ile aboneleri ve bağlı olduğu taraflar arasındaki anlaşmanın bir parçasıdır. Ayrıca kimlik doğrulama politikalarını ve sağladıkları sertifikaların sunduğu güvence düzeyleri de anlatır.
Sertifika İptal Listesi (CRL)	İptal edilen ve artık bağlı olunmaması gereken yayımlanmış sertifikaların bir listesi. AIR, bir AIR uygulaması imzalandığında ve mevcut zaman damgası yoksa yine uygulama yüklendiğinde CRL'yi kontrol eder.
Sertifika zinciri	Bir sertifika zinciri, zincirdeki her sertifikanın bir sonraki sertifika tarafından imzalandığı bir sertifika sırası.
Dijital Sertifika	Sahibin kimliği, sahibin ortak anahtarı ve sertifikanın kendi kimliği hakkında bilgi içeren dijital bir belge. Bir sertifika yetkilisi tarafından yayımlanan sertifika da, yayımlayan CA'ya ait bir sertifika tarafından imzalanır.
Dijital İmza	Yalnızca ortak-özel anahtar çiftinin yarısı olan ortak anahtarla şifresi çözülebilen şifreli bir mesaj veya özet. Bir PKI'de dijital imza, sonunda sertifika yetkilisine kadar izlenebilen bir veya daha fazla dijital sertifika içerir. Kişinin yayımlayan sertifika yetkilisine, imzalayanın kimliğine güvendiğini varsayarak, dijital imza bir mesajın (veya bilgisayar dosyasının) imzalandığından beri değiştirilmediğini doğrulamak için kullanılabilir (kullanılan şifreleme algoritmasının sağladığı güvence sınırları dahilinde).
Anahtar Deposu	Dijital sertifikaları ve bazı durumlarda ilgili özel anahtarları içeren veri tabanı.
Java Şifreleme Mimarisi (JCA)	Anahtar depolarını yönetmek ve bunlara erişmek için kullanılan genişletilebilir bir mimari. Daha fazla bilgi için bkz. Java Şifreleme Mimarisi Başvuru Kılavuzu (Java Cryptography Architecture Reference Guide).
PKCS #11	RSA Laboratuvarları'nın Şifreleme Belirteci Arabirim Standardı. Donanım belirteci tabanlı anahtar deposu.
PKCS #12	RSA Laboratuvarları'nın Kişisel Bilgi Alışverişi Sözdizimi Standardı. Genellikle bir özel anahtarı ve ilişkilendirilmiş dijital sertifikasını içeren dosya tabanlı anahtar deposu.
Özel Anahtar	İki parçalı ortak-özel anahtar asimetrik şifreleme sisteminin özel olan yarısı. Özel anahtar gizli tutulmalıdır ve hiçbir zaman bir ağ üzerinden iletilmemelidir. Dijital olarak imzalanan mesajlar, imzalayan tarafından özel anahtarla şifrelenir.
Ortak Anahtar	İki parçalı ortak-özel anahtar asimetrik şifreleme sisteminin ortak olan yarısı. Ortak anahtar kullanıma açıktır ve özel anahtarla şifrelenen mesajların şifresini çözmek için kullanılır.

Terim	Açıklama
Ortak Anahtar Altyapısı (PKI)	Sertifika yetkililerinin ortak anahtar sahiplerinin kimliğini doğruladığı güven sistemi. Ağın istemcileri, dijital bir mesajı (veya dosyayı) imzalayanın kimliğini doğrulamak için güvenilen bir CA tarafından yayımlanan dijital sertifikalara güvenir.
Zaman damgası	Bir olayın gerçekleştiği tarihi ve saati içeren dijital olarak imzalanmış veri. ADT, zaman damgasını AIR paketindeki RFC 3161 uyumlu zaman sunucusundan dahil edebilir. AIR mevcut olduğunda zaman damgasını, sertifikanın imzalama sırasındaki geçerliliğini belirlemek için kullanır. Bu sayede imzalayıcı sertifikasının süresi dolduktan sonra da AIR uygulaması yüklenebilir.
Zaman damgası yetkilisi	Zaman damgalarını yayımlayan yetkili. Bir zaman damgasının AIR tarafından tanınması için, damganın RFC 3161 ile uyumlu olması ve zaman damgası imzasının yükleme bilgisayarındaki güvenilen kök sertifikaya zincirle bağlı olması gerekir.

iOS Sertifikaları

Apple tarafından verilen kod imzalama sertifikaları Adobe AIR ile geliştirilenler de dahil olmak üzere iOS uygulamalarını imzalamak için kullanılır. Test aygıtlarına bir uygulama yüklemek için Apple geliştirme sertifikasını kullanarak imza uygulamak gerekir. Bitmiş uygulamayı dağıtmak için dağıtım sertifikasını kullanarak imza uygulamak gerekir.

Bir uygulamayı imzalamak için ADT hem kod imzalama sertifikasına hem de ilgili özel anahtara erişim gerektirir. Sertifika dosyası özel anahtarı içermez. Hem sertifikayı hem de özel anahtarı içeren, Kişisel Bilgi Değişimi dosyası (.p12 veya .pfx) biçiminde bir anahtar deposu oluşturmanız gerekir. Bkz. “Bir geliştirici sertifikasını P12 anahtar deposu dosyasına dönüştürme” sayfa 194.

Bir sertifika imzalama isteği oluşturma

Geliştirici sertifikası almak için, Apple iOS Provisioning Portal sitesine gönderilen bir sertifika imzalama isteği oluşturursunuz.

Sertifika imzalama isteği işlemi bir genel-özel anahtar çifti üretir. Özel anahtar bilgisayarınızda kalır. Genel anahtarı ve tanımlayıcı bilgilerinizi içeren imzalama isteğini Sertifika Yetkilisi rolündeki Apple'a gönderirsiniz. Apple sertifikanızı kendi World Wide Developer Relations sertifikasıyla imzalar.

Mac OS'de bir sertifika imzalama isteği oluşturma

Mac OS'de, bir kod imzalama isteği oluşturmak için Keychain Access uygulamasını kullanabilirsiniz. Keychain Access uygulaması, Applications (Uygulamalar) dizininin Utilities (Yardımcı Programlar) adlı alt dizinindedir. Sertifika imzalama isteği oluşturma talimatları Apple iOS Provisioning Portal sitesinde bulunabilir.

Windows'ta bir sertifika imzalama isteği oluşturma

Windows geliştiricileri için iPhone geliştirici sertifikasını bir Mac bilgisayarında edinmek en kolay yol olabilir. Ancak, bir Windows bilgisayarında da sertifika almak mümkündür. İlk olarak, OpenSSL kullanarak bir sertifika imzalama talebi (CSR dosyası) oluşturursunuz:

1. OpenSSL'i Windows bilgisayarınıza yükleyin. (http://www.openssl.org/related/binaries.html adresine gidin.)

   Ayrıca Open SSL indirme sayfasında listelenen Visual C++ 2008 Yeniden Dağıtılabilir dosyalarını da yüklemeniz gerekebilir. (Visual C++ uygulamasının bilgisayarınızda yüklü olması *gerekmez*.)

2. OpenSSL sepet dizinine bir Windows komut oturumu ve CD açın (c:\OpenSSL\bin\ gibi).

3. Komut satırına şunları girerek özel anahtarı oluşturun:

   ```
   openssl genrsa -out mykey.key 2048
   ```

   Bu özel anahtar dosyasını kaydedin. Daha sonra kullanılacaktır.

OpenSSL'yi açarken hata mesajlarını göz ardı etmeyin. OpenSSL bir hata mesajı oluşturursa, hala dosya çıktısı oluşturabilir. Ancak, bu dosyalar kullanılabilir durumda olmayabilir. Hata görürseniz, söz diziminizi kontrol edin ve komutu tekrardan çalıştırın.

4 Komut satırına şunları girerek CSR dosyasını oluşturun:

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest  -subj
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

E-posta adresini, CN (sertifika adı) ve C (ülke) değerlerini kendinizinkilerle değiştirin.

5 CSR dosyasını adresindeki Apple iPhone geliştiricisi sitesine yükleyin. (Bkz "iPhone geliştirici sertifikası için başvurma ve bir temel hazırlık profili oluşturma".)

Bir geliştirici sertifikasını P12 anahtar deposu dosyasına dönüştürme

P12 anahtar deposu oluşturmak için Apple geliştirici sertifikanızı ve ilgili özel anahtarı tek bir dosyada birleştirmelisiniz. Keystore dosyasını oluşturma işlemi, orijinal sertifikayı imzalama isteğini oluşturmak için kullandığınız yönteme ve özel anahtarın nerede saklandığına bağlıdır.

iPhone geliştirici sertifikasını Mac OS'de P12 dosyasına dönüştürme

Apple iPhone sertifikasını indirdiğinizde, P12 anahtar deposu biçimine dışa aktarın. Mac® OS'ta bunu yapmak için:

1 Keychain Access uygulamasını açın (Applications/Utilities (Uygulamalar/Yardımcı Programlar) klasöründe)

2 Anahtarlığa sertifikayı eklemediyseniz, File (Dosya) > Import (İçe Aktar) öğesini seçin. Daha sonra Apple'dan aldığınız sertifika dosyasına (.cer dosyası) gidin.

3 Daha sonra Keychain Access'te Keys (Anahtarlar) kategorisini seçin.

4 iPhone Geliştirme Sertifikasıyla ilgili özel anahtarı seçin.

Özel anahtar iPhone Geliştiricisi tarafından belirlenir: <Adı> <Soyadı> onunla eşleştirilen genel sertifika.

5 iPhone Geliştiricisi sertifikasını command tuşuna basarken tıklatın ve *"iPhone Geliştiricisi: Ad..." öğesini dışa aktar* öğesini seçin.

6 Anahtar deponuzu Kişisel Bilgi Değişimi (.p12) dosya biçiminde kaydedin.

7 Uygulamaları imzalamak için anahtar deposunu kullanırken veya bu anahtar deposundaki anahtarı veya sertifikayı başka bir anahtar deposuna aktarırken kullanılacak bir şifre oluşturmanız istenir.

Apple geliştirici sertifikasını Windows'ta P12 dosyasına dönüştürme

AIR for iOS uygulamaları geliştirmek için bir P12 sertifika dosyası kullanmanız gerekir. Bu sertifikayı Apple'dan aldığınız Apple iPhone geliştirici sertifika dosyasına göre oluşturursunuz.

1 Apple'dan aldığınız geliştirici sertifika dosyasını bir PEM sertifika dosyasına dönüştürün. OpenSSL bölme dizininden aşağıdaki komut satırı ifadesini çalıştırın:

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```

2 Bir Mac bilgisayarındaki anahtarlıktan özel anahtar kullanıyorsanız, bunu bir PEM anahtarına dönüştürün:

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```

3 Artık anahtara ve iPhone geliştirici sertifikasının PEM sürümüne göre geçerli bir P12 dosyası oluşturabilirsiniz:

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

Mac OS anahtarlığından bir anahtar kullanıyorsanız, bir önceki adımda oluşturduğunuz PEM sürümünü kullanın. Aksi takdirde, daha önce oluşturduğunuz (Windows'ta) OpenSSL anahtarını kullanın.

ADT ile imzalanmamış bir AIR ara dosyası oluşturma

İmzalanmamış AIR ara dosyası oluşturmak için `-prepare` komutunu kullanın. Geçerli bir AIR yükleme dosyası üretmek için, AIR ara dosyası ADT `-sign` komutuyla imzalanmalıdır.

`-prepare` komutu, `-package` komutuyla aynı bayrakları ve parametreleri alır (imzalama seçenekleri dışında). Aradaki tek fark çıktı dosyasının imzalanmamasıdır. Ara dosya, dosya adı uzantısıyla oluşturulur: `airi`.

AIR ara dosyasını imzalamak için, ADT `-sign` komutunu kullanın. (Bkz. "ADT prepare komutu" sayfa 170.)

ADT -komut hazırlama örneği

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

AIR ara dosyasını ADT ile imzalama

AIR ara dosyasını ADT ile imzalamak için, `-sign` komutunu kullanın. sign komutu yalnızca AIR ara dosyalarıyla çalışır (`airi` uzantısı). AIR dosyası ikinci kez imzalanamaz.

AIR ara dosyası oluşturmak için, adt `-prepare` komutunu kullanın. (Bkz. "ADT prepare komutu" sayfa 170.)

AIRI dosyasını imzalama

❖ ADT `-sign` komutunu aşağıdaki sözdizimiyle kullanın:

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS İmzalama seçenekleri, AIR dosyasının imzalanacağı özel anahtarı ve sertifikayı tanımlar. Bu seçenekler "ADT kod imzalama seçenekleri" sayfa 177 açıklanmaktadır.

airi_file İmzalanacak olan imzalanmamış AIR ara dosyasının yolu.

air_file Oluşturulacak AIR dosyasının adı.

ADT -komut imzalama örneği

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Daha fazla bilgi için bkz. "ADT sign komutu" sayfa 170.

Bir AIR uygulamasının güncellenmiş sürümünü imzalama

Varolan bir AIR uygulamasının güncel sürümünü her oluşturduğunuzda güncel uygulamayı imzalarsınız. En iyi durumda, güncel sürümü imzalamak için, önceki sürümü imzalamak üzere kullandığınız sertifikanın aynısını kullanabilirsiniz. Bu durumda imzalama, uygulamanın ilk defa imzalanması ile tamamen aynıdır.

Uygulamanın önceki sürümünün imzalanması için kullanılan sertifikanın süresi dolduysa ve sertifika yenilenmiş ya da değiştirilmişse, güncel sürümü imzalamak için yenilenmiş veya yeni (değiştirilmiş) sertifikayı kullanabilirsiniz. Bunu yapmak için, uygulamayı yeni sertifikayla imzalar *ve* orijinal sertifikayı kullanarak bir geçiş imzası uygularsınız. Geçiş sertifikası, orijinal sertifikanın sahibinin güncellemeyi yayınladığını doğrular.

Geçiş imzası uygulamadan önce aşağıdaki noktaları göz önünde bulundurun:

- Bir geçiş imzası uygulamak için, orijinal sertifikanın hala geçerli olması veya süresinin son 365 gün içinde dolmuş olması gerekir. Bu dönem 'yetkisiz kullanım süresi' olarak adlandırılır ve süresi gelecekte değişebilir.

  Not: *AIR 2.6 sürümünden önce, yetkisiz kullanım süresi 180 gündü.*

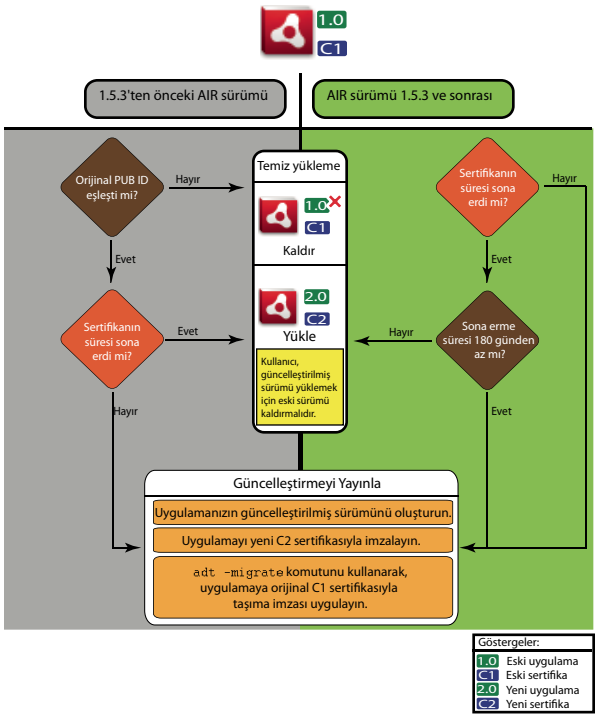
- Sertifikanın süresi dolduktan sonra ve 365 günlük yetkisiz kullanım süresi bittikten sonra geçiş imzası uygulayamazsınız. Bu durumda kullanıcıların güncel sürümü yüklemeden önce varolan sürümü kaldırması gerekir.
- 365 günlük yetkisiz kullanım süresi yalnızca uygulama tanımlayıcısı ad alanında AIR 1.5.3 sürümü veya daha sonrasını belirten uygulamalar için geçerlidir.

Önemli: *Güncellemeleri süresi geçmiş sertifikaların geçiş imzaları ile imzalamak geçici bir çözümdür. Kapsamlı bir çözüm için, uygulama güncellemelerinin dağıtımını yönetmek üzere standartlaştırılmış bir imza iş akışı oluşturun. Örneğin, her bir güncellemeyi en yeni sertifikayla imzalayın ve önceki güncellemeyi (mevcut ise) imzalamak için kullanılan sertifikayı kullanarak bir geçiş sertifikası uygulayın. Her bir güncellemeyi, kullanıcıların uygulamayı indirebileceği kendi URL'sine yükleyin. Daha fazla bilgi için bkz. "Uygulama güncellemeleri için iş akışını imzalama" sayfa 256.*

Aşağıdaki tabloda ve şekilde geçiş imzalarının iş akışı özetlenmektedir:

Senaryo	**Orijinal Sertifikanın Durumu**	**Geliştirici Eylemi**	**Kullanıcı Eylemi**
Adobe AIR çalışma zamanı sürümü 1.5.3 veya üstüne bağlı uygulama	Geçerli	AIR uygulamasının en son sürümünü yayınlayın	Eylem gerekmiyor Uygulama otomatik olarak yükseltir
	Süresi dolmuş, ancak 365 günlük yetkisiz kullanım süresi içinde	Uygulamayı yeni sertifikayla imzalayın. Süresi dolmuş sertifikayı kullanarak bir geçiş imzası uygulayın.	Eylem gerekmiyor Uygulama otomatik olarak yükseltir
	Süresi doldu ve yetkisiz kullanım sürümünde değil	Geçiş imzasını AIR uygulama güncellemesine uygulayamazsınız. Bunun yerine, yeni bir sertifika kullanarak AIR uygulamasının başka bir sürümünü yayınlamanız gerekir. Kullanıcılar mevcut AIR uygulaması sürümünü kaldırdıktan sonra yeni sürümü yükleyebilirler.	AIR uygulamasının geçerli sürümünü kaldırın ve en son sürümünü yükleyin

Senaryo	Orijinal Sertifikanın Durumu	Geliştirici Eylemi	Kullanıcı Eylemi
• Adobe AIR çalışma zamanı sürümü 1.5.2 veya daha eski bir sürüme bağlı uygulama • Güncellemenin uygulama tanımlayıcısındaki yayıncı kimliği önceki sürümün yayıncı kimliği ile eşleşiyor	Geçerli	AIR uygulamasının en son sürümünü yayınlayın	Eylem gerekmiyor Uygulama otomatik olarak yükseltir
	Süresi doldu ve yetkisiz kullanım sürümünde değil	Geçiş imzasını AIR uygulama güncellemesine uygulayamazsınız. Bunun yerine, yeni bir sertifika kullanarak AIR uygulamasının başka bir sürümünü yayınlamanız gerekir. Kullanıcılar mevcut AIR uygulaması sürümünü kaldırdıktan sonra yeni sürümü yükleyebilirler.	AIR uygulamasının geçerli sürümünü kaldırın ve en son sürümünü yükleyin
• Adobe AIR çalışma zamanı sürümü 1.5.2 veya daha eski bir sürüme bağlı uygulama • Güncellemenin uygulama tanımlayıcısındaki yayıncı kimliği önceki sürümün yayıncı kimliği ile **eşleşmiyor**	Herhangi bir	Geçerli bir sertifika kullanarak AIR uygulamasını imzalama ve AIR uygulamasının en son sürümünü yayınlama	AIR uygulamasının geçerli sürümünü kaldırın ve en son sürümünü yükleyin



Güncellemeler için iş akışını imzalama

Bir AIR uygulamasını yeni bir sertifika kullanmaya geçirme

Bir AIR uygulamasını, uygulamanın yüklenmesi sırasında yeni bir sertifika kullanmaya geçirmek için:

1 Uygulamanız için bir güncelleme oluşturun

2 Güncelleme AIR dosyasını **yeni** sertifikayla paketleyin ve imzalayın

3 AIR dosyasını `-migrate` komutunu kullanarak **orijinal** sertifikayla yeniden imzalayın

`-migrate` komutuyla imzalanmış bir AIR dosyası, eski sertifikayla imzalanan önceki sürümlerin güncellenmesi için kullanılmasının yanı sıra uygulamanın yeni sürümünün yüklenmesi amacıyla da kullanılabilir.

Not: *1.5.3'ten daha önceki bir AIR sürümü için yayınlanan bir uygulamayı güncellerken, uygulama tanımlayıcısında orijinal yayıncı kimliğini belirtin. Aksi halde, uygulamanızın kullanıcılarının güncellemeyi yüklemeden önce önceki sürümü kaldırması gerekir.*

ADT `-migrate` komutunu aşağıdaki sözdizimiyle kullanın:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** İmzalama seçenekleri, AIR dosyasının imzalanacağı özel anahtarı ve sertifikayı tanımlar. Bu seçenekler **orijinal** imzalayıcı sertifikayı tanımlamalıdır ve "ADT kod imzalama seçenekleri" sayfa 177 açıklanmaktadır.
- **air_file_in Yeni** sertifikayla imzalanan, güncelleme için AIR dosyası.
- **air_file_out** Oluşturulacak AIR dosyası.

Not: *Girdi ve çıktı AIR dosyaları için kullanılan dosya adları farklı olmalıdır.*

Aşağıdaki örnekte, bir geçiş imzasının güncel bir AIR uygulaması sürümüne uygulanması için `-migrate` bayrağı ile ADT'nin çağrılması gösterilmektedir:

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Not: *`-migrate` komutu ADT'ye AIR 1.1 sürümünde eklenmiştir.*

Bir yerel yükleyici AIR uygulamasını yeni bir sertifika kullanmaya geçirme

Yerel yükleyici olarak yayınlanan bir AIR uygulaması (örneğin, yerel uzantı api'si kullanan bir uygulama) ADT `-migrate` komutunun kullanılmasıyla imzalanamaz. Bunun nedeni, bir .air dosyası değil, platforma özel bir yerel uygulama olmasıdır. Bunun yerine, yerel uzantı olarak yayınlanan bir AIR uygulamasını yeni bir sertifika kullanmaya geçirmek için:

1. Uygulamanız için bir güncelleme oluşturun.
2. Uygulama tanımlayıcı (app.xml) dosyanızda `<supportedProfiles>` etiketinin hem masaüstü profilini hem de extendedDesktop profilini içerdiğinden emin olun (veya `<supportedProfiles>` etiketini uygulama tanımlayıcısından kaldırın).
3. Güncelleme uygulamasını **yeni** sertifikayla ADT `-package` komutunu kullanarak **bir .air dosyası şeklinde** paketleyin ve imzalayın.
4. Geçiş sertifikasını ADT `-migrate` komutunu kullanarak **Orijinal** sertifikayla .air dosyasına uygulayın (daha önce "Bir AIR uygulamasını yeni bir sertifika kullanmaya geçirme" sayfa 198 bölümünde açıklandığı şekilde).
5. .air dosyasını ADT `-package` komutunu kullanarak `-target native` bayrağıyla yerel bir yükleyici içinde paketleyin. Uygulama zaten imzalanmış olduğundan, bu adımın bir parçası olarak bir imzalama sertifikası belirtmezsiniz.

Aşağıdaki örnekte bu işleme ilişkin 3-5 arasındaki adımlar gösterilmektedir. Kod, bir AIR uygulamasının güncel sürümünü yerel bir yükleyici olarak paketlemek için ADT'yi `-package` komutuyla çağırır, ADT'yi `-migrate` komutuyla çağırır, ardından ADT'yi `-package` komutuyla tekrar çağırır:

```
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
```

Yerel uzantı kullanan bir AIR uygulamasını yeni bir sertifika kullanmaya geçirme

Yerel uzantı kullanan bir AIR uygulaması ADT `-migrate` komutunun kullanılmasıyla imzalanamaz. Ayrıca, bir ara .air dosyası olarak yayınlanamadığından yerel yükleyici AIR uygulaması geçirme işlemi kullanılarak geçirilemez. Bunun yerine, yerel uzantı kullanan bir AIR uygulamasını yeni bir sertifika kullanmaya geçirmek için:

1. Uygulamanız için bir güncelleme oluşturun
2. ADT `-package` komutunu kullanarak güncel yerel yükleyiciyi paketleyin ve imzalayın. Uygulamayı **yeni** sertifikayla paketleyin ve **orijinal** sertifikayı belirterek `-migrate` bayrağını ekleyin.

`-migrate` bayrağıyla ADT `-package` komutunu çağırmak için aşağıdaki sözdizimini kullanın:

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** İmzalama seçenekleri, AIR dosyasının imzalanacağı özel anahtarı ve sertifikayı tanımlar. Bu seçenekler **yeni** imzalama sertifikasını tanımlar ve "ADT kod imzalama seçenekleri" sayfa 177 açıklanmaktadır.
- **MIGRATION_SIGNING_OPTIONS** İmzalama seçenekleri, AIR dosyasının imzalanacağı özel anahtarı ve sertifikayı tanımlar. Bu seçenekler **orijinal** imzalama sertifikasını tanımlar ve "ADT kod imzalama seçenekleri" sayfa 177 açıklanmaktadır.

- Diğer seçenekler, yerel yükleyici AIR uygulamasını paketlemek için kullanılan seçeneklerle aynıdır ve "ADT package komutu" sayfa 164 açıklanmaktadır.

Aşağıdaki örnekte, yerel uzantı kullanan güncel bir AIR uygulaması sürümünü paketlemek üzere `-package` komutu ve `-migrate` bayrağıyla ADT'yi çağırma gösterilmektedir:

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore
original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Not: *`-package` komutunun `-migrate` bayrağı AIR 3.6 ve üst sürümlerinde ADT'de kullanılabilir.*

ADT ile kendinden imzalı bir sertifika oluşturma

Geçerli bir AIR kurulum dosyası oluşturmak için kendiliğinden imzalı sertifikalar kullanabilirsiniz. Ancak, kendiliğinden imzalı sertifikalar, kullanıcılarınıza yalnızca sınırlı bir güvenlik sağlar. Kendiliğinden imzalı sertifikaların orijinalliği doğrulanamaz. Kendinden imzalı bir AIR dosyası yüklendiğinde, yayıncı bilgileri kullanıcıya Bilinmeyen olarak görüntülenir. ADT tarafından oluşturulan sertifika beş yıl boyunca geçerlidir.

Kendinden oluşturulan bir sertifikayla imzalanan AIR uygulaması için bir güncelleme oluşturursanız, hem orijinal hem de güncelleme AIR dosyalarını imzalamak için aynı sertifikayı kullanmanız gerekir. ADT'nin ürettiği sertifikalar, aynı parametreler kullanılsa bile her zaman benzersizdir. Bu nedenle, güncellemelerin ADT tarafından oluşturulan bir sertifikayla kendinden imzalanmasını istiyorsanız, orijinal sertifikayı güvenli bir konumda saklayın. Buna ek olarak, ADT tarafından oluşturulan orijinal sertifikanın süresi dolduktan sonra, güncellenmiş bir AIR dosyası üretemezsiniz. (Yeni uygulamaları farklı bir sertifikayla yayınlayabilirsiniz, ancak aynı uygulamanın yeni sürümlerini yayınlayamazsınız.)

Önemli: *Adobe, kendinden imzalı sertifika sınırlamaları nedeniyle, genel olarak yayınlanan AIR uygulamalarının imzalanması için kesinlikle güvenilir bir sertifika yetkilisi tarafından yayınlanan bir ticari sertifika kullanmanızı önerir.*

ADT tarafından oluşturulan sertifika ve ilişkilendirilmiş özel anahtar, PKCS12 türü bir anahtar deposu dosyasında saklanır. Belirtilen şifre anahtar deposunda değil, anahtarın kendisinde ayarlanır.

Sertifika oluşturma örnekleri

```
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
```

AIR dosyalarını imzalarken bu sertifikaları kullanmak için, aşağıdaki imzalama seçeneklerini ADT `-package` veya `-prepare` komutlarıyla kullanırsınız:

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

Not: *1.5 ve üstü Java sürümleri, PKCS12 sertifika dosyalarını korumak için kullanılan şifrelerde yüksek ASCII karakterlerini kabul etmez. Şifrede normal ASCII karakterleri kullanın.*

Bölüm 14: AIR uygulama tanımlayıcı dosyaları

Her AIR uygulaması bir uygulama tanımlayıcı dosyası gerektirir. Uygulama tanımlayıcı dosyası, uygulamanın temel özelliklerini tanımlayan bir XML belgesidir.

AIR'yi destekleyen çoğu geliştirme ortamı, bir proje oluşturduğunuzda otomatik olarak bir uygulama tanımlayıcısı oluşturur. Yoksa kendi tanımlayıcı dosyanızı oluşturmanız gerekir. Hem AIR hem de Flex SDK'sinin `samples` dizininde `descriptor-sample.xml` adlı bir örnek tanımlayıcı dosya bulunabilir.

Uygulama tanımlayıcı dosyası için herhangi bir dosya adı kullanılabilir. Uygulamayı paketlediğinizde, uygulama tanımlayıcı dosyası `application.xml` olarak yeniden adlandırılır ve AIR paketinin içinde özel bir dizine yerleştirilir.

Örnek uygulama tanımlayıcısı

Aşağıdaki uygulama tanımlayıcı belgesi birçok AIR uygulaması tarafından kullanılan temel özellikleri ayarlar:

```
<?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>
```

Uygulama kök içeriği olarak bir SWF dosyası yerine bir HTML dosyası kullanırsa, yalnızca `<content>` öğesi farklı olur:

```
<content>
    HelloWorld.html
</content>
```

Uygulama tanımlayıcısı değişiklikleri

AIR uygulama tanımlayıcısı aşağıdaki AIR sürümlerinde değişmiştir.

AIR 1.1 tanımlayıcı değişiklikleri

Uygulamanın `name` ve `description` öğelerinin `text` öğesi kullanılarak yerelleştirilmesine izin verildi.

AIR 1.5 tanımlayıcı değişiklikleri

`contentType` öğesi `fileType` öğesinin gerekli alt öğesi oldu.

AIR 1.5.3 tanımlayıcı değişiklikleri

Uygulamaların yayıncı kimliği değeri belirtmesine izin vermek için `publisherID` öğesi eklendi.

AIR 2.0 tanımlayıcı değişiklikleri

Eklenenler:

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (bkz. imageNxN)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

AIR 2.5 tanımlayıcı değişiklikleri

Kaldırılanlar: `version`

Eklenenler:

- `android`
- `extensionID`
- `extensions`
- `image36x36` (bkz. imageNxN)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

AIR 2.6 tanımlayıcı değişiklikleri

Eklenenler:

- `image114x114` (bkz. `imageNxN`)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

AIR 3.0 tanımlayıcı değişiklikleri

Eklenenler:

- `colorDepth`
- `renderMode` için geçerli bir değer olarak *direct*
- renderMode öğesi, masaüstü platformlar için artık yoksayılmıyor.
- Android `<uses-sdk>` öğesi belirtilebilir. (Önceden izin verilmemekteydi.)

AIR 3.1 tanımlayıcı değişiklikleri

Eklenenler:

- "Entitlements" sayfa 215

AIR 3.2 tanımlayıcı değişiklikleri

Eklenenler:

- `depthAndStencil`
- `supportedLanguages`

AIR 3.3 tanımlayıcı değişiklikleri

Eklenenler:

- `aspectRatio` artık `ANY` seçeneği içerir.

AIR 3.4 tanımlayıcı değişiklikleri

Eklenenler:

- `image50x50` (bkz. "`imageNxN`" sayfa 223)
- `image58x58` (bkz. "`imageNxN`" sayfa 223)
- `image100x100` (bkz. "`imageNxN`" sayfa 223)
- `image1024x1024` (bkz. "`imageNxN`" sayfa 223)

AIR 3.6 tanımlayıcı değişiklikleri

Eklenenler:

- "requestedDisplayResolution" sayfa 233 öğesi dahilindeki "iPhone" sayfa 227, şimdi de `excludeDevices` niteliği ekleyerek hangi iOS hedeflerinin yüksek hangilerinin standart çözünürlük kullanacağını belirtmenize olanak tanır

- "initialWindow" sayfa 225 dahilindeki yeni "requestedDisplayResolution" sayfa 233 öğesi, yüksek çözünürlüklü ekranlara sahip Mac bilgisayarlar gibi masaüstü platformlarda yüksek çözünürlüğün mü yoksa standart çözünürlüğün mü kullanılacağını belirtir

AIR 3.7 tanımlayıcı değişiklikleri

Eklenenler:

- "iPhone" sayfa 227 öğesi şimdi de çalışma zamanında yüklenecek bir SWF listesi belirtmenize olanak tanıyan "externalSwfs" sayfa 217 öğesini sunmaktadır.
- "iPhone" sayfa 227 öğesi şimdi de CPU oluşturma modunu belirtilen bir aygıt kümesi için zorlamanıza olanak tanıyan "forceCPURenderModeForDevices" sayfa 220 öğesini sunmaktadır.

Uygulama tanımlayıcı dosyasının yapısı

Uygulama tanımlayıcı dosyası, aşağıdaki yapıda bir XML belgesidir:

```
<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" sayfa 235>...</"supportedLanguages" sayfa 235>
    <supportedProfiles>...</supportedProfiles>
    <versionNumber>...</versionNumber>
    <versionLabel>...</versionLabel>
</application>
```

AIR uygulama tanımlayıcısı öğeleri

Aşağıdaki öğe sözlüğü, bir AIR uygulaması tanımlayıcı dosyasının geçerli öğelerinden her birini açıklar.

allowBrowserInvocation

Adobe AIR 1.0 ve üstü — İsteğe bağlı

AIR tarayıcı içi API'sinin uygulamayı algılamasını ve başlatmasını sağlar.

Bu ayarı `true` olarak ayarlarsanız, güvenlik sonuçlarını göz önünde bulundurduğunuzdan emin olun. Bunlar, Tarayıcıdan AIR uygulaması başlatma (ActionScript geliştiricileri için) ve Tarayıcıdan AIR uygulaması başlatma (HTML geliştiricileri için) başlıklarında açıklanmaktadır.

Daha fazla bilgi için bkz. "Yüklenmiş bir AIR uygulamasını tarayıcıdan başlatma" sayfa 252.

Üst öğe:“application” sayfa 206

Alt öğeler: yok

İçerik

`true` veya `false` (varsayılan)

Örnek

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 ve üstü — İsteğe bağlı

Android bildirim dosyasına öğe eklemenize izin verir. AIR, her APK paketi için AndroidManifest.xml dosyası oluşturur. Ek öğeler eklemek için AIR uygulama tanımlayıcısında android öğesini kullanabilirsiniz. Android dışında tüm platformlarda yoksayılır.

Üst öğe:“application” sayfa 206

Alt öğeler:

- “colorDepth” sayfa 210
- “manifestAdditions” sayfa 228

İçerik

Android uygulama bildirimine eklemek için Android'e özgü özellikleri tanımlayan öğeler.

Örnek

```
<android>
    <manifestAdditions>
        ...
    </manifestAdditions>
</android>
```

Daha fazla Yardım konusu

“Android ayarları” sayfa 73

AndroidManifest.xml Dosyası

application

Adobe AIR 1.0 ve üstü — Zorunlu

Bir AIR uygulama tanımlayıcısı belgesinin kök öğesi.

Üst öğe: yok

Alt öğeler:

- “allowBrowserInvocation” sayfa 205
- “android” sayfa 206

- "copyright" sayfa 212
- "customUpdateUI" sayfa 212
- "description" sayfa 213
- "extensions" sayfa 216
- "filename" sayfa 217
- "fileTypes" sayfa 219
- "icon" sayfa 222
- "id" sayfa 222
- "initialWindow" sayfa 225
- "installFolder" sayfa 226
- "iPhone" sayfa 227
- "name" sayfa 230
- "programMenuFolder" sayfa 232
- "publisherID" sayfa 232
- "supportedLanguages" sayfa 235
- "supportedProfiles" sayfa 236
- "version" sayfa 238
- "versionLabel" sayfa 239
- "versionNumber" sayfa 239

Nitelikler

minimumPatchLevel — Bu uygulama tarafından gerekli olan AIR çalışma zamanı en az düzeltme düzeyi.

xmlns — XML ad alanı niteliği, uygulamanın gerekli AIR çalışma zamanı sürümünü belirler.

Ad alanı AIR'nin her ana sürümüyle (küçük yamalarla değil) birlikte değişir. "3.0" gibi ad alanının son parçası, uygulama tarafından gerekli kılınan çalışma zamanı sürümünü gösterir.

Ana AIR sürümlerinin xmlns değerleri şu şekildedir:

```
xmlns="http://ns.adobe.com/air/application/1.0"
xmlns="http://ns.adobe.com/air/application/1.1"
xmlns="http://ns.adobe.com/air/application/1.5"
xmlns="http://ns.adobe.com/air/application/1.5.2"
xmlns="http://ns.adobe.com/air/application/1.5.3"
xmlns="http://ns.adobe.com/air/application/2.0"
xmlns="http://ns.adobe.com/air/application/2.5"
xmlns="http://ns.adobe.com/air/application/2.6"
xmlns="http://ns.adobe.com/air/application/2.7"
xmlns="http://ns.adobe.com/air/application/3.0"
xmlns="http://ns.adobe.com/air/application/3.1"
xmlns="http://ns.adobe.com/air/application/3.2"
xmlns="http://ns.adobe.com/air/application/3,3"
xmlns="http://ns.adobe.com/air/application/3.4"
xmlns="http://ns.adobe.com/air/application/3.5"
xmlns="http://ns.adobe.com/air/application/3.6"
xmlns="http://ns.adobe.com/air/application/3.7"
```

SWF tabanlı uygulamalar için, uygulama tanımlayıcıda belirtilen AIR çalışma zamanı sürümü uygulamanın başlangıç içeriği olarak yüklenebilecek maksimum SWF sürümünü belirler. AIR 1.0 veya AIR 1.1'i belirten uygulamalar, AIR 2 çalışma zamanını kullanırken bile başlangıç içeriği olarak yalnızca SWF9 (Flash Player 9) dosyalarını kullanabilir. AIR 1.5'i (veya daha sonrasını) belirten uygulamalar başlangıç içeriği olarak SWF9 veya SWF10 (Flash Player 10) dosyalarını kullanabilir.

SWF sürümü, hangi AIR veya Flash Player API sürümünün kullanılabilir olduğunu belirler. AIR 1.5 uygulamasının başlangıç içeriği olarak bir SWF9 dosyası kullanıldığında, uygulamanın yalnızca AIR 1.1 ve Flash Player 9 API'lerine erişimi olacaktır. Ayrıca, AIR 2.0 veya Flash Player 10.1'deki API'lerde yapılan davranış değişiklikleri etkili olmaz. (API'lerdeki güvenlik ile ilgili önemli değişiklikler bu ilkelere bir istisna oluşturabilir; güncel ve gelecekteki çalışma zamanı yamalarında geriye dönük olarak uygulanabilir.)

HTML tabanlı uygulamalar için, uygulama tanımlayıcıda belirtilen çalışma zamanı sürümü hangi AIR ve Flash Player API sürümünün uygulama için kullanılabilir olduğunu belirler. HTML, CSS ve JavaScript davranışları uygulama tanımlayıcı tarafından değil, her zaman yüklenmiş olan AIR çalışma zamanında kullanılan Webkit sürümü tarafından belirlenir.

AIR uygulaması SWF içeriğini yüklediğinde, bu içerik için kullanılabilir olan AIR ve Flash Player API'lerinin sürümü içeriğin nasıl yüklendiğine bağlıdır. Etkili sürüm bazen uygulama tanımlayıcısı ad alanı tarafından, bazen yüklenen içeriğin sürümü tarafından, bazen de yüklenmiş olan içeriğin sürümü tarafından belirlenir. Aşağıdaki tablo, API sürümünün yükleme yöntemine göre nasıl belirlendiğini gösterir:

İçerik nasıl yüklenir	**API sürümü nasıl belirlenir**
Başlangıç içeriği, SWF tabanlı uygulama	**Yüklenen** dosyanın SWF sürümü
Başlangıç içeriği, HTML tabanlı uygulama	Uygulama tanımlayıcı ad alanı
SWF içeriği tarafından yüklenen SWF	**Yükleme** içeriği sürümü
<script> etiketi kullanılarak HTML içeriği tarafından yüklenen SWF kitaplığı	Uygulama tanımlayıcı ad alanı
HTML içeriği tarafından AIR veya Flash Player API'leri (flash.display.Loader gibi) kullanılarak yüklenen SWF	Uygulama tanımlayıcı ad alanı
HTML içeriği tarafından <object> veya <embed> etiketleri (veya eşdeğer JavaScript API'leri) kullanılarak yüklenen SWF	**Yüklenen** dosyanın SWF sürümü

Yükleme içeriğinden farklı bir sürümün SWF dosyasını yükleme sırasında iki sorunla karşılaşabilirsiniz:

- Daha eski bir sürüme sahip SWF ile daha yeni bir sürüme sahip SWF yükleme— AIR'nin daha yeni sürümlerinde eklenen API'lere ve yüklü içerikteki Flash Player'a yapılan başvurular çözümlenmez.
- Daha yeni bir sürüme sahip SWF ile daha eski bir SWF yükleme — AIR'nin yeni sürümlerinde değiştirilen API'ler ve Flash Player yüklenen içeriğin beklemediği şekillerde davranabilir.

İçerik

Uygulama öğesi bir AIR uygulamasının özelliklerini tanımlayan alt öğeleri içerir.

Örnek

```
<?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 ve sonraki sürümleri, iOS ve Android — İsteğe Bağlı

Uygulamanın en boy oranını belirtir.

Belirtilmezse, uygulama aygıtın "doğal" en boy oranını ve yönlendirmesini açar. Doğal yönlendirme aygıttan aygıta değişir. Genellikle, telefon gibi küçük ekranlı aygıtlarda dikey en boy oranıdır. iPad tableti gibi bazı aygıtlarda uygulama geçerli yönlendirmede açılır. AIR 3.3 ve sonraki sürümlerde bu, yalnızca başlangıç görünümü değil uygulamanın tamamı için geçerlidir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

`portrait, landscape` veya `any`

Örnek

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

autoOrients

Adobe AIR 2.0 ve sonraki sürümleri, iOS ve Android — İsteğe Bağlı

Aygıt kendi fiziksel yönlendirmesini değiştirdiğinde uygulamadaki içerik yönlendirmesinin de kendiliğinden değişip değişmediğini belirtir. Daha fazla bilgi için bkz. Sahne alanı yönlendirmesi.

Otomatik yönlendirme kullanırken Sahne Alanı'nın `align` ve `scaleMode` özelliklerini aşağıdakilere ayarlamayı göz önünde bulundurun:

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Bu ayarlar, uygulamanın sol üst köşenin etrafında dönmesine izin verir ve uygulama içeriğinizin otomatik olarak ölçeklenmesini engeller. Diğer ölçek modları içeriğinizi dönmüş sahne alanı boyutlarına sığdırmak için ayarlarken, içeriği kırpar, deformasyona uğratır veya aşırı derecede küçültür. İçeriği kendiniz yeniden çizerek veya tekrar aktararak hemen hemen her zaman daha iyi sonuçlara ulaşabilirsiniz.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

`true` veya `false` (varsayılan)

Örnek

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

colorDepth

Adobe AIR 3 ve üstü — İsteğe bağlı

16 bit rengin mi yoksa 32 bit rengin mi kullanılacağını belirtir.

16 bit renk kullanmak oluşturma performansını arttırır, ancak rengin aslına uygunluğu azalabilir. AIR 3'ten önce Android'de her zaman 16 bit renk kullanılmaktadır. AIR 3'te 32 bit renk varsayılan olarak kullanılır.

***Not:** Uygulamanız StageVideo sınıfını kullanıyorsa 32 bit renk kullanmanız gerekir.*

Üst öğe:"android" sayfa 206

Alt öğeler: yok

İçerik
Aşağıdaki değerlerden biri:

- 16 bit
- 32 bit

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

containsVideo

Uygulamanın herhangi bir video içerip içermeyeceğini belirtir.

Üst öğe:"android" sayfa 206

Alt öğeler: yok

İçerik
Aşağıdaki değerlerden biri:

- true
- false

Örnek
```
<android>
    <containsVideo>true</containsVideo>
    <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 ve üstü — Zorunlu

`content` öğesi için belirtilen değer, uygulamanın ana içerik dosyasının URL'sidir. Bu bir SWF dosyası veya HTML dosyası olabilir. URL, uygulama yükleme klasörünün köküne bağlı olarak belirtilir. (ADL ile bir AIR uygulamasını çalıştırırken, URL, uygulama tanımlayıcı dosyasını içeren klasöre bağlıdır. Farklı bir kök dizin belirtmek için, ADL'nin `root-dir` parametresini kullanabilirsiniz.)

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik
Uygulama dizinine göre bir URL. İçerik öğesinin değeri bir URL gibi işlev gördüğü için, içerik dosyasının adındaki karakterler RFC 1738'de belirtilen kurallara göre URL kodlamalı olmalıdır. Örneğin boşluk karakterleri `%20` olarak kodlanmalıdır.

Örnek

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0 ila 1.1 — İsteğe bağlı; AIR 1.5 ve üstü — Gerekli

AIR 1.5'ten itibaren `contentType` zorunludur (AIR 1.0 ve 1.1'de isteğe bağlıydı). Özellik bazı işletim sistemlerinin dosyayı açmak için en iyi uygulamayı bulmasına yardımcı olur. Değer, dosya içeriğinin MIME türü olmalıdır. Dosya türü önceden kaydedildiyse ve atanmış bir MIME türüne sahipse, değerin Linux'ta yok sayıldığını dikkate alın.

Üst öğe:"fileType" sayfa 218

Alt öğeler: yok

İçerik

MIME türü ve alt tür. MIME türleriyle ilgili daha fazla bilgi için bkz. RFC2045.

Örnek

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 ve üstü — İsteğe bağlı

AIR uygulamasının telif hakkı bilgisi. Mac OS'de telif hakkı metni, yüklü uygulamanın Hakkında iletişim penceresinde görüntülenir. Mac OS'de telif hakkı bilgileri ayrıca, uygulamanın Info.plist dosyasının NSHumanReadableCopyright alanında da kullanılır.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik

Uygulamanın telif hakkı bilgilerini içeren bir dize.

Örnek

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Bir uygulamanın kendi güncelleme iletişim kutularını sağlayıp sağlayamayacağını gösterir. Değer `false` ise AIR kullanıcıya standart güncelleme iletişim kutuları sunar. Yalnızca AIR dosyaları olarak dağıtılmış uygulamalar yerleşik AIR güncelleme sistemini kullanabilir.

Uygulamanızın yüklü sürümünde `customUpdateUI` öğesi `true` olarak ayarlıyken kullanıcı yeni bir sürüm için AIR dosyasını çift tıklattığında veya kesintisiz yükleme özelliğini kullanarak uygulamanın güncellemesini yüklediğinde, çalışma zamanı, uygulamanın yüklü sürümünü açar. Çalışma zamanı varsayılan AIR uygulama yükleyicisini açmaz. Uygulama mantığınız güncelleme işlemine nasıl devam edileceğini belirleyebilir. (Bir yükseltme işleminin devam edebilmesi için, AIR dosyasındaki uygulama kimliği ve yayıncı kimliği yüklü uygulamadaki değerlerle eşleşmelidir.)

Not: *`customUpdateUI` mekanizması, yalnızca uygulamanın önceden yüklenmiş olduğu ve kullanıcının bir güncelleme içeren AIR yükleme dosyasını çift tıklattığı veya kesintisiz yükleme özelliğini kullanarak uygulamanın güncellemesini yüklediği durumlarda çalışır. `customUpdateUI`true olsa da olmasa da, özel kullanıcı arabiriminizi gerekli şekilde görüntüleyerek kendi uygulama mantığınızı kullanıp bir güncellemeyi indirebilir ve başlatabilirsiniz.*

Daha fazla bilgi için bkz. "AIR uygulamalarını güncelleme" sayfa 254.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik

`true` veya `false` (varsayılan)

Örnek

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 ve üstü — İsteğe bağlı

Uygulamanın, derinlik veya şablon arabelleği kullanımını gerektirdiğini gösterir. Genelde 3B içeriği ile çalışırken bu arabellekleri kullanırsınız. Varsayılan olarak, derinlik ve şablon arabelleklerini devre dışı bırakmak için bu öğenin değeri `false` şeklindedir. Arabelleklerin, herhangi bir içerik yüklenmeden önce uygulamanın başlatılması sırasında ayrılması gerektiğinden bu öğe gereklidir.

Bu öğenin ayarı, `enableDepthAndStencil` argümanı için `Context3D.configureBackBuffer()` yöntemine aktarılan değerle eşleşmelidir. Değerler eşleşmiyorsa AIR bir hata bildirir.

Bu öğe yalnızca `renderMode` = `direct` durumunda uygulanabilir. `renderMode`, `direct` ile eşit değilse ADT, hata 118'i oluşturur:

```
<depthAndStencil> element unexpected for render mode cpu.  It requires "direct" render mode.
```

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

`true` veya `false` (varsayılan)

Örnek

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

description

Adobe AIR 1.0 ve üstü — İsteğe bağlı

AIR uygulama yükleyicisinde görüntülenen uygulama açıklaması.

Tek bir metin düğümü belirtirseniz (birden çok text öğesi değil), AIR uygulama yükleyicisi sistem diline bakılmaksızın bu açıklamayı kullanır. Aksi takdirde, AIR uygulama yükleyicisi, kullanıcının işletim sisteminin kullanıcı arabirimi diline en yakın eşleşme olan açıklamayı kullanır. Örneğin, uygulama tanımlayıcı dosyasının `description` öğesinin en (İngilizce) yerel ayarları için bir değer içerdiği bir yükleme düşünün. Kullanıcının sistemi en (İngilizce) adını kullanıcı arabirimi dili olarak tanımlıyorsa, AIR uygulama yükleyicisi en açıklamasını kullanır. Ayrıca sistem kullanıcı arabirimi adı en-US (ABD İngilizcesi) olduğunda da en açıklamasını kullanır. Ancak sistem kullanıcı arabirimi dili en-US ise ve uygulama tanımlayıcı dosyası hem en-US hem de en-GB adlarını tanımlıyorsa, AIR uygulama yükleyicisi en-US değerini kullanır. Uygulama, sistem kullanıcı arabirimi diliyle eşleşen bir ad tanımlamazsa, AIR uygulama yükleyicisi, uygulama tanımlayıcı dosyasında tanımlı olan ilk `description` değerini kullanır.

Çok dilli uygulamaları geliştirme konusunda daha fazla bilgi için bkz. “AIR uygulamalarını yerelleştirme” sayfa 289.

Üst öğe:“application” sayfa 206

Alt öğeler:“text” sayfa 237

İçerik

AIR 1.0 uygulama tanımlayıcı şeması, ad için yalnızca bir tane basit metin düğümünün tanımlanmasına izin verir. (birden çok `text` öğesi değil.)

AIR 1.1'de (veya daha yüksek sürümlerde), `description` öğesinde birden çok dil belirleyebilirsiniz. Her metin öğesinin `xml:lang` niteliği, RFC4646'da da (http://www.ietf.org/rfc/rfc4646.txt) tanımlandığı gibi bir dil kodu belirtir.

Örnek

Basit metin düğümüne sahip açıklama:

```
<description>This is a sample AIR application.</description>
```

İngilizce, Fransızca ve İspanyolca için yerelleştirilmiş metin öğelerine sahip açıklama (AIR 1.1 ve üstünde geçerlidir):

```
<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 ve üstü — Zorunlu

Dosya türü açıklaması kullanıcıya işletim sistemi tarafından görüntülenir. Dosya türü açıklaması yerelleştirilebilir değildir.

Ayrıca bkz.: uygulama öğesinin alt öğesi olarak “description” sayfa 213

Üst öğe:“fileType” sayfa 218

Alt öğeler: yok

İçerik

Dosya içeriğini açıklayan bir dize.

Örnek

```
<description>PNG image</description>
```

embedFonts

AIR uygulamasındaki StageText üzerinde özel yazı tipleri kullanmanıza olanak tanır. Bu öğe isteğe bağlıdır.

Üst öğe:"application" sayfa 206

Alt öğeler:"font" sayfa 219

İçerik

EmbedFonts öğesi herhangi bir sayıda font öğesi içerebilir.

Örnek

```
<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 ve sonraki sürümleri — yalnızca iOS, İsteğe Bağlı

iOS, ek kaynaklara ve yeteneklere uygulama erişimi sağlamak için entitlements adı verilen özellikler kullanır. Bir mobil iOS uygulamasında bu bilgileri belirtmek için Entitlements öğesini kullanın.

Üst öğe:"iPhone" sayfa 227

Alt öğeler: iOS Entitlements.plist elements

İçerik

Uygulamanın Entitlements.plist ayarları olarak kullanmak üzere anahtar-değer çiftleri belirten alt öğeler içerir. Entitlements öğesinin içeriği, bir CDATA bloğu içine alınmalıdır. Daha fazla bilgi için iOS Geliştirici Kütüphanesi'ndeki Entitlement Key Reference (Entitlement Anahtar Başvurusu) konusuna bakın.

Örnek

```
<iphone>
...
   <Entitlements>
      <![CDATA[
         <key>aps-environment</key>
         <string>development</string>
      ]]>
   </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 ve üstü — Zorunlu

Bir dosya türünün uzantı dizesi.

Üst öğe:"fileType" sayfa 218

Alt öğeler: yok

İçerik
Dosya uzantısı karakterlerini tanımlayan bir dize (nokta "." olmadan)

Örnek
```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 ve üstü

Uygulama tarafından kullanılan bir ActionScript uzantısının kimliğini belirtir. Kimlik uzantı tanımlayıcı belgesinde tanımlanır.

Üst öğe:"extensions" sayfa 216

Alt öğeler: yok

İçerik
ActionScript uzantısının kimliğini tanımlayan bir dize.

Örnek
```
<extensionID>com.example.extendedFeature</extensionID>
```

extensions

Adobe AIR 2.5 ve üstü — İsteğe bağlı

Bir uygulama tarafından kullanılan ActionScript uzantılarını tanımlar.

Üst öğe:"application" sayfa 206

Alt öğeler:"extensionID" sayfa 216

İçerik
Uzantı tanımlayıcı dosyasındaki ActionScript uzantısı kimliklerini içeren alt `extensionID` öğeleri.

Örnek
```
<extensions>
    <extensionID>extension.first</extensionID>
    <extensionID>extension.next</extensionID>
    <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 ve üstü, yalnızca iOS— İsteğe bağlı

Uzaktan barındırma işlemi için ADT tarafından yapılandırılacak bir SWF listesi içeren metin dosyasının adını belirtir. Uygulamanız tarafından kullanılan SWF'lerin bir alt kümesini paketleyerek ve çalışma zamanında `Loader.load()` yöntemiyle kalan (yalnızca varlık içeren) harici SWF'lerini yükleyerek ilk uygulama indirme boyutunuzu küçültebilirsiniz. Bu özelliği kullanmak için uygulamayı, ADT harici olarak yüklenen SWF dosyalarındaki tüm ActionScript ByteCode (ABC) öğelerini ana uygulama SWF'sine taşıyıp, yalnızca varlık içeren bir SWF dosyası bırakacak şekilde paketlemeniz gerekir. Bunun, bir uygulamanın yüklenmesinin ardından herhangi bir kodun indirilmesini yasaklayan Apple Store kuralıyla uyumlu olması gerekir.

Daha fazla bilgi için bkz. “Harici, yalnızca varlık içeren SWF'leri yükleyerek indirme boyutunu küçültme” sayfa 84.

Üst öğe:“iPhone” sayfa 227, “initialWindow” sayfa 225

Alt öğeler: yok

İçerik

Barındırma işlemi uzaktan gerçekleştirilecek, çizgiyle sınırlandırılmış SWF listesi içeren metin dosyasının adı.

Nitelikler:

Yok.

Örnekler

iOS:

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

filename

Adobe AIR 1.0 ve üstü — Zorunlu

Uygulama yüklendiğinde uygulamanın dosya adı olarak kullanılacak dize (uzantısız). Uygulama dosyası çalışma zamanında AIR uygulamasını başlatır. `name` değeri sağlanmazsa, `filename` öğesi aynı zamanda yükleme klasörünün adı olarak da kullanılır.

Üst öğe:“application” sayfa 206

Alt öğeler: yok

İçerik

`filename` özelliği, çeşitli dosya sistemlerinde dosya adları olarak kullanımı yasak olan aşağıdaki karakterler dışında herhangi bir Unicode (UTF-8) karakterini içerebilir:

Karakter	Onaltılık Kod
çeşitli	0x00 – x1F
*	x2A
"	x22

Karakter	Onaltılık Kod
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

`filename` değeri noktayla bitemez.

Örnek

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Uygulamanın kaydolabileceği tek dosya türünü açıklar.

Üst öğe:"fileTypes" sayfa 219

Alt öğeler:

- "contentType" sayfa 212
- "description" sayfa 214
- "extension" sayfa 216
- "icon" sayfa 222
- "name" sayfa 231

İçerik

Bir dosya türünü açıklayan öğeler.

Örnek

```
<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 ve üstü — İsteğe bağlı

`fileTypes` öğesi sayesinde, bir AIR uygulamasının ilişkilendirildiği dosya türlerini bildirebilirsiniz.

Bir AIR uygulaması yüklendiğinde, bildirilen tüm dosya türleri işletim sistemiyle kaydedilir ve önceden başka bir uygulamayla ilişkilendirilmemişse, AIR uygulamasıyla ilişkilendirilir. Bir dosya türüyle başka bir uygulama arasındaki mevcut ilişkiyi geçersiz kılmak için, çalışma zamanında `NativeApplication.setAsDefaultApplication()` yöntemini kullanın (tercihen kullanıcının izniyle).

***Not:** Çalışma zamanı yöntemleri yalnızca uygulama tanımlayıcıda bildirilen dosya türlerinin ilişkilendirmelerini yönetebilir.*

`fileTypes` öğesi isteğe bağlıdır.

Üst öğe:"application" sayfa 206

Alt öğeler:"fileType" sayfa 218

İçerik

`fileTypes` öğesi herhangi bir sayıda `fileType` öğesi içerebilir.

Örnek

```
<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>
```

font

AIR uygulamasında kullanılabilecek tek bir özel yazı tipini açıklar.

Üst öğe:"embedFonts" sayfa 215

Alt öğeler:"fontName" sayfa 220, "fontPath" sayfa 220

İçerik

Özel yazı tipi adını ve yolunu belirten öğeler.

Örnek

```
<font>
   <fontPath>ttf/space age.ttf</fontPath>
   <fontName>space age</fontName>
</font>
```

fontName

Özel yazı tipinin adını belirtir.

Üst öğe:"font" sayfa 219

Alt öğeler: Yok

İçerik

StageText.fontFamily içinde belirtilecek özel yazı tipinin adı

Örnek

```
<fontName>space age</fontName>
```

fontPath

Özel yazı tipi dosyasının konumunu verir.

Üst öğe:"font" sayfa 219

Alt öğeler: Yok

İçerik

Özel yazı tipi dosyasının yolu (kaynağa göre).

Örnek

```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 ve üstü, yalnızca iOS— İsteğe bağlı

CPU oluşturma modunu belirtilen bir aygıt kümesi için zorlayın. Bu özellik, kalan iOS aygıtları için GPU oluşturma modunu seçime bağlı olarak etkinleştirmenize etkili bir şekilde olanak tanır.

Bu etiketi `iPhone` etiketinin alt öğesi olarak ekleyin ve boşlukla ayrılmış bir aygıt modeli ad listesi belirtin. Geçerli aygıt modeli adlarına şunlar dahildir:

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		

Üst öğe:"iPhone" sayfa 227, "initialWindow" sayfa 225

Alt öğeler: yok

İçerik
Boşlukla ayrılmış aygıt modeli ad listesi.

Nitelikler:
Yok.

Örnekler
iOS:

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

fullScreen

Adobe AIR 2.0 ve sonraki sürümleri, iOS ve Android — İsteğe Bağlı

Uygulamanın tam ekran modunda başlayıp başlamadığını belirtir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik
`true` veya `false` (varsayılan)

Örnek

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Uygulamanın ana penceresinin başlangıçtaki yüksekliği.

Bir yükseklik ayarlamazsanız, yükseklik kök SWF dosyasındaki ayarlar tarafından veya HTML tabanlı bir AIR uygulaması olması durumunda işletim sistemi tarafından belirlenir.

AIR 2'de bir pencerenin maksimum yüksekliği 2048 pikselden 4096 piksele değişmiştir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik
Maksimum 4095 değerine sahip pozitif bir tamsayı.

Örnek
```
<height>4095</height>
```

icon

Adobe AIR 1.0 ve üstü — İsteğe bağlı

`icon` özelliği, uygulama için kullanılacak bir veya daha fazla simge dosyasını belirtir. Bir simge dahil etmek isteğe bağlıdır. Bir `icon` özelliği belirtmezseniz, işletim sistemi varsayılan bir simge görüntüler.

Belirtilen yol, uygulama kök dizinine bağlıdır. Simge dosyaları PNG biçiminde olmalıdır. Aşağıdaki simge boyutlarının tümünü belirtebilirsiniz:

Belirli bir boyut için öğe mevcutsa, dosyadaki görüntü tam olarak belirtilen boyutta olmalıdır. Tüm boyutlar sağlanmamışsa en yakın boyut, işletim sistemi tarafından simgenin kullanımı için uygun şekilde ölçeklenir.

***Not:** Belirtilen simgeler AIR paketine otomatik olarak eklenmez. Uygulama paketlendiğinde simge dosyaları doğru, göreceli konumlarına dahil edilmelidir.*

En iyi sonuç için, her kullanılabilir boyut için bir görüntü sağlayın. Buna ek olarak, simgelerin hem 16 hem de 32 bit renk modlarında düzgün göründüğünden emin olun.

Üst öğe:"application" sayfa 206

Alt öğeler:"imageNxN" sayfa 223

İçerik
İstenen her simge boyutu için bir imageNxN öğesi.

Örnek
```
<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 ve üstü — Zorunlu

Uygulama kimliği olarak da bilinen, uygulamaya ilişkin tanımlayıcı bir dize. Genellikle ters DNS stili tanımlayıcı kullanılır ama bu stil önerilmez.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik
ID değeri aşağıdaki karakterlerle sınırlıdır:

- 0–9
- a–z
- a–z
- . (nokta)
- - (tire)

Değer 1 ila 212 karakter arasında olmalıdır. Bu öğe gereklidir.

Örnek

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Bir simgenin yolunu uygulama tanımlayıcısına göre tanımlar.

Her biri farklı bir simge boyutunu belirten aşağıdaki simge görüntüleri kullanılabilir:

- 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+)

Simge tam olarak image öğesi tarafından belirtilen boyuttaki bir PNG grafiği olmalıdır. Simge dosyaları uygulama paketine dahil edilmelidir; uygulama tanımlayıcısı belgesinde başvurulan simgeler otomatik olarak dahil edilmez.

Üst öğe:“application” sayfa 206

Alt öğeler: yok

İçerik

Simgenin dosya yolu, çeşitli dosya sistemlerinde dosya adı olarak kullanılması yasaklanmış aşağıdaki karakterler dışında her Unicode (UTF-8) karakteri içerebilir:

Karakter	Onaltılık Kod
çeşitli	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

Örnek

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Bir iOS uygulamasının ek özelliklerini belirtmenize izin verir.

Üst öğe:“iPhone” sayfa 227

Alt öğeler: iOS Info.plist elements

İçerik

Uygulama için Info.plist ayarları olarak kullanılacak tuş değeri çiftlerini belirten alt öğeler içerir. InfoAdditions öğesinin içeriği bir CDATA bloğu içine alınmalıdır.

Anahtar değer çiftleriyle ve bunların XML'de nasıl ifade edilebileceğiyle ilgili bilgi için Apple iPhone Reference Library'deki (Apple iPhone Reference Kütüphanesi) Information Property List Key Reference (Bilgi Özelliği Listesi Ana Başvurusu) bölümüne bakın.

Örnek

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

Daha fazla Yardım konusu

“iOS Ayarları” sayfa 79

initialWindow

Adobe AIR 1.0 ve üstü — Zorunlu

Ana içerik dosyasını ve uygulamanın başlangıç görünümünü tanımlar.

Üst öğe:"application" sayfa 206

Alt öğeler: Aşağıdaki öğelerin tümü initialWindow öğesinin alt öğesi olarak görünebilir. Ancak, bazı öğeler AIR'nin bir platformda pencere destekleyip desteklememesine göre yoksayılır:

Öğe	Masaüstü	Mobil
"aspectRatio" sayfa 209	yoksayılır	kullanılır
"autoOrients" sayfa 210	yoksayılır	kullanılır
"content" sayfa 211	kullanılır	kullanılır
"depthAndStencil" sayfa 213	kullanılır	kullanılır
"fullScreen" sayfa 221	yoksayılır	kullanılır
"height" sayfa 221	kullanılır	yoksayılır
"maximizable" sayfa 229	kullanılır	yoksayılır
"maxSize" sayfa 229	kullanılır	yoksayılır
"minimizable" sayfa 230	kullanılır	yoksayılır
"minSize" sayfa 230	kullanılır	yoksayılır
"renderMode" sayfa 232	kullanılır (AIR 3.0 ve sonraki sürümleri)	kullanılır
"requestedDisplayResolution" sayfa 233	kullanılır (AIR 3.6 ve sonraki sürümleri)	yoksayılır
"resizable" sayfa 234	kullanılır	yoksayılır
"softKeyboardBehavior" sayfa 235	yoksayılır	kullanılır
"systemChrome" sayfa 237	kullanılır	yoksayılır
"title" sayfa 238	kullanılır	yoksayılır
"transparent" sayfa 238	kullanılır	yoksayılır
"visible" sayfa 240	kullanılır	yoksayılır
"width" sayfa 240	kullanılır	yoksayılır
"x" sayfa 240	kullanılır	yoksayılır
"y" sayfa 241	kullanılır	yoksayılır

İçerik

Uygulama görünümünü ve davranışını tanımlayan alt öğeler.

Örnek

```
<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 ve üstü — İsteğe bağlı

Varsayılan yükleme dizininin alt dizinini tanımlar.

Windows'ta varsayılan yükleme alt dizini, Program Dosyaları dizinidir. Mac OS'de bu dizin /Applications dizinidir. Linux'ta, /opt/ dizinidir. Örneğin `installFolder` özelliği `"Acme"` olarak ayarlıysa ve bir uygulama `"ExampleApp"` olarak adlandırılırsa, uygulama Windows'ta C:\Program Files\Acme\ExampleApp dizinine, MacOS'de /Applications/Acme/Example.app ve Linux'ta /opt/Acme/ExampleApp dizinine yüklenir.

`installFolder` özelliği isteğe bağlıdır. `installFolder` özelliği tanımlamazsanız uygulama, `name` özelliğine dayalı olarak varsayılan yükleme dizininin alt dizinine yüklenir.

Üst öğe:“application” sayfa 206

Alt öğeler: Yok

İçerik

`installFolder` özelliği, çeşitli dosya sistemlerinde klasör adı olarak kullanımı yasaklanmış olanların dışında tüm Unicode (UTF-8) karakterlerini içerebilir (istisnaların bir listesi için `filename` özelliğine bakın).

Yuvalanmış bir alt dizin belirtmek istiyorsanız, dizin ayırıcı karakteri olarak eğik çizgi (/) karakterini kullanın.

Örnek

```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, yalnızca iOS — İsteğe Bağlı

iOS'a özgü uygulama özelliklerini tanımlar.

Üst öğe:"application" sayfa 206

Alt öğeler:

- "Entitlements" sayfa 215
- "externalSwfs" sayfa 217
- "forceCPURenderModeForDevices" sayfa 220
- "InfoAdditions" sayfa 224
- "requestedDisplayResolution" sayfa 233

Daha fazla Yardım konusu

"iOS Ayarları" sayfa 79

manifest

Adobe AIR 2.5 ve üstü, yalnızca Android — İsteğe bağlı

Uygulama için Android bildirim dosyasına eklenecek bilgileri belirtir.

Üst öğe:"manifestAdditions" sayfa 228

Alt öğeler: Android SDK tarafından tanımlanır.

İçerik

Teknik olarak manifest öğesi AIR uygulama tanımlayıcısı şemasının bir parçası değildir. Android bildirimi XML belgesinin köküdür. Manifest öğesinin içine yerleştirdiğiniz her içerik AndroidManifest.xml şemasıyla uyumlu olmalıdır. AIR araçlarıyla bir APK dosyası oluşturduğunuzda, bildirim öğesindeki bilgi uygulamanın oluşturulan AndroidManifest.xml dosyasının karşılık gelen bölümüne kopyalanır.

Yalnızca AIR tarafından doğrudan desteklenenden daha yeni bir SDK sürümünde kullanılabilir olan Android bildirimi değerleri belirtirseniz, uygulamayı paketlerken ADT'ye `-platformsdk` bayrağını ayarlamanız gerekir. Bayrağı, eklediğiniz değerleri destekleyen bir Android SDK sürümüne giden dosya sistemi yoluna ayarlayın.

Manifest öğesinin AIR uygulama tanımlayıcısında CDATA bloğunun içine alınması gerekir.

Örnek

```
<![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>
]]>
```

Daha fazla Yardım konusu

"Android ayarları" sayfa 73

AndroidManifest.xml Dosyası

manifestAdditions

Adobe AIR 2.5 ve üstü, yalnızca Android

Android bildirim dosyasına eklenecek bilgileri belirtir.

Her Android uygulaması temel uygulama özelliklerini tanımlayan bir bildirim dosyası içerir. Android bildirimi kavram olarak AIR uygulama tanımlayıcısına benzerdir. AIR for Android uygulaması, hem uygulama tanımlayıcıya hem de otomatik olarak oluşturulan Android bildirim dosyasına sahiptir. Bir AIR for Android uygulaması paketlendiğinde, bu `manifestAdditions` öğesindeki bilgiler Android bildirim belgesinin karşılık gelen bölümlerine kopyalanır.

Üst öğe:"android" sayfa 206

Alt öğeler:"manifest" sayfa 227

İçerik

`manifestAdditions` öğesindeki bilgi AndroidManifest XML belgesine eklenir.

AIR uygulama ve çalışma zamanı özelliklerinin düzgün çalıştığından emin olmak için oluşturulan Android bildirim belgesinde çeşitli bildirim girişleri ayarlar. Aşağıdaki ayarları geçersiz kılamazsınız:

Bildirim öğesinin aşağıdaki niteliklerini ayarlayamazsınız:

- package
- android:versionCode
- android:versionName

Ana etkinlik öğesinin aşağıdaki niteliklerini ayarlayamazsınız:

- android:label
- android:icon

Uygulama öğesinin aşağıdaki niteliklerini ayarlayamazsınız:

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

Örnek

```
<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>
```

Daha fazla Yardım konusu

"Android ayarları" sayfa 73

AndroidManifest.xml Dosyası

maximizable

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Pencerenin büyütülüp büyütülemeyeceğini belirtir.

Not: *Mac OS X gibi, pencereleri büyütmenin yeniden boyutlandırma işlemi olduğu işletim sistemlerinde, pencerenin yakınlaştırılmasını ve yeniden boyutlandırılmasını önlemek için hem maximizable hem de resizable* `false` *olarak ayarlı olmalıdır.*

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

`true` (varsayılan) veya `false`

Örnek

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Pencerenin maksimum boyutları. Maksimum boyutu ayarlamazsanız, işletim sistemi tarafından belirlenir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

Maksimum genişliği ve yüksekliği temsil eden, boşlukla ayrılmış iki tamsayı.

Not: *AIR tarafından desteklenen maksimum pencere boyutu, AIR 2'de 2048x2048 pikselden 4096x4096 piksele yükselmiştir. (Ekran koordinatları sıfır tabanlı olduğundan, genişlik ve yükseklik için kullanabileceğiniz maksimum değer 4095'tir.)*

Örnek

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Pencerenin simge durumuna küçültülüp küçültülemeyeceğini belirtir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: Yok

İçerik

`true` (varsayılan) veya `false`

Örnek

```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Pencere için izin verilen minimum değeri belirtir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

Minimum genişliği ve yüksekliği temsil eden, boşlukla ayrılmış iki tamsayı. İşletim sisteminin uyguladığı minimum değerin uygulama tanımlayıcısındaki değer kümesinden öncelikli olduğunu unutmayın.

Örnek

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 ve üstü — İsteğe bağlı

AIR uygulama yükleyicisi tarafından görüntülenen uygulama başlığı.

Hiçbir `name` öğesi belirtilmemişse, AIR uygulama yükleyicisi `filename` öğesini uygulama adı olarak görüntüler.

Üst öğe:"application" sayfa 206

Alt öğeler:"text" sayfa 237

İçerik

Tek bir metin düğümü belirtirseniz (birden çok `<text>` öğesi yerine), AIR uygulama yükleyicisi sistem diline bakılmaksızın bu adı kullanır.

AIR 1.0 uygulama tanımlayıcı şeması, ad için yalnızca bir tane basit metin düğümünün tanımlanmasına izin verir (birden çok `text` öğesi değil). AIR 1.1'de (veya daha yüksek sürümlerde), `name` öğesinde birden çok dil belirleyebilirsiniz.

Her metin öğesinin `xml:lang` niteliği, RFC4646'da da (http://www.ietf.org/rfc/rfc4646.txt) tanımlandığı gibi bir dil kodu belirtir.

AIR uygulama yükleyicisi, kullanıcının işletim sisteminin kullanıcı arabirimi diline en yakın eşleşme olan adı kullanır. Örneğin, uygulama tanımlayıcı dosyasının `name` öğesinin en (English) yerel ayarları için bir değer içerdiği bir yükleme düşünün. İşletim sistemi en (English) adını kullanıcı arabirimi dili olarak tanımlıyorsa, AIR uygulama yükleyicisi en adını kullanır. Ayrıca sistem kullanıcı arabirimi adı en-US (U.S. English) olduğunda da en adını kullanır. Ancak kullanıcı arabirimi dili en-US ise ve uygulama tanımlayıcı dosyası hem en-US hem de en-GB adlarını tanımlıyorsa, AIR uygulama yükleyicisi en-US değerini kullanır. Uygulama, sistem kullanıcı arabirimi dilleriyle eşleşen bir ad tanımlamazsa, AIR uygulama yükleyicisi, uygulama tanımlayıcı dosyasında tanımlı olan ilk `name` değerini kullanır.

`name` öğesi yalnızca AIR uygulama yükleyicisinde kullanılan uygulama başlığını tanımlar. AIR uygulama yükleyicisi birden çok dili destekler: Geleneksel Çince, Basitleştirilmiş Çince, Çekçe, Felemenkçe, İngilizce, Fransızca, Almanca, İtalyanca, Japonca, Korece, Lehçe, Brezilya Portekizcesi, Rusça, İspanyolca, İsveççe ve Türkçe. AIR uygulama yükleyicisi görüntülenen dilini (uygulama başlığı ve açıklamasının dışındaki metinler için), sistem kullanıcı arabirimi diline dayalı olarak seçer. Bu dil seçimi, uygulama tanımlayıcı dosyasındaki ayarlardan bağımsızdır.

`name` öğesi, çalışan, yüklü uygulama için kullanılabilir olan yerel ayarları *tanımlamaz*. Çok dilli uygulamaları geliştirme konusunda ayrıntılı bilgi için bkz. “AIR uygulamalarını yerelleştirme” sayfa 289.

Örnek

Aşağıdaki örnek basit bir metin düğümüyle ad tanımlar:

```
<name>Test Application</name>
```

AIR 1.1 ve üstünde geçerli olan aşağıdaki örnek, <text> öğesi düğümlerini kullanarak adı üç dilde (İngilizce, Fransızca ve İspanyolca) belirtir:

```
<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 ve üstü — Zorunlu

Bir dosya türünün adını tanımlar.

Üst öğe:“fileType” sayfa 218

Alt öğeler: yok

İçerik

Dosya türünün adını temsil eden bir dize.

Örnek

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Windows işletim sisteminde Tüm Programlar menüsünde veya Linux'ta Uygulamalar menüsünde yer alan uygulamalara ait kısayolların nerede konumlandırılacağını belirler. (Bu ayar şu anda diğer işletim sistemlerinde yok sayılmaktadır.)

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik

`programMenuFolder` değeri için kullanılan dize çeşitli dosya sistemlerinde klasör adı olarak kullanımı yasaklanmış olanların dışında tüm Unicode (UTF-8) karakterlerini içerebilir (istisnalar listesi için `filename` öğesine bakın). Bu değerin son karakteri olarak eğik çizgi (/) karakterini *kullanmayın*.

Örnek

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 veya üstü — İsteğe bağlı

Orijinal olarak AIR 1.5.2 sürümü veya daha önceki sürümlerle oluşturulmuş bir AIR uygulamasını güncellemek için yayıncı kimliğini tanımlar.

Yalnızca bir uygulama güncellemesi oluştururken bir yayıncı kimliği belirtin. `publisherID` öğesinin değeri, uygulamanın önceki sürümü için AIR tarafından oluşturulmuş yayıncı kimliğiyle eşleşmelidir. Yüklü bir uygulama için, yayıncı kimliği uygulamanın yüklü olduğu klasörde, `META-INF/AIR/publisherid` dosyasında bulunabilir.

AIR 1.5.3 veya daha üstüyle oluşturulmuş yeni uygulamalar yayıncı kimliği belirtmemelidir.

Daha fazla bilgi için bkz. "AIR yayıncı kimlikleri hakkında" sayfa 187.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik

Bir yayıncı kimliği dizesi.

Örnek

```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 veya üstü — İsteğe bağlı

Geçerli bilgi işlem aygıtında destekleniyorsa grafik işleme birimi (GPU) hızlandırmasının kullanılıp kullanılmayacağını belirtir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik
Aşağıdaki değerlerden biri:

- `auto` (varsayılan) — şu anda İşlemci moduna geri döner.
- `cpu` — donanım hızlandırma kullanılmaz.
- `direct` — oluşturma kompozisyonu CPU'da oluşur; doldurarak oluşturma GPU'yu kullanır. AIR 3 ve üstünde kullanılabilir

  Not: *Mobil platformlar için AIR ile Flash içeriğinin GPU hızlandırmasını desteklemek üzere Adobe, renderMode="gpu" yerine renderMode="direct" (yani Stage3D) kullanmanızı önerir. Adobe resmi olarak şu Stage3D tabanlı çerçeveleri desteklemekte ve önermektedir: Starling (2D) ve Away3D (3D). Stage3D ve Starling/Away3D hakkında daha fazla ayrıntı için bkz. http://gaming.adobe.com/getstarted/.*
- `gpu` — varsa donanım hızlandırma kullanılır.

  Önemli: *Flex uygulamaları için GPU görüntü oluşturma modunu kullanmayın.*

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

requestedDisplayResolution

Adobe AIR 2.6 ve üstü, yalnızca iOS; Adobe AIR 3.6 ve üstü, OS X — İsteğe bağlı

Uygulamanın yüksek çözünürlüklü ekrana sahip bir aygıtta veya bilgisayar monitöründe standart çözünürlüğü mü yoksa yüksek çözünürlüğü mü kullanmak istediğini belirtir. Varsayılan olan *standart* olarak ayarlandığında, ekran uygulamaya standart çözünürlüklü bir ekran olarak görünür. *Yüksek* olarak ayarlandığında, uygulama her yüksek çözünürlüklü piksele erişebilir.

Örneğin, 640x960 yüksek çözünürlüklü iPhone ekranında, ayar *standart* ise tam ekran sahne alanı boyutları 320x480 olur ve her bir uygulama pikseli, dört ekran pikseli kullanılarak oluşturulur. Ayar *yüksek* ise, tam ekran sahne alanı boyutları 640x960 olur.

Standart çözünürlüklü ekranlara sahip aygıtlarda, sahne alanı boyutları hangi ayarın kullanıldığına bakılmaksızın ekran boyutlarıyla eşleşir.

`requestedDisplayResolution` öğesi, `iPhone` öğesinde yuvalanmışsa, iOS aygıtlarında geçerli olur. Bu durumda `excludeDevices` niteliği, ayarın geçerli olmadığı aygıtları belirtmek için kullanılabilir.

`requestedDisplayResolution` öğesi, `initialWindow` öğesinde yuvalanmışsa, yüksek çözünürlüklü ekranları destekleyen MacBook Pro bilgisayarlardaki masaüstü AIR uygulamaları için geçerli olur. Belirtilen değer, uygulamada kullanılan tüm yerel pencereler için geçerlidir. `requestedDisplayResolution` öğesinin `initialWindow` öğesine yuvalanması AIR 3.6 ve üst sürümlerinde desteklenir.

Üst öğe:“iPhone” sayfa 227, “initialWindow” sayfa 225

Alt öğeler: yok

İçerik
Varsayılan olan *standart* veya *yüksek*.

Nitelik:
excludeDevices — iOS model adlarının veya model adı öneklerinin boşlukla ayrılmış bir listesi. Bu, geliştiricinin bazı aygıtların yüksek çözünürlük kullanmasını, bazılarının ise standart çözünürlük kullanmasını sağlamasına olanak tanır. Bu nitelik yalnızca iOS'ta kullanılabilir (`requestedDisplayResolution` öğesi `iPhone` öğesine yuvalanır). `excludeDevices` niteliği AIR 3.6 ve üst sürümlerinde kullanılabilir.

Model adı bu nitelikte belirtilen aygıtlar için, `requestedDisplayResolution` değeri belirtilen değerin karşıtıdır. Yani `requestedDisplayResolution` değeri *yüksek* ise, hariç tutulan aygıtlar standart çözünürlüğü kullanır. `requestedDisplayResolution` değeri *standart* ise, hariç tutulan aygıtlar yüksek çözünürlük kullanır.

Değerler, iOS aygıt modeli adları veya model adı önekleridir. Örneğin, iPad3,1 değeri özellikle Wi-Fi 3. nesil iPad'i (ancak GSM veya CDMA 3. nesil iPad'leri değil) işaret eder. Alternatif olarak, iPad3 değeri herhangi bir 3. nesil iPad'i işaret eder. iOS model adlarının resmi olmayan bir listesi iPhone wiki Modeller sayfasında mevcuttur.

Örnekler
Masaüstü:

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

iOS:

```
<iPhone>
    <requestedDisplayResolution excludeDevices="iPad3
iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Pencerenin yeniden boyutlandırılıp boyutlandırılamayacağını belirtir.

***Not:** Mac OS X gibi, pencereleri büyütmenin yeniden boyutlandırma işlemi olduğu işletim sistemlerinde, pencerenin yakınlaştırılmasını ve yeniden boyutlandırılmasını önlemek için hem maximizable hem de resizable `false` olarak ayarlı olmalıdır.*

Üst öğe:“initialWindow” sayfa 225

Alt öğeler:

İçerik
`true` (varsayılan) veya `false`

Örnek

```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 ve üstü, mobil profil — İsteğe bağlı

Sanal bir klavye görüntülendiğinde uygulamanın varsayılan davranışını belirtir. Varsayılan davranış uygulamayı yukarı kaydırmaktır. Çalışma zamanı, odaklanılmış metin alanını veya etkileşimli nesneyi ekranda tutar. Uygulamanız kendi klavye işleme mantığını sağlamıyorsa *pan* seçeneğini kullanın.

Ayrıca `softKeyboardBehavior` öğesini *none* olarak ayarlayarak otomatik davranışı kapatabilirsiniz. Bu durumda, metin alanları ve etkileşimli nesneler yazılım klavyesi kaldırıldığında bir SoftKeyboardEvent öğesi gönderir ancak çalışma zamanı uygulamayı kaydırmaz veya yeniden boyutlandırmaz. Metin giriş alanını görünümde tutmak uygulamanızın sorumluluğundadır.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

none veya *pan*. Varsayılan değer *pan* değeridir.

Örnek

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Daha fazla Yardım konusu

SoftKeyboardEvent

supportedLanguages

Adobe AIR 3.2 ve üstü - İsteğe bağlı

Uygulama tarafından desteklenen dilleri tanımlar. Bu öğe yalnızca iOS, Mac sabit çalışma zamanı ve Android uygulamaları tarafından kullanılır. Bu öğe diğer tüm uygulama türleri tarafından yoksayılır.

Bu öğeyi belirtmezseniz, varsayılan olarak paketleyici, uygulama türüne bağlı olarak aşağıdaki eylemleri gerçekleştirir:

- iOS — AIR çalışma zamanı tarafından desteklenen tüm diller iOS app store'da uygulamanın desteklediği diller olarak listelenir.
- Mac sabit çalışma zamanı — Sabit paketle paketlenmiş olan uygulamada yerelleştirme bilgisi yoktur.
- Android — Uygulama paketinde AIR çalışma zamanı tarafından desteklenen tüm dillere yönelik kaynaklar bulunur.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik

Desteklenen dillerin boşlukla ayrılmış bir listesi. Geçerli dil değerleri, AIR çalışma zamanı tarafından desteklenen dillere yönelik ISO 639-1 değerleridir: `en`, `de`, `es`, `fr`, `it`, `ja`, `ko`, `pt`, `ru`, `cs`, `nl`, `pl`, `sv`, `tr`, `zh`, `da`, `nb`, `iw`.

Paketleyici, `<supportedLanguages>` öğesinin boş bir değeri için hata oluşturur.

Not: *Yerelleştirilmiş etiketler (ad etiketi gibi), `<supportedLanguages>` etiketini kullanmanız durumunda dil değerini yoksayar ve söz konusu dili içermez. Yerel bir uzantıda `<supportedLangauges>` etiketi tarafından belirtilmeyen bir dile yönelik kaynaklar bulunuyorsa, bir uyarı bildirilir ve söz konusu dile yönelik kaynaklar yoksayılır.*

Örnek

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 veya üstü — İsteğe bağlı

Uygulama için desteklenen profilleri tanımlar.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik

Bu değerlerden herhangi birini `supportedProfiles` öğesine dahil edebilirsiniz:

- `desktop`: Masaüstü profili, AIR dosyası kullanılarak masaüstü bilgisayara yüklenen AIR uygulamaları içindir. Bu uygulamaların NativeProcess sınıfına erişimi yoktur (bu da yerel uygulamalarla iletişimi sağlar).
- `extendedDesktop`: Genişletilmiş masaüstü profili, yerel uygulama yükleyicisi kullanılarak bir masaüstü bilgisayara yüklenen AIR uygulamaları içindir. Bu uygulamaların NativeProcess sınıfına erişimi vardır (bu da yerel uygulamalarla iletişimi sağlar).
- `mobileDevice`—Mobil aygıt profili mobil uygulamalar içindir.
- `extendedMobileDevice`—Genişletilmiş mobil aygıt profili şu anda kullanım dışıdır.

`supportedProfiles` özelliği isteğe bağlıdır. Bu öğeyi uygulama tanımlayıcı dosyasına dahil etmediğinizde, uygulama herhangi bir profil için derlenebilir ve konuşlandırılabilir.

Çoklu profiller belirlemek için, her birini birer boşluk karakteriyle ayırın. Örneğin, aşağıdaki ayar uygulamanın yalnızca masaüstü ve genişletilmiş profillerde kullanılabilir olduğunu belirler:

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

Not: *ADL ile bir uygulama çalıştırdığınızda ve ADL `-profile` seçeneği için bir değer belirtmediğinizde, uygulama tanımlayıcısındaki ilk profil kullanılır. (Uygulama tanımlayıcıda da herhangi bir profil belirtilmemişse, masaüstü profili kullanılır.)*

Örnek

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

Daha fazla Yardım konusu

"Aygıt profilleri" sayfa 242

"Desteklenen profiller" sayfa 72

systemChrome

Adobe AIR 1.0 ve üstü — İsteğe bağlı

İlk uygulama penceresinin, işletim sistemi tarafından sağlanan standart başlık çubuğu, kenarlıklar ve denetimlerle oluşturulup oluşturulmadığını belirtir.

Pencerenin sistem kromu ayarı çalışma zamanında değiştirilemez.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

Aşağıdaki değerlerden biri:

- `none` — Sistem kromu sağlanmaz. Uygulama (veya Flex gibi bir uygulama çerçevesi) pencere kromunu görüntülemekten sorumludur.
- `standard` (varsayılan) — Sistem kromu işletim sistemi tarafından sağlanır.

Örnek

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 ve üstü — İsteğe bağlı

Yerelleştirilmiş bir dizeyi belirtir.

Bir metin öğesinin `xml:lang` niteliği, RFC4646'da (http://www.ietf.org/rfc/rfc4646.txt) tanımlandığı gibi bir dil kodu belirtir.

AIR uygulama yükleyicisi, `text` öğesini kullanıcının işletim sistemi kullanıcı arabirimi diliyle en yakından eşleşen `xml:lang` nitelik değeriyle birlikte kullanır.

Örneğin, `text` öğesinin en (İngilizce) yerel ayarı için değer içerdiği bir yükleme düşünün. İşletim sistemi en (English) adını kullanıcı arabirimi dili olarak tanımlıyorsa, AIR uygulama yükleyicisi en adını kullanır. Ayrıca sistem kullanıcı arabirimi adı en-US (U.S. English) olduğunda da en adını kullanır. Ancak kullanıcı arabirimi dili en-US ise ve uygulama tanımlayıcı dosyası hem en-US hem de en-GB adlarını tanımlıyorsa, AIR uygulama yükleyicisi en-US değerini kullanır.

Uygulama sistem kullanıcı arabirimi dilleriyle eşleşen bir `text` öğesi tanımlamıyorsa, AIR uygulama yükleyicisi uygulama tanımlayıcı dosyasında tanımlanan ilk `name` değerini kullanır.

Üst öğeler:

- "name" sayfa 230
- "description" sayfa 213

Alt öğeler: yok

İçerik

Bir yerel ayarı ve yerelleştirilmiş metin dizesini belirten `xml:lang` niteliği.

Örnek

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 ve üstü — İsteğe bağlı

İlk uygulama penceresinin başlık çubuğunda görüntülenen başlığı belirtir.

Başlık yalnızca `systemChrome` öğesi `standard` olarak ayarlanmışsa görüntülenir.

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

Pencere başlığını içeren bir dize.

Örnek

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 ve üstü — İsteğe bağlı

İlk uygulama penceresinin masaüstü ile alfa karışımlı olup olmadığını belirtir

Saydamlık özelliği etkin bir pencere daha yavaş çizebilir ve daha fazla bellek gerektirebilir. Saydamlık ayarı çalışma zamanında değiştirilemez.

Önemli: *`transparent` öğesini `true` olarak yalnızca `systemChrome`none olduğunda ayarlayabilirsiniz.*

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik

`true` veya `false` (varsayılan)

Örnek

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

version

Adobe AIR 1.0 ila 2.0 — Zorunlu; AIR 2.5 veya üstünde izin verilmez

Uygulamanın sürüm bilgilerini belirtir.

Sürüm dizesi, uygulama tanımlı bir belirleyicidir. AIR, sürüm dizesini hiçbir şekilde yorumlamaz. Bu nedenle "3.0" sürümünün "2.0" sürümünden daha güncel olduğu varsayılmaz. Örnekler: `"1.0"`, `".4"`, `"0.5"`, `"4.9"`, `"1.3.4a"`.

AIR 2.5 veya üstünde, `versionNumber` ve `versionLabel` öğeleri, `version` öğesinin yerine geçer.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik
Uygulama sürümünü içeren bir dize.

Örnek
```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 ve üstü — İsteğe bağlı

Kişiler tarafından okunabilen sürüm dizesini belirtir.

Yükleme iletişim kutularında `versionNumber` öğesinin değeri yerine sürüm etiketinin değeri görüntülenir. `versionLabel` kullanılmıyorsa ikisi için de `versionNumber` kullanılır.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik
Genel olarak görüntülenen sürüm metnini içeren bir dize.

Örnek
```
<versionLabel>0.9 Beta</versionlabel>
```

versionNumber

Adobe AIR 2.5 ve üstü — Zorunlu

Uygulama sürüm numarası.

Üst öğe:"application" sayfa 206

Alt öğeler: yok

İçerik
Sürüm numarası, noktalarla ayrılmış, en çok üç basamaklı tam sayılar sıralaması içerebilir. Her tamsayı 0 ile 999 (dahil) arasında bir sayı olmalıdır.

Örnekler
```
<versionNumber>1.0.657</versionNumber>

<versionNumber>10</versionNumber>

<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 ve üstü — İsteğe bağlı

İlk uygulama penceresinin oluşturulur oluşturulmaz görünür olup olmadığını belirtir.

İlk pencere de dahil olmak üzere AIR pencereleri varsayılan olarak görünmez durumda oluşturulur. NativeWindow nesnesinin `activate()` yöntemini çağırarak veya `visible` özelliğini `true` olarak ayarlayarak bir pencereyi görüntüleyebilirsiniz. Ana pencereyi başlangıçta gizli bırakmak isteyebilirsiniz, böylece pencerenin konumunda, boyutunda ve içeriğinin mizanpajında yapılan değişiklikler gösterilmez.

`visible` niteliği MXML tanımında `false` olarak ayarlı olmadığı sürece, Flex `mx:WindowedApplication` bileşeni `applicationComplete` olayı gönderilmeden hemen önce pencereyi otomatik olarak görüntüler ve etkinleştirir.

Pencereleri desteklemeyen mobil profilindeki aygıtlarda görünürlük ayarı yoksayılır.

Üst öğe:“initialWindow” sayfa 225

Alt öğeler: yok

İçerik

`true` veya `false` (varsayılan)

Örnek

```
<visible>true</visible>
```

width

Adobe AIR 1.0 ve üstü — İsteğe bağlı

Uygulamanın ana penceresinin başlangıçtaki genişliği.

Bir genişlik ayarlamazsanız, genişlik kök SWF dosyasındaki ayarlar tarafından veya HTML tabanlı bir AIR uygulaması olması durumunda işletim sistemi tarafından belirlenir.

AIR 2'de bir pencerenin maksimum genişliği 2048 pikselden 4096 piksele değişmiştir.

Üst öğe:“initialWindow” sayfa 225

Alt öğeler: yok

İçerik

Maksimum 4095 değerine sahip pozitif bir tamsayı.

Örnek

```
<width>1024</width>
```

x

Adobe AIR 1.0 ve üstü — İsteğe bağlı

İlk uygulama penceresinin yatay konumu.

Çoğu durumda, sabit bir değer atamak yerine işletim sisteminin pencerenin ilk konumunu belirlemesine izin vermek daha iyidir.

Ekran koordinatı sisteminin başlangıç noktası (0,0) ana masaüstü ekranının üst kısmında, sol taraftaki köşesidir (işletim sistemi tarafından belirlendiği şekilde).

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik
Bir tamsayı değeri.

Örnek

```
<x>120</x>
```

y

Adobe AIR 1.0 ve üstü — İsteğe bağlı

İlk uygulama penceresinin dikey konumu.

Çoğu durumda, sabit bir değer atamak yerine işletim sisteminin pencerenin ilk konumunu belirlemesine izin vermek daha iyidir.

Ekran koordinatı sisteminin başlangıç noktası (0,0) ana masaüstü ekranının üst kısmında, sol taraftaki köşesidir (işletim sistemi tarafından belirlendiği şekilde).

Üst öğe:"initialWindow" sayfa 225

Alt öğeler: yok

İçerik
Bir tamsayı değeri.

Örnek

```
<y>250</y>
```

Bölüm 15: Aygıt profilleri

Adobe AIR 2 ve üstü

Profiller, uygulamanızın çalıştığı bilgi işlem aygıtlarının sınıflarını tanımlayan bir mekanizmadır. Profil, belirli bir aygıt sınıfında genellikle desteklenen API gruplarını ve özelliklerini tanımlar. Kullanılabilir profiller şunları içerir:

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Uygulamanız için uygulama tanımlayıcısında profil tanımlayabilirsiniz. Dahil olan profillerdeki bilgisayarların ve aygıtların kullanıcıları uygulamanızı yükleyebilir. Diğer bilgisayar ve aygıtların kullanıcıları yükleyemez. Örneğin, uygulama tanımlayıcınıza yalnızca masaüstü profilini dahil ederseniz, kullanıcılar yalnızca masaüstü bilgisayarlarda uygulamanızı yükleyip çalıştırabilir.

Uygulamanızın gerçekten desteklemediği bir profil dahil ederseniz, bu tür ortamlardaki kullanıcı deneyimi kötü olabilir. Uygulama tanımlayıcıda herhangi bir profil belirtmezseniz, AIR uygulamanızı sınırlandırmaz. Uygulamanızı desteklenen biçimlerden herhangi biriyle paketleyebilirsiniz ve herhangi bir profile sahip kullanıcı uygulamayı yükleyebilir, ancak uygulama çalışma zamanında düzgün çalışmayabilir.

Mümkün olduğu yerlerde, uygulamanızı paketlerken profil sınırlamaları zorla uygulanır. Örneğin, yalnızca extendedDesktop profilini dahil ederseniz, uygulamanızı AIR dosyası olarak paketleyemezsiniz. Yalnızca yerel yükleyici olarak paketleyebilirsiniz. Aynı şekilde, mobileDevice profilini dahil etmezseniz, uygulamanızı Android APK olarak paketleyemezsiniz.

Tek bir bilgi işlem aygıtı birden fazla profili destekleyebilir. Örneğin, masaüstü bilgisayarlardaki AIR hem masaüstü hem de extendedDesktop profillerinin uygulamalarını destekler. Ancak, genişletilmiş masaüstü profili uygulaması yerel işlemlerle iletişim kurabilir ve yerel yükleyici (exe, dmg, deb veya rpm) olarak PAKETLENMELİDİR. Diğer tarafta bir masaüstü profili uygulaması yerel bir işlemle iletişim kuramaz. Bir masaüstü profili uygulaması AIR dosyası veya yerel yükleyici olarak paketlenebilir.

Profile bir özelliğin dahil edilmesi, profilin tanımlandığı aygıt sınıfı için bu özelliğin yaygın olarak desteklendiğini gösterir. Ancak bu, bir profildeki her aygıtın her özelliği desteklediği anlamına gelmez. Örneğin, tümü olmasa da bir çok mobil telefon bir akselerometre içerir. Evrensel desteğe sahip olmayan sınıflar ve özellikler genelde özelliği kullanmadan önce denetleyebileceğiniz bir boolean özelliğine sahiptir. Örneğin akselerometre durumunda, geçerli aygıtın desteklenen bir akselerometreye sahip olup olmadığını belirlemek için statik `Accelerometer.isSupported` özelliğini test edebilirsiniz.

Aşağıdaki profiller uygulama tanımlayıcısındaki supportedProfiles öğesi kullanılarak AIR uygulamanıza atanabilir:

Masaüstü Masaüstü profili, bir masaüstü bilgisayarda AIR dosyaları olarak yüklenen AIR uygulamaları için bir takım özellikler tanımlar. Bu uygulamalar desteklenen masaüstü platformlarında (Mac OS, Windows ve Linux) yüklenir ve çalışır. AIR 2'den önceki AIR sürümlerinde geliştirilen AIR uygulamaları masaüstü profilinde sayılabilir. Bazı API'ler bu profilde işlev görmez. Örneğin, masaüstü uygulamaları yerel işlemlerle iletişim kuramaz.

Genişletilmiş masaüstü Genişletilmiş masaüstü profili bir yerel yükleyiciye paketlenip onunla yüklenen AIR uygulamaları için bir takım özellikler tanımlar. Bu yerel yükleyiciler Windows'ta EXE dosyaları, Mac OS'de DMG dosyaları ve Linux'ta BIN, DEB veya RPM dosyalarıdır. Genişletilmiş masaüstü uygulamaları masaüstü profili

uygulamalarında olmayan ek özelliklere sahiptir. Daha fazla bilgi için bkz. “Masaüstü yerel yükleyicisini paketleme” sayfa 54.

Mobil aygıt Mobil aygıt profili, cep telefonu ve tablet gibi mobil aygıtlara yüklenen uygulamalar için bir takım özellikler tanımlar. Bu uygulamalar, Android, Blackberry Tablet OS ve iOS gibi desteklenen mobil platformlarda desteklenir.

Genişletilmiş mobil aygıt Genişletilmiş mobil aygıt profili, mobil aygıtlarda yüklü olan uygulamalar için bir takım özellikler tanımlar. Şu anda, bu profili destekleyen aygıt bulunmamaktadır.

Uygulama tanımlayıcı dosyasında hedef profillerinin kısıtlanması

Adobe AIR 2 ve üstü

AIR 2'den itibaren, uygulama tanımlayıcı dosyası hedef profilleri kısıtlamanıza izin veren bir `supportedProfiles` öğesi içerir. Örneğin, aşağıdaki ayar uygulamanın yalnızca masaüstü profillerinde kullanılabilir olduğunu belirler:

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

Bu öğe ayarlandığında, uygulama yalnızca sizin listelediğiniz profillerde paketlenebilir. Aşağıdaki değerleri kullanın:

- `desktop`—Masaüstü profili
- `extendedDesktop`—Genişletilmiş masaüstü profili
- `mobileDevice`—Mobil aygıt profili

`supportedProfiles` öğesi isteğe bağlıdır. Bu öğeyi uygulama tanımlayıcı dosyasına dahil etmediğinizde, uygulama herhangi bir profil için paketlenebilir ve dağıtılabilir.

`supportedProfiles` öğesinde birden fazla profil belirtmek için, her birini aşağıda gösterildiği gibi bir boşluk karakteriyle bölün:

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

Farklı profillerin yetenekleri

Adobe AIR 2 ve üstü

Aşağıdaki tablo tüm profillerde desteklenmeyen sınıfları ve özellikleri listeler.

Sınıf veya Özellik	desktop	extendedDesktop	mobileDevice
Akselerometre (Accelerometer.isSupported)	Hayır	Hayır	Kontrol edin
Erişilebilirlik (Capabilities.hasAccessibility)	Evet	Evet	Hayır
Akustik yankı iptali (Microphone.getEnhancedMicrophone())	Evet	Evet	Hayır
ActionScript 2	Evet	Evet	Hayır
CacheAsBitmap matrisi	Hayır	Hayır	Evet

Sınıf veya Özellik	**desktop**	**extendedDesktop**	**mobileDevice**
Camera (Camera.isSupported)	Evet	Evet	Evet
CameraRoll	Hayır	Hayır	Evet
CameraUI (CameraUI.isSupported)	Hayır	Hayır	Evet
Sabit çalışma zamanı paketleri	Evet	Evet	Evet
ContextMenu (ContextMenu.isSupported)	Evet	Evet	Hayır
DatagramSocket (DatagramSocket.isSupported)	Evet	Evet	Evet
DockIcon (NativeApplication.supportsDockIcon)	Kontrol edin	Kontrol edin	Hayır
Sürükle bırak (NativeDragManager.isSupported)	Evet	Evet	Kontrol edin
EncyptedLocalStore (EncyptedLocalStore.isSupported)	Evet	Evet	Evet
Flash Erişimi (DRMManager.isSupported)	Evet	Evet	Hayır
GameInput (GameInput.isSupported)	Hayır	Hayır	Hayır
Geolocation (Geolocation.isSupported)	Hayır	Hayır	Kontrol edin
HTMLLoader (HTMLLoader.isSupported)	Evet	Evet	Hayır
IME (IME.isSupported)	Evet	Evet	Kontrol edin
LocalConnection (LocalConnection.isSupported)	Evet	Evet	Hayır
Microphone (Microphone.isSupported)	Evet	Evet	Kontrol edin
Çok kanallı ses (Capabilities.hasMultiChannelAudio())	Hayır	Hayır	Hayır
Yerel Uzantılar	Hayır	Evet	Evet
NativeMenu (NativeMenu.isSupported)	Evet	Evet	Hayır
NativeProcess (NativeProcess.isSupported)	Hayır	Evet	Hayır
NativeWindow (NativeWindow.isSupported)	Evet	Evet	Hayır
NetworkInfo (NetworkInfo.isSupported)	Evet	Evet	Kontrol edin
Dosyaları varsayılan uygulamayla açma	Sınırlı	Evet	Hayır
PrintJob (PrintJob.isSupported	Evet	Evet	Hayır
SecureSocket (SecureSocket.isSupported)	Evet	Evet	Kontrol edin
ServerSocket (ServerSocket.isSupported)	Evet	Evet	Evet
Shader	Evet	Evet	Sınırlı
Stage3D (Stage.stage3Ds.length)	Evet	Evet	Evet
Sahne alanı yönlendirmesi (Stage.supportsOrientationChange)	Hayır	Hayır	Evet
StageVideo	Hayır	Hayır	Kontrol edin

Sınıf veya Özellik	desktop	extendedDesktop	mobileDevice
StageWebView (StageWebView.isSupported)	Evet	Evet	Evet
Oturum açıldığında uygulamayı başlatma (NativeApplication.supportsStartAtLogin)	Evet	Evet	Hayır
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Evet	Evet	Hayır
Sistem boşta modu	Hayır	Hayır	Evet
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Kontrol edin	Kontrol edin	Hayır
Metin Mizanpajı Çerçevesi girdisi	Evet	Evet	Hayır
Updater (Updater.isSupported)	Evet	Hayır	Hayır
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Evet	Evet	Hayır

Tablodaki girişler şu anlamlara sahiptir:

- *Kontrol edin* — Özellik profildeki bazı aygıtlar tarafından desteklenir, ancak hepsi tarafından desteklenmez. Uygulamanın desteklenip desteklenmediğini kullanmadan önce çalışma zamanında kontrol edebilirsiniz.
- *Sınırlı* — Özellik desteklenir ancak önemli sınırlamalara sahiptir. Daha fazla bilgi için ilgili belgelere bakın.
- *Hayır* — Özellik profilde desteklenmez.
- *Evet* — Özellik profilde desteklenir. Bireysel bilgi işlem aygıtlarında bir özellik için gerekli olan donanımın olmayabileceğini unutmayın. Örneğin, tüm telefonlar kameraya sahip değildir.

ADL ile hata ayıklarken profilleri belirleme

Adobe AIR 2 ve üstü

ADL, uygulama açıklayıcı dosyasının `supportedProfiles` öğesinde desteklenen profilleri belirtip belirtmediğinizi kontrol eder. Belirtmeniz durumunda, ADL varsayılan olarak, hata ayıklama yaparken profil olarak listelenen ilk desteklenen profili kullanır.

`-profile` komut satırı argümanını kullanarak ADL hata ayıklama oturumu için bir profil belirtebilirsiniz. (Bkz. "AIR Hata Ayıklama Başlatıcısı (ADL)" sayfa 157.) Bu argümanı uygulama açıklayıcı dosyasının `supportedProfiles` öğesinde bir profil belirleseniz de belirlemeseniz de kullanabilirsiniz. Ancak, bir `supportedProfiles` öğesi belirlerseniz, bu öğe komut satırında belirttiğiniz profili içermelidir. Aksi takdirde, ADL hata üretir.

Bölüm 16: AIR.SWF tarayıcı içi API'si

Kesintisiz yükleme badge.swf dosyası özelleştirme

SDK'yle beraber sağlanan badge.swf dosyasını kullanmaya ek olarak, bir tarayıcı sayfasında kullanmak için kendi SWF dosyanızı oluşturabilirsiniz. Özel SWF dosyanız çalışma zamanıyla şu şekillerde etkileşimde bulunabilir:

- Bir AIR uygulaması yükleyebilir. Bkz. “Tarayıcıdan AIR uygulaması yükleme” sayfa 252.
- Belirli bir AIR uygulamasının yüklenip yüklenmediğine bakmak için kontrol edebilir. Bkz. “AIR uygulamasının yüklenip yüklenmediğini bir web sayfasından kontrol etme” sayfa 251.
- Çalışma zamanının yüklenip yüklenmediğine bakmak için kontrol edebilir. Bkz. “Çalışma zamanının yüklenip yüklenmediğini kontrol etme” sayfa 250.
- Yüklenmiş bir AIR uygulamasını kullanıcının sisteminde başlatabilir. Bkz. “Yüklenmiş bir AIR uygulamasını tarayıcıdan başlatma” sayfa 252.

Bu özelliklerin tümü, adobe.com adresinde bulunan bir SWF dosyasındaki API'ler çağrılarak sağlanır: air.swf. badge.swf dosyasını özelleştirebilir ve kendi SWF dosyanızdan air.swf API'lerini çağırabilirsiniz.

Buna ek olarak tarayıcıda çalışan bir SWF dosyası, LocalConnection sınıfını kullanarak, çalışan bir AIR uygulamasıyla iletişim kurabilir. Daha fazla bilgi için bkz. Diğer Flash Player ve AIR örnekleri ile iletişim (ActionScript geliştiricileri için) veya Diğer Flash Player ve AIR örnekleri ile iletişim (HTML geliştiricileri için).

Önemli: *Bu bölümde açıklanan özellikler (ve air.swf dosyasındaki API'ler), son kullanıcının Windows veya Mac OS'deki web tarayıcısında Adobe® Flash® Player 9 güncellemesi 3 veya daha üstünün yüklemiş olmasını gerektirir. Kesintisiz yükleme özelliği Linux'ta Flash Player 10'u (10,0,12,36 veya daha yeni bir sürüm) gerektirir. Flash Player'ın yüklü sürümünü kontrol etmek için kod yazabilir ve Flash Player'ın gerekli sürümü yüklü değilse, kullanıcıya alternatif bir arabirim sağlayabilirsiniz. Örneğin Flash Player'ın daha eski bir sürümü yüklüyse, AIR dosyasının yükleme sürümüne bir bağlantı verebilirsiniz (bir uygulama yüklemek için badge.swf dosyasını veya air.swf API'sini kullanmak yerine).*

AIR uygulamasını yüklemek için badge.swf dosyasını kullanma

AIR SDK ve Flex SDK'de bir badge.swf dosyası mevcuttur, bu dosya sayesinde kesintisiz yükleme özelliğini kolaylıkla kullanabilirsiniz. badge.swf dosyası, çalışma zamanını ve AIR uygulamasını web sayfasındaki bir bağdan yükleyebilir. badge.swf dosyası ve kaynak kodu, web sitenizde yayınlamanız için size sağlanır.

badge.swf dosyasını bir web sayfasına gömme

1 AIR SDK ve Flex SDK'nin samples/badge dizininde sağlanan aşağıdaki dosyaları bulun veya web sunucunuza ekleyin.
   - badge.swf
   - default_badge.html
   - AC_RunActiveContent.js
2 default_badge.html sayfasını metin düzenleyicisinde açın.

3 default_badge.html sayfasında, `AC_FL_RunContent()` JavaScript işlevinde, aşağıdaki öğeler için `FlashVars` parametre tanımlarını ayarlayın:

Parametre	Açıklama
`appname`	Çalışma zamanı yüklü olmadığında SWF dosyası tarafından görüntülenen uygulama adı.
`appurl`	(Zorunlu). Yüklenecek AIR dosyasının URL'si. Göreceli değil, mutlak bir URL kullanmalısınız.
`airversion`	(Zorunlu). Çalışma zamanının 1.0 sürümü için bunu 1.0'a ayarlayın.
`imageurl`	Kimlik kartında görüntülenecek görüntünün (isteğe bağlı) URL'si.
`buttoncolor`	Yükleme düğmesinin rengi. (`FFCC00` gibi, onaltılık bir değer olarak belirtilir.)
`messagecolor`	Çalışma zamanı yüklü olmadığında düğmenin altında görüntülenen metin mesajının rengi. (`FFCC00` gibi, onaltılık bir değer olarak belirtilir.)

4 badge.swf dosyasının minimum boyutu 217 piksel genişlik, 180 piksel yükseklikliktir. `AC_FL_RunContent()` işlevinin `width` ve `height` parametrelerinin değerlerini ihtiyaçlarınızı karşılayacak şekilde ayarlayın.

5 İhtiyaçlarınızı karşılayacak şekilde default_badge.html dosyasını yeniden adlandırın ve kodunu ayarlayın (veya başka bir HTML sayfasına dahil edin).

Not: *badge.swf dosyasını yükleyen HTML `embed` etiketi için, `wmode` niteliğini ayarlamayın; onu `"Window"` varsayılanına ayarlı bırakın. Diğer `wmode` ayarları bazı sistemlerde yüklemeyi engeller. Ayrıca, diğer `wmode` ayarları bir hata oluşturur: "Hata #2044: İşlenmemiş ErrorEvent:. text=Hata #2074: Sahne alanı, indirme arabirimine sığamayacak kadar küçük."*

Ayrıca badge.swf dosyasını düzenleyebilir ve yeniden derleyebilirsiniz. Ayrıntılar için bkz. "badge.swf dosyasını değiştirme" sayfa 248.

AIR uygulamasını bir web sayfasındaki kesintisiz yükleme bağlantısından yükleyin

Kesintisiz yükleme bağını bir sayfaya ekledikten sonra kullanıcı, SWF dosyasındaki bağı tıklatarak AIR uygulamasını yükleyebilir.

1 Flash Player'ın (Windows veya Mac OS X'te sürüm 9 güncelleme 3 veya daha yenisinin veya Linux'ta sürüm 10'un) yüklenmiş olduğu bir web tarayıcısında HTML sayfasına gidin.

2 Web sayfasında, badge.swf dosyasındaki bağı tıklatın.

- Çalışma zamanını yüklediyseniz, bir sonraki adıma geçin.
- Çalışma zamanını yüklemediyseniz, yüklemek isteyip istemediğinizi soran bir iletişim kutusu görüntülenir. Çalışma zamanını yükleyin (bkz. "Adobe AIR yüklemesi" sayfa 3) ve bir sonraki adımla devam edin.

3 Yükleme penceresinde, varsayılan ayarları seçili halde bırakın ve ardından Devam düğmesini tıklatın.

Windows kullanan bir bilgisayarda AIR otomatik olarak şunları gerçekleştirir:

- Uygulamayı c:\Program Files\ konumuna yükler
- Uygulama için bir masaüstü kısayolu oluşturur
- Başlat Menüsü kısayolu oluşturur.
- Program Ekle / Kaldır Denetim Masası'na uygulama için bir giriş ekler

Mac OS'de yükleyici uygulamayı Uygulamalar dizinine ekler (örneğin Mac OS'de /Applications dizininde).

Linux ile çalışan bir bilgisayarda AIR otomatik olarak şunu yapar:

- Uygulamayı /opt hedefine yükler.
- Uygulama için bir masaüstü kısayolu oluşturur
- Başlat Menüsü kısayolu oluşturur.
- Sistem paket yöneticisindeki uygulama için bir giriş ekler

4 İstediğiniz seçenekleri belirleyin, ardından Yükle düğmesini tıklatın.

5 Yükleme işlemi tamamlandığında, Son düğmesini tıklatın.

badge.swf dosyasını değiştirme

Flex SDK ve AIR SDK badge.swf dosyası için kaynak dosyaları sağlar. Bu dosyalar SDK'nin samples/badge klasöründe bulunur:

Kaynak dosyalar	Açıklama
badge.fla	badge.swf dosyasını derlemek için kullanılan kaynak Flash dosyası. badge.fla dosyası bir SWF 9 dosyasına (Flash Player'da yüklenebilir) derlenir.
AIRBadge.as	basdge.fla dosyasında kullanılan temel sınıfı tanımlayan bir ActionScript 3.0 sınıfı.

Flash Professional uygulamasını badge.fla dosyasının görsel arabirimini yeniden tasarlamak için kullanabilirsiniz.

AIRBadge sınıfında tanımlanan `AIRBadge()` yapıcı işlevi, http://airdownload.adobe.com/air/browserapi/air.swf adresinde barındırılan air.swf dosyasını yükler. air.swf dosyası, kesintisiz yükleme özelliğini kullanma için kod içerir.

air.swf dosyası başarıyla yüklendiğinde, `onInit()` yöntemi (AIRBadge sınıfında) çağrılır:

```
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;
    }
}
```

Kod, global `_air` değişkenini yüklenen air.swf dosyasının ana sınıfına ayarlar. Bu sınıf, badge.swf dosyasının kesintisiz yükleme işlevini çağırmak için eriştiği aşağıdaki genel yöntemleri içerir:

Yöntem	Açıklama
`getStatus()`	Çalışma zamanının bilgisayara yüklenip yüklenmediğini (veya yüklenip yüklenemeyebileceğini) belirler. Ayrıntılar için bkz. "Çalışma zamanının yüklenip yüklenmediğini kontrol etme" sayfa 250. • `runtimeVersion`—Yüklenecek uygulamanın gerektirdiği çalışma zamanını (`"1.0.M6"` gibi) belirten bir dize.
`installApplication()`	Kullanıcının bilgisayarında belirtilen uygulamayı yükler. Ayrıntılar için bkz. "Tarayıcıdan AIR uygulaması yükleme" sayfa 252. • `url`—URL'yi tanımlayan bir dize. Göreceli değil, mutlak bir URL yolu kullanmalısınız. • `runtimeVersion`—Yüklenecek uygulamanın gerektirdiği çalışma zamanını (`"2.5"` gibi) belirten bir dize. • `arguments`— Yüklemenin ardından başlatılırsa uygulamaya iletilecek olan argümanlar. Uygulama tanımlayıcı dosyasında `allowBrowserInvocation` öğesi `true` olarak ayarlanmışsa, yüklemenin ardından uygulama başlatılır. (Uygulama tanımlayıcı dosyası hakkında daha fazla bilgi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201). Uygulama, tarayıcıdan kesintisiz yüklemenin bir sonucu olarak başlatılırsa (kullanıcının yüklemeden sonra başlatmayı seçmesiyle), yalnızca argümanlar iletilirse uygulamanın NativeApplication nesnesi bir BrowserInvokeEvent nesnesi gönderir. Uygulamaya ilettiğiniz verilerin güvenlik sonuçlarını göz önünde bulundurun. Ayrıntılar için bkz. "Yüklenmiş bir AIR uygulamasını tarayıcıdan başlatma" sayfa 252.

`url` ve `runtimeVersion` için ayarlar, kap HTML sayfasındaki FlashVars ayarları kullanılarak SWF dosyasına iletilir.

Uygulama, yüklemeden sonra otomatik olarak başlarsa, yüklenen uygulamanın başlatmadan sonra badge.swf dosyasına başvurması için LocalConnection iletişimini kullanabilirsiniz. Daha fazla bilgi için bkz. Diğer Flash Player ve AIR örnekleri ile iletişim (ActionScript geliştiricileri için) veya Diğer Flash Player ve AIR örnekleri ile iletişim (HTML geliştiricileri için).

Ayrıca, bir uygulamanın yüklenip yüklenmediğini kontrol etmek için air.swf dosyasının `getApplicationVersion()` yöntemini de çağırabilirsiniz. Bu yöntemi uygulama yükleme işleminden önce veya yükleme başladıktan sonra çağırabilirsiniz. Ayrıntılar için bkz. "AIR uygulamasının yüklenip yüklenmediğini bir web sayfasından kontrol etme" sayfa 251.

air.swf dosyasını yükleme

Bir tarayıcıdaki web sayfasından çalışma zamanı ve AIR uygulamalarıyla etkileşimde bulunmak için air.swf dosyasındaki API'leri kullanan kendi SWF dosyanızı oluşturabilirsiniz. air.swf dosyası http://airdownload.adobe.com/air/browserapi/air.swf adresinde bulunmaktadır. air.swf API'lerine SWF dosyanızdan başvurmak için, air.swf dosyasını SWF dosyanızla aynı uygulama etki alanına yükleyin. Aşağıdaki kod, yükleme SWF dosyasının uygulama etki alanına air.swf dosyasını yüklemeye dair bir örnek gösterir:

```
 var airSWF:Object; // This is the reference to the main class of air.swf
var airSWFLoader:Loader = new Loader(); // Used to load the SWF
var loaderContext:LoaderContext = new LoaderContext();
                                // Used to set the application domain

loaderContext.applicationDomain = ApplicationDomain.currentDomain;

airSWFLoader.contentLoaderInfo.addEventListener(Event.INIT, onInit);
airSWFLoader.load(new URLRequest("http://airdownload.adobe.com/air/browserapi/air.swf"),
                    loaderContext);

function onInit(e:Event):void
{
    airSWF = e.target.content;
}
```

air.swf dosyası yüklendikten sonra (Loader nesnesinin `contentLoaderInfo` nesnesi `init` olayını gönderdiğinde), aşağıdaki bölümlerde anlatılan biçimde air.swf API'lerinden herhangi birini çağırabilirsiniz.

Not: *AIR SDK ve Flex SDK'yle sağlanan badge.swf dosyası, air.swf dosyasını otomatik olarak yükler. Bkz. "AIR uygulamasını yüklemek için badge.swf dosyasını kullanma" sayfa 246. Bu bölümdeki talimatlar, air.swf dosyasını yükleyen kendi SWF dosyanızı oluşturmaya yöneliktir.*

Çalışma zamanının yüklenip yüklenmediğini kontrol etme

SWF dosyası, http://airdownload.adobe.com/air/browserapi/air.swf adresinden yüklenen air.swf dosyasındaki `getStatus()` yöntemini çağırarak, çalışma zamanının yüklenip yüklenmediğini kontrol edebilir. Ayrıntılar için bkz. "air.swf dosyasını yükleme" sayfa 249.

air.swf dosyası yüklendikten sonra, SWF dosyası, aşağıda olduğu gibi air.swf dosyasının `getStatus()` yöntemini çağırabilir:

```
 var status:String = airSWF.getStatus();
```

`getStatus()` yöntemi, bilgisayardaki çalışma zamanının durumuna bağlı olarak aşağıdaki dize değerlerinden birini döndürür:

Dize değeri	Açıklama
`"available"`	Bu bilgisayara çalışma zamanı yüklenebilir, ancak şu anda yüklü değil.
`"unavailable"`	Bu bilgisayara çalışma zamanı yüklenemez.
`"installed"`	Bu bilgisayarda çalışma zamanı yüklü.

Gerekli Flash Player sürümü (Windows ve Mac OS'de sürüm 9 güncelleme 3 veya Linux'ta sürüm 10) tarayıcıya yüklenmediğinde `getStatus()` yöntemi bir hata atar.

AIR uygulamasının yüklenip yüklenmediğini bir web sayfasından kontrol etme

SWF dosyası, http://airdownload.adobe.com/air/browserapi/air.swf adresinden yüklenen air.swf dosyasındaki `getApplicationVersion()` yöntemini çağırarak AIR uygulamasının (eşleşen bir uygulama kimliği ve yayıncı kimliğiyle) yüklenip yüklenmediğini kontrol edebilir. Ayrıntılar için bkz. “air.swf dosyasını yükleme” sayfa 249.

air.swf dosyası yüklendikten sonra, SWF dosyası, aşağıda olduğu gibi air.swf dosyasının `getApplicationVersion()` yöntemini çağırır:

```
 var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
airSWF.getApplicationVersion(appID, pubID, versionDetectCallback);

function versionDetectCallback(version:String):void
{
    if (version == null)
    {
        trace("Not installed.");
        // Take appropriate actions. For instance, present the user with
        // an option to install the application.
    }
    else
    {
        trace("Version", version, "installed.");
        // Take appropriate actions. For instance, enable the
        // user interface to launch the application.
    }
}
```

`getApplicationVersion()` yöntemi aşağıdaki parametreleri içerir:

Parametreler	Açıklama
`appID`	Uygulama için uygulama kimliği. Ayrıntılar için bkz. “id” sayfa 222.
`pubID`	Uygulama için yayıncı kimliği. Ayrıntılar için bkz. “publisherID” sayfa 232. Söz konusu uygulama bir yayıncı kimliğine sahip değilse, `pubID` parametresini boş bir dizeye (“”) ayarlayın.
`callback`	İşleyici işlevi olarak hizmet görecek bir geri çağırma işlevi. getApplicationVersion() yöntemi senkronize çalışmaz ve yüklü sürümü (veya yüklü sürüm olmadığını) saptadıktan sonra, bu geri çağırma yöntemi çağrılır. Geri çağırma yöntemi tanımı bir parametre, yüklenmiş uygulamanın sürüm dizesine ayarlı bir dize içermelidir. Uygulama yüklenmemişse, önceki kod örneğinde gösterildiği gibi işleve bir null değeri iletilir.

Gerekli Flash Player sürümü (Windows ve Mac OS'de sürüm 9 güncelleme 3 veya daha yenisi veya Linux'ta sürüm 10) tarayıcıda yüklü değilse `getApplicationVersion()` yöntemi hata verir.

Not: *AIR 1.5.3'teki gibi, yayıncı kimliği uygun bulunmaz. Yayıncı kimlikleri artık otomatik olarak bir uygulamaya atanmamaktadır. Geriye doğru uyumluluk için, uygulamalar bir yayıncı kimliği belirlemeye devam edebilir.*

Tarayıcıdan AIR uygulaması yükleme

SWF dosyası, `http://airdownload.adobe.com/air/browserapi/air.swf` adresinden yüklenen air.swf dosyasındaki `installApplication()` yöntemini çağırarak bir AIR uygulaması yükleyebilir. Ayrıntılar için bkz. "air.swf dosyasını yükleme" sayfa 249.

air.swf dosyası yüklendikten sonra, SWF dosyası, aşağıdaki kodda olduğu gibi air.swf dosyasının `installApplication()` yöntemini çağırabilir:

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

`installApplication()` yöntemi, kullanıcının bilgisayarında belirtilen uygulamayı yükler. Bu yöntem aşağıdaki parametrelere sahiptir:

Parametre	Açıklama
`url`	Yüklenecek AIR dosyasının URL'sini tanımlayan dize. Göreceli değil, mutlak bir URL yolu kullanmalısınız.
`runtimeVersion`	Uygulama tarafından yüklenmesi gerekli kılınan çalışma zamanının sürümünü ("1.0" gibi) gösteren bir dize.
`arguments`	Yüklemenin ardından başlatılırsa uygulamaya iletilecek olan argüman dizisi. Argümanlarda yalnızca alfasayısal karakterler tanınır. Diğer değerleri aktarmanız gerektiğinde bir kodlama şeması kullanmayı deneyin. Uygulama tanımlayıcı dosyasında `allowBrowserInvocation` öğesi `true` olarak ayarlanmışsa, yüklemenin ardından uygulama başlatılır. (Uygulama tanımlayıcı dosyası hakkında daha fazla bilgi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201). Uygulama, tarayıcıdan kesintisiz yüklemenin bir sonucu olarak başlatılırsa (kullanıcının yüklemeden sonra başlatmayı seçmesiyle), yalnızca argümanlar iletilmişse uygulamanın NativeApplication nesnesi bir BrowserInvokeEvent nesnesi gönderir. Ayrıntılar için bkz. "Yüklenmiş bir AIR uygulamasını tarayıcıdan başlatma" sayfa 252.

`installApplication()` yöntemi yalnızca, fare tıklatması gibi bir kullanıcı olayına yönelik olay işleyicisinde çağrıldığında çalışabilir.

Gerekli Flash Player sürümü (Windows ve Mac OS'de sürüm 9 güncelleme 3 veya daha yenisi veya Linux'ta sürüm 10) tarayıcıda yüklü değilse `installApplication()` yöntemi hata verir.

Mac OS'de bir uygulamanın güncellenen sürümünü yüklemek için, kullanıcının uygulama dizinine yüklemek üzere yeterli sistem ayrıcalığına (ve uygulama çalışma zamanını güncelliyorsa yönetici ayrıcalıklarına) sahip olması gerekir. Windows'ta kullanıcı yönetici ayrıcalıklarına sahip olmalıdır.

Ayrıca, bir uygulamanın önceden yüklenip yüklenmediğini kontrol etmek için air.swf dosyasının `getApplicationVersion()` yöntemini de çağırabilirsiniz. Bu yöntemi uygulama yükleme işlemi başlamadan önce veya yükleme başladıktan sonra çağırabilirsiniz. Ayrıntılar için bkz. "AIR uygulamasının yüklenip yüklenmediğini bir web sayfasından kontrol etme" sayfa 251. Uygulama çalıştıktan sonra, LocalConnection sınıfını kullanarak tarayıcıdaki SWF içeriğiyle iletişim kurabilir. Daha fazla bilgi için bkz. Diğer Flash Player ve AIR örnekleri ile iletişim (ActionScript geliştiricileri için) veya Diğer Flash Player ve AIR örnekleri ile iletişim (HTML geliştiricileri için).

Yüklenmiş bir AIR uygulamasını tarayıcıdan başlatma

Tarayıcı başlatma özelliğini kullanmak için (tarayıcıdan başlatılmasına izin vererek), hedef uygulamanın uygulama tanımlayıcı dosyası aşağıdaki ayarı içermelidir:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Uygulama tanımlayıcı dosyası hakkında daha fazla bilgi için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201.

Tarayıcıdaki bir SWF dosyası, http://airdownload.adobe.com/air/browserapi/air.swf adresinden yüklenen air.swf dosyasındaki `launchApplication()` yöntemini çağırarak bir AIR uygulamasını başlatabilir. Ayrıntılar için bkz. "air.swf dosyasını yükleme" sayfa 249.

air.swf dosyası yüklendikten sonra, SWF dosyası, aşağıdaki kodda olduğu gibi air.swf dosyasının `launchApplication()` yöntemini çağırabilir:

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

`launchApplication()` yöntemi air.swf dosyasının (kullanıcı arabirimi SWF dosyasının uygulama etki alanında yüklüdür) üst düzeyinde tanımlanır. Bu yöntemi çağırmak AIR'in belirtilen uygulamayı başlatmasına neden olur (yüklüyse ve tarayıcı başlatması olanağı varsa, uygulama tanımlayıcı dosyasındaki `allowBrowserInvocation` ayarını kullanarak). Bu yöntem aşağıdaki parametrelere sahiptir:

Parametre	Açıklama
`appID`	Başlatılacak uygulamanın uygulama kimliği. Ayrıntılar için bkz. "id" sayfa 222.
`pubID`	Başlatılacak uygulamanın yayıncı kimliği. Ayrıntılar için bkz. "publisherID" sayfa 232. Söz konusu uygulama bir yayıncı kimliğine sahip değilse, `pubID` parametresini boş bir dizeye ("") ayarlayın
`arguments`	Uygulamaya iletilecek bir argüman dizisi. Uygulamanın NativeApplication nesnesi, bu diziye ayarlanmış bir özellikler kümesine sahip olan BrowserInvokeEvent olayını gönderir. Argümanlarda yalnızca alfasayısal karakterler tanınır. Diğer değerleri aktarmanız gerektiğinde bir kodlama şeması kullanmayı deneyin.

`launchApplication()` yöntemi yalnızca, fare tıklatması gibi bir user olayı için olay işleyicisinde çağrıldığında çalışabilir.

Gerekli Flash Player sürümü (Windows ve Mac OS'de sürüm 9 güncelleme 3 veya daha yenisi veya Linux'ta sürüm 10) tarayıcıda yüklü değilse `launchApplication()` yöntemi hata verir.

Uygulama tanımlayıcı dosyasında `allowBrowserInvocation` öğesi `false` olarak ayarlıysa, `launchApplication()` yöntemini çağırmanın hiçbir etkisi yoktur.

Kullanıcı arabirimini uygulamayı başlatmak için sunmadan önce, air.swf dosyasında `getApplicationVersion()` yöntemini çağırmak isteyebilirsiniz. Ayrıntılar için bkz. "AIR uygulamasının yüklenip yüklenmediğini bir web sayfasından kontrol etme" sayfa 251.

Uygulama tarayıcı başlatma özelliği üzerinden çağrıldığında, uygulamanın NativeApplication nesnesi bir BrowserInvokeEvent nesnesi gönderir. Ayrıntılar için bkz. Tarayıcıdan AIR uygulaması başlatma (ActionScript geliştiricileri için) veya Tarayıcıdan AIR uygulaması başlatma (HTML geliştiricileri için).

Tarayıcı başlatma özelliğini kullanırsanız, güvenlik sonuçlarını dikkate aldığınızdan emin olun. Bu sonuçlar, Tarayıcıdan AIR uygulaması başlatma (ActionScript geliştiricileri için) ve Tarayıcıdan AIR uygulaması başlatma (HTML geliştiricileri için) başlıklarında açıklanmaktadır.

Uygulama çalıştıktan sonra, LocalConnection sınıfını kullanarak tarayıcıdaki SWF içeriğiyle iletişim kurabilir. Daha fazla bilgi için bkz. Diğer Flash Player ve AIR örnekleri ile iletişim (ActionScript geliştiricileri için) veya Diğer Flash Player ve AIR örnekleri ile iletişim (HTML geliştiricileri için).

Not: *AIR 1.5.3'teki gibi, yayıncı kimliği uygun bulunmaz. Yayıncı kimlikleri artık otomatik olarak bir uygulamaya atanmamaktadır. Geriye doğru uyumluluk için, uygulamalar bir yayıncı kimliği belirlemeye devam edebilir.*

Bölüm 17: AIR uygulamalarını güncelleme

Kullanıcılar bilgisayarlarında veya tarayıcıdan (kesintisiz yükleme özelliğini kullanarak) bir AIR dosyasını çift tıklatarak AIR uygulamasını yükleyebilir veya güncelleyebilir. Adobe® AIR® yükleyici uygulaması, kullanıcıyı önceden var olan bir uygulamayı güncellemekte olduğuna ilişkin uyararak yüklemeyi yönetir.

Ancak, Updater sınıfını kullanarak kendini yeni sürüme güncelleyen yüklenmiş bir uygulamanız da olabilir. (Yüklenmiş bir uygulama, yeni bir sürümün indirilmek ve yüklenmek üzere kullanılabilir olduğunu saptayabilir.) Updater sınıfı, kullanıcının bilgisayarında yer alan bir AIR dosyasını göstermenize ve bu sürüme güncellemenizi sağlayan bir `update()` yöntemini içerir. Uygulamanız Updater sınıfını kullanmak için AIR dosyası olarak paketlenmelidir. Yerel yürütülebilir veya paket olarak paketlenmiş uygulamalar yerel platform tarafından sağlanan güncelleme olanaklarını kullanmalıdır.

Güncelleme AIR dosyasının hem uygulama kimliği hem de yayıncı kimliği, güncellenecek uygulamanınkilerle eşleşmelidir. Yayıncı kimliği imzalanan sertifikadan türetilir. Hem güncelleme hem de güncellenecek uygulama, aynı sertifikayla imzalanmış olmalıdır.

AIR 1.5.3 veya sonrası için, uygulama açıklayıcı dosyası bir `<publisherID>` öğesi içerir. Uygulamanızın AIR 1.5.2 veya öncesini kullanarak geliştirilen herhangi bir sürümü varsa bu öğeyi kullanmanız gerekir. Daha fazla bilgi için bkz. “publisherID” sayfa 232.

AIR 1.1 ve daha yeni sürümlerde, yeni bir kod imzalayıcı sertifika kullanmak için uygulamayı taşıyabilirsiniz. Yeni bir imza kullanması için bir uygulamayı taşıma, hem yeni hem de orijinal sertifikayı içeren güncelleme AIR dosyasını imzalamayı içerir. Sertifika taşıma, tek yönlü bir işlemdir. Taşımadan sonra, yalnızca yeni sertifikayla (veya her iki dosyayla) imzalanan AIR dosyaları var olan bir yüklemeye güncellemeler olarak dikkate alınır.

Uygulama güncellemelerinin yönetilmesi karmaşık olabilir. AIR 1.5, *Adobe® AIR uygulamaları* için yeni güncelleme çerçevesini içerir. Bu çerçeve, AIR uygulamalarında iyi güncelleme özellikleri sunmada geliştiricilere yardımcı olacak API'ler sağlar.

Kendinden imzalı sertifikayı ticari kod imzalayan sertifika olarak değiştirmek veya bir kendinden imzalı veya ticari sertifikadan diğerine geçmek için sertifika taşımayı kullanabilirsiniz. Sertifikayı taşımazsanız, var olan kullanıcılar yeni sürümü yüklemeden önce geçerli uygulama sürümlerini kaldırmak zorunda kalır. Daha fazla bilgi için bkz. “Sertifikaları değiştirme” sayfa 191.

Uygulamanızda bir güncelleme mekanizması kullanmakta fayda vardır. Uygulamanın yeni bir sürümünü oluşturursanız, güncelleme mekanizması kullanıcıya yeni sürümü yüklemek isteyip istemediğini sorabilir.

AIR uygulama yükleyicisi bir AIR uygulaması yüklendiğinde, güncellendiğinde veya kaldırıldığında günlük dosyaları oluşturur. Herhangi bir yükleme sorununun nedenini belirlemenize yardımcı olması için bu günlüklere başvurabilirsiniz. Bkz. Yükleme günlükleri.

Not: *Adobe AIR çalışma zamanının yeni sürümleri güncellenmiş WebKit sürümleri içerebilir. WebKit öğesinin güncellenmiş bir sürümü dağıtmak bir AIR uygulamasındaki HTML içeriğinde beklenmeyen değişikliklere neden olabilir. Bu değişiklikler sizden uygulamanızı güncellemenizi isteyebilir. Güncelleme mekanizması, uygulamanın yeni sürümünün kullanıcısını bilgilendirebilir. Daha fazla bilgi için bkz.HTML ortamı hakkında (ActionScript geliştiricileri için) veya HTML ortamı hakkında (HTML geliştiricileri için).*

Uygulamaları güncelleme hakkında

Updater sınıfı (flash.desktop paketinde), geçerli olarak çalışan uygulamayı farklı bir sürüme güncellemek için kullanabileceğiniz bir yöntemi, `update()` öğesini içerir. Örneğin, kullanıcının masaüstünde bir AIR dosyası sürümü ("Sample_App_v2.air") bulunuyorsa, aşağıdaki kod uygulamayı günceller.

ActionScript örneği:

```
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 örneği:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Bir uygulama Updater sınıfını kullanmadan önce kullanıcı veya uygulama AIR dosyasının güncellenmiş sürümünü bilgisayara indirmelidir. Daha fazla bilgi için bkz. “Kullanıcının bilgisayarına bir AIR dosyası indirme” sayfa 257.

Updater.update() yöntem çağrısının sonuçları

Çalışma zamanındaki bir uygulama `update()` yöntemini çağırdığında, çalışma zamanı uygulamayı kapatır ve daha sonra AIR dosyasından yeni sürümü yüklemeye çalışır. Çalışma zamanı, AIR dosyasında belirtilen uygulama kimliği ve yayıncı kimliğinin `update()` yöntemini çağıran uygulamaya ilişkin uygulama kimliği ve yayıncı kimliğiyle eşleşip eşleşmediğini kontrol eder. (Uygulama kimliği ve yayıncı kimliği hakkında bilgi için bkz. “AIR uygulama tanımlayıcı dosyaları” sayfa 201). Çalışma zamanı ayrıca, sürüm dizesinin `update()` yöntemine iletilen `version` dizesiyle eşleşip eşleşmediğini kontrol eder. Yükleme başarıyla tamamlanırsa, çalışma zamanı uygulamanın yeni sürümünü açar. Aksi halde (yükleme tamamlanamazsa), uygulamanın var olan (yükleme öncesi) sürümünü yeniden açar.

Mac OS'de uygulamanın güncellenmiş bir sürümünü yüklemek için, kullanıcının uygulama dizinine yüklemek için yeterli sistem ayrıcalıklarına sahip olması gerekir. Windows ve Linux'ta, kullanıcı yönetici ayrıcalıklarına sahip olmalıdır.

Uygulamanın güncellenmiş sürümü çalışma zamanının güncellenmiş sürümünü gerektiriyorsa, yeni çalışma zamanı sürümü yüklenir. Kullanıcı, çalışma zamanını güncellemek için bilgisayara ilişkin yönetici ayrıcalıklarına sahip olmalıdır.

ADL kullanarak uygulama test edilirken `update()` yönteminin çağrılması bir çalışma zamanı istisnasına neden olur.

Sürüm dizesi hakkında

`update()` yönteminin `version` parametresi olarak belirtilen dize, yüklenecek AIR dosyasına ilişkin uygulama tanımlayıcı dosyasında bulunan `version` veya `versionNumber` öğesindeki dizeyle eşleşmelidir. `Sürüm` parametresinin belirtilmesi, güvenlik nedenleri için gereklidir. AIR dosyasında sürüm sayısını doğrulamayı gerektirdiğinden, uygulaması kasıtsız olarak daha eski bir sürüm yüklemeyecektir. (Uygulamanın eski sürümü, şu an kurulu olan uygulamada giderilmiş olan bazı güvenlik açıkları içerebilir.) Uygulama, indirgeme saldırılarını engellemek için yüklenmiş uygulamadaki sürüm dizesiyle AIR dosyasındaki sürüm dizesinin eşleşip eşleşmediğini de kontrol etmelidir.

AIR 2.5'ten önceki sürümlerde sürüm dizesi herhangi bir biçimde olabilir. Örneğin, "2.01" veya "sürüm 2" olabilir. AIR 2.5 veya sonraki sürümlerde, sürüm dizesi noktalarla ayrılmış en fazla üç basamaktan oluşan sıralamalar olmalıdır. Örneğin, ".0", "1.0" ve "67.89.999" geçerli sürüm numaralarıdır. Uygulamayı güncellemeden önce güncelleme sürümü dizesini doğrulamanız gerekir.

Adobe AIR uygulaması web üzerinden bir AIR dosyası indirirse, web servisinin Adobe AIR uygulamasını indirilen sürüme ilişkin bilgilendirebileceği bir mekanizmaya sahip olmak, iyi bir uygulamadır. Bu şekilde, uygulama dizeyi `update()` yönteminin `version` parametresi olarak kullanabilir. AIR dosyası sürümünün bilinmediği diğer yöntemlerle elde edildiyse, AIR uygulaması sürüm bilgisini belirlemek için AIR dosyasını inceleyebilir. (AIR dosyası, sıkıştırılmış bir ZIP arşividir ve uygulama tanımlayıcı dosyası arşivdeki ikinci kayıttır.)

Uygulama tanımlayıcı dosyası ile ilgili ayrıntılar için bkz. "AIR uygulama tanımlayıcı dosyaları" sayfa 201.

Uygulama güncellemeleri için iş akışını imzalama

Güncellemeleri geçici bir şekilde yayınlama birden fazla uygulama sürümünü yönetme görevlerini karmaşıklaştırır ve sertifika sona erme tarihlerini izlemeyi zorlaştırır. Sertifikaların süresi siz bir güncelleme yayınlamadan önce sona erebilir.

Adobe AIR çalışma zamanı, geçiş imzası olmadan yayınlanmış uygulama güncellemelerini yeni bir uygulama olarak değerlendirir. Kullanıcılar uygulama güncellemesini yüklemeden önce geçerli AIR uygulamalarını kaldırmalıdır.

Sorunu çözmek için her güncellenen uygulamayı en güncel sertifikayla birlikte ayrı bir dağıtım URL'sine yükleyin. Sertifikanız 180 günlük yetkisiz kullanım süresindeyken geçiş imzası uygulamanızı hatırlatacak bir mekanizma dahil edin. Daha fazla bilgi için bkz. "Bir AIR uygulamasının güncellenmiş sürümünü imzalama" sayfa 195.

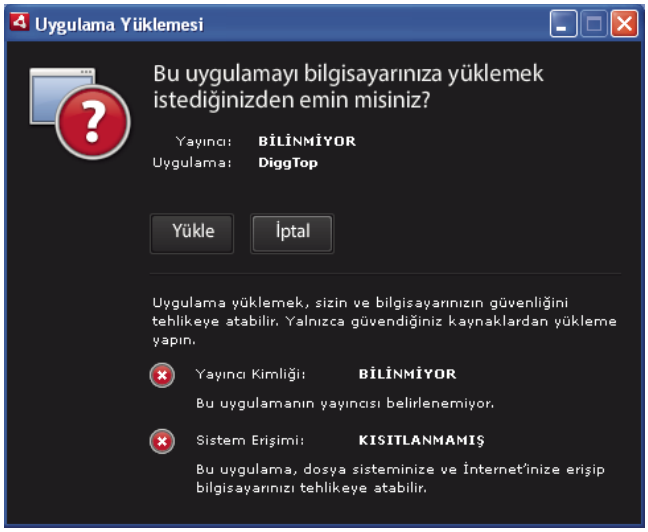
Nasıl imza uygulanacağıyla ilgili bilgi için bkz. "ADT komutları" sayfa 163.

Geçiş imzası uygulama sürecini kolaylaştırmak için aşağıdaki görevleri gerçekleştirin:

- Her güncellenmiş uygulamayı ayrı bir dağıtım URL'sine yükleyin.
- Yükseltme tanımlayıcı XML dosyasını ve güncellemenin en son sertifikasını aynı URL'ye yükleyin.
- Güncellenmiş uygulamayı en son sertifikayla imzalayın.
- Farklı bir URL'de bulunan önceki sürümü imzalamak için kullanılmış sertifikayla güncellenmiş uygulamaya bir geçiş imzası uygulayın.

Özel bir uygulama güncelleme kullanıcı arabirimi sunma

AIR, varsayılan bir güncelleme arabirimi içerir:



Bu arabirim, her zaman kullanıcı bir uygulama sürümünü bilgisayara ilk yüklediğinde kullanılır. Ancak, sonraki örneklerde kullanmak için kendi arabiriminizi tanımlayabilirsiniz. Uygulamanız özel bir güncelleme arabirimi tanımlıyorsa, güncel olarak yüklü uygulamaya ilişkin uygulama tanımlayıcı dosyasında bir `customUpdateUI` öğesi belirleyin:

```
<customUpdateUI>true</customUpdateUI>
```

Uygulama yüklendiğinde ve kullanıcı yüklenen uygulamayla eşleşen, uygulama kimliğine ve yayıncı kimliğine sahip bir AIR dosyasını açtığında, çalışma zamanı varsayılan AIR uygulama yükleyicisi yerine uygulamayı açar. Daha fazla bilgi için bkz. "customUpdateUI" sayfa 212.

Uygulama, çalıştırıldığında (`NativeApplication.nativeApplication` nesnesi bir `load` olayı gönderdiğinde), uygulamanın güncellenip güncellenmeyeceğine karar verebilir (Updater sınıfını kullanarak). Güncellemeye karar verirse, kullanıcıya kendi yükleme arabirimini (çalışan standart arabirimden farklı olan) sunabilir.

Kullanıcının bilgisayarına bir AIR dosyası indirme

Updater sınıfını kullanmak için, kullanıcı veya uygulama önce kullanıcının bilgisayarına bir AIR dosyasını yerel olarak kaydetmelidir.

Not: *AIR 1.5, geliştiricilere AIR uygulamalarında iyi güncelleme özellikleri sağlamada yardımcı olan bir güncelleme çerçevesi içerir. Bu çerçeveyi kullanmak, doğrudan Update sınıfının `update()` yöntemini kullanmaktan çok daha kolay olabilir. Ayrıntılar için bkz. "Güncelleme çerçevesini kullanma" sayfa 261.*

Aşağıdaki kod bir URL'den AIR dosyasını okur(http://example.com/air/updates/Sample_App_v2.air) ve AIR dosyasını uygulama depo dizinine kaydeder.

ActionScript örneği:

```
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 örneği:

```
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.");
}
```

Daha fazla bilgi için bkz.:

- Dosyaları okuma ve yazma iş akışı (ActionScript geliştiricileri için)
- Dosyaları okuma ve yazma iş akışı (HTML geliştiricileri için)

Uygulamanın ilk kez çalışıp çalışmadığını görmek için kontrol edin

Uygulamayı güncelledikten sonra kullanıcıya bir "başlangıç" veya "hoş geldiniz" mesajı sunmak isteyebilirsiniz. Başlatmadan sonra, uygulama mesajın görüntülenip görüntülenmeyeceğini belirlemek için ilk kez çalışıp çalışmadığını kontrol eder.

Not: *AIR 1.5, geliştiricilere AIR uygulamalarında iyi güncelleme özellikleri sağlamada yardımcı olan bir güncelleme çerçevesi içerir. Bu çerçeve, bir uygulama sürümünün ilk kez çalışıp çalışmadığını kontrol etmek için kolay yöntemler sağlar. Ayrıntılar için bkz. "Güncelleme çerçevesini kullanma" sayfa 261.*

Bunu yapmanın bir yolu, uygulamanın başlatılmasından sonra uygulama depo dizinine bir dosya kaydetmektir. Uygulama her başladığında, bu dosyanın mevcudiyetini kontrol etmelidir. Dosya mevcut değilse, uygulama geçerli kullanıcı için ilk kez çalışmaktadır. Dosya mevcutsa, uygulama önceden en az bir kez çalışmıştır. Dosya mevcutsa ve geçerli sürüm numarasından daha eski bir sürüm numarası içeriyorsa, kullanıcının yeni sürümü ilk kez çalıştığını bilirsiniz.

Aşağıdaki Flex örneği, bu kavramı gösterir:

```
<?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>
```

Aşağıdaki örnek kavramı JavaScript'te gösterir:

```
<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>
```

Uygulamanız verileri yerel olarak (örneğin, uygulama depo dizinine) kaydediyorsa, ilk çalıştırmadan sonra önceden kaydedilmiş veri (önceki sürümlerden) olup olmadığını kontrol etmek isteyebilirsiniz.

Güncelleme çerçevesini kullanma

Uygulama güncellemelerini yönetmek yorucu olabilir. *AdobeAIR uygulamaları için güncelleme çerçevesi*, geliştiricilerin AIR uygulamalarında sağlam güncelleme özellikleri sağlamasına izin veren API'ler sağlar. AIR güncelleme çerçevesi geliştiriciler için aşağıdaki görevleri gerçekleştirir:

- Bir aralığa bağlı olarak veya kullanıcı istediğinde periyodik olarak güncellemeleri denetler
- Web kaynağından AIR dosyalarını (güncellemeleri) indirir
- Yeni yüklenen sürümün ilk çalıştırılmasında kullanıcıyı uyarır
- Kullanıcının güncellemeleri denetlemek isteyip istemediğini teyit eder
- Yeni güncelleme sürümü hakkındaki bilgileri kullanıcıya gösterir
- İndirme ilerlemesi ve hata bilgisini kullanıcıya gösterir

AIR güncelleme çerçevesi, uygulamanız için örnek bir kullanıcı arabirimi sağlar. Kullanıcıya temel bilgileri ve uygulama güncellemelerine ilişkin konfigürasyon seçeneklerini sunar. Uygulamanız, güncelleme çerçevesiyle birlikte kullanmak üzere özel bir kullanıcı arabirimi de tanımlayabilir.

AIR güncelleme çerçevesi, AIR uygulamasının güncelleme sürümü hakkındaki bilgiyi basit XML yapılandırma dosyalarında saklamanıza olanak verir. Çoğu uygulama için, bazı basit kodları dahil etmek için bu konfigürasyon dosyalarını kurma, son kullanıcıya kullanışlı güncelleme işlevleri sağlar.

Adobe AIR, AIR uygulamalarının güncelleme çerçevesini kullanmadan bile yeni sürümlere yükseltilebileceği bir Updater sınıfı içerir. Updater sınıfı, bir uygulamanın kullanıcının bilgisayarındaki AIR dosyasında yer alan bir sürüme yükseltilmesine olanak verir. Ancak, yükseltme yönetimi yalnızca yerel olarak saklanan bir AIR dosyasına dayanan uygulama güncellemesini yapmaktan daha fazlasını içerebilir.

AIR güncelleme çerçevesi dosyaları

AIR güncelleme çerçevesi, AIR 2 SDK'in frameworks/libs/air dizinine dahil edilmiştir. Şu dosyaları içerir:

- applicationupdater.swc—ActionScript'te kullanılacak güncelleme kütüphanesinin temel işlevini tanımlar. Bu sürüme hiçbir kullanıcı arabirimi dahil edilmemiştir.
- applicationupdater.swf—JavaScript'te kullanılacak güncelleme kütüphanesinin temel işlevini tanımlar. Bu sürüme hiçbir kullanıcı arabirimi dahil edilmemiştir.
- applicationUpdater_ui.swc—Uygulamanızın güncelleme seçeneklerini görüntülemek için kullanabileceği bir kullanıcı arabirimi dahil olmak üzere, temel güncelleme kitaplığının Flex 4 sürümü işlevlerini tanımlar.
- applicationUpdater_ui.swf—Uygulamanızın güncelleme seçeneklerini görüntülemek için kullanabileceği bir kullanıcı arabirimi dahil olmak üzere, temel güncelleme kitaplığının JavaScript sürümü işlevlerini tanımlar.

Daha fazla bilgi için şu bölümlere bakın:

- "Flex geliştirme ortamınızı ayarlama" sayfa 262
- "Çerçeve dosyalarını HTML tabanlı bir AIR uygulamasına dahil etme" sayfa 262
- "Temel örnek: ApplicationUpdaterUI sürümünü kullanma" sayfa 263

Flex geliştirme ortamınızı ayarlama

AIR 2 SDK'deki frameworks/libs/air dizinindeki SWC dosyaları, Flex ve Flash geliştirmede kullanabileceğiniz sınıfları tanımlar.

Flex SDK ile derleme sırasında güncelleme çerçevesini kullanmak için ApplicationUpdater.swc veya ApplicationUpdater_UI.swc dosyasını amxmlc derleyicisine yapılan çağrıya dahil edin. Derleyici, aşağıdaki örnekte Flex SDK dizininin lib alt dizininde yer alan ApplicationUpdater.swc dosyasını yükler:

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

Derleyici, aşağıdaki örnekte Flex SDK dizininin lib alt dizininde yer alan ApplicationUpdater_UI.swc dosyasını yükler:

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Flash yapıcısı aracılığıyla geliştirme yaparken, SWC dosyasını Özellikler iletişim kutusundaki Flex Build Path ayarlarının Library Path sekmesine ekleyin.

SWC dosyalarını amxmlc derleyicisinde (Flex SDK kullanarak) veya Flash Builder'da başvuracağınız dizine kopyaladığınızdan emin olun.

Çerçeve dosyalarını HTML tabanlı bir AIR uygulamasına dahil etme

Güncelleme çerçevesinin frameworks/html dizini şu SWF dosyalarını içerir:

- applicationupdater.swf—Herhangi bir kullanıcı arabirimi olmadan güncelleme kütüphanesinin temel işlevlerini tanımlar
- applicationupdater_ui.swf—Uygulamanızın güncelleme seçeneklerini görüntülemek için kullanabileceği bir kullanıcı arabirimi dahil, temel güncelleme kütüphanesi işlevlerini tanımlar

AIR uygulamalarındaki JavaScript kodu, SWF dosyalarında tanımlanan sınıfları kullanabilir.

Güncelleme çerçevesini kullanmak için applicationupdater.swf veya applicationupdater_ui.swf dosyasını uygulama dizininize (veya bir alt dizine) dahil edin. Daha sonra, çerçeveyi kullanacak HTML dosyasına (JavaScript kodunda), dosyayı yükleyen bir `script` etiketini dahil edin:

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

Veya applicationupdater_ui.swf dosyasını yüklemek için bu `script` etiketini kullanın:

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

Bu iki dosyada tanımlanan API, bu belgenin devamında açıklanmıştır.

Temel örnek: ApplicationUpdaterUI sürümünü kullanma

Güncelleme çerçevesinin ApplicationUpdaterUI sürümü, uygulamanızda kolayca kullanabileceğiniz basit bir arabirim sunar. Aşağıdaki, temel bir örnektir.

Önce, güncelleme çerçevesini çağıran bir AIR uygulaması oluşturun:

1. Uygulamanız HTML tabanlı bir AIR uygulamasıysa, applicationupdaterui.swf dosyasını yükleyin:

   ```
   <script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
   ```

2. AIR uygulama program mantığınızda, bir ApplicationUpdaterUI nesnesini başlatın.

   ActionScript'te şu kodu kullanın:

   ```
   var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
   ```

   JavaScript'te şu kodu kullanın:

   ```
   var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
   ```

   Bu kodu uygulama yüklendiğinde yürütülen bir başlatma işlevine eklemek isteyebilirsiniz.

3. updateConfig.xml adlı bir metin dosyası oluşturun ve bu dosyaya aşağıdakini ekleyin:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
       <url>http://example.com/updates/update.xml</url>
       <delay>1</delay>
   </configuration>
   ```

   updateConfig.xml dosyasının `URL` öğesini web sunucunuzdaki güncelleme tanımlayıcı dosyasının son konumuyla eşleşecek biçimde düzenleyin. (Sonraki yordama bakın.)

   `delay`, uygulamanın kontroller ve güncellemeler arasında beklediği gün sayısıdır.

4. updateConfig.xml dosyasını AIR uygulamanızın proje dizinine ekleyin.

5. Updater nesnesinin updateConfig.xml dosyasına başvurmasını sağlayın ve nesnenin `initialize()` yöntemini çağırın.

   ActionScript'te şu kodu kullanın:

   ```
   appUpdater.configurationFile = new File("app:/updateConfig.xml");
   appUpdater.initialize();
   ```

   JavaScript'te şu kodu kullanın:

   ```
   appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
   appUpdater.initialize();
   ```

6. İlk uygulamadan farklı bir sürüme sahip olan AIR uygulamasının ikinci sürümünü oluşturun. (Sürüm, uygulama tanımlayıcı dosyasında, `version` öğesinde belirtilmiştir.)

Daha sonra, AIR uygulamasının güncelleme sürümünü web sunucunuza ekleyin:

1. AIR dosyasının güncelleme sürümünü web sunucunuza yerleştirin.

2. updateDescriptor.2.5.xml adlı bir metin dosyası oluşturun ve bu dosyaya aşağıdakileri ekleyin:

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
       <versionNumber>1.1</versionNumber>
       <url>http://example.com/updates/sample_1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

updateDescriptor.xml dosyasının `versionNumber`, `URL` ve `description` öğelerini güncelleme AIR dosyanızla eşleşecek biçimde düzenleyin. Bu güncelleme tanımlayıcısı biçimi, AIR 2.5 SDK'ye (veya üstüne) dahil olan güncelleme çerçevesini kullanan uygulamalar tarafından kullanılır.

3 updateDescriptor.1.0.xml adlı bir metin dosyası oluşturun ve bu dosyaya aşağıdakileri ekleyin:

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1</version>
       <url>http://example.com/updates/sample_1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

updateDescriptor.xml dosyasının `version`, `URL` ve `description` öğelerini güncelleme AIR dosyanızla eşleşecek biçimde düzenleyin. Bu güncelleme tanımlayıcısı biçimi, AIR 2 SDK'ye (veya önceki sürümlere) dahil olan güncelleme çerçevesini kullanan uygulamalar tarafından kullanılır.

***Not:** Bu ikinci güncelleme tanımlayıcısı dosyasının oluşturulması, yalnızca AIR 2.5'ten önce oluşturulmuş uygulamalar için güncelleme destekliyorsanız gereklidir.*

4 updateDescriptor.2.5.xml ve updateDescriptor.1.0.xml dosyasını, güncelleme AIR dosyasını içeren aynı web sunucusu dizinine ekleyin.

Bu temel bir örnektir, ancak çoğu uygulama için yeterli olan güncelleme işlevlerini sağlar. Bu belgenin devamı, güncelleme çerçevesini ihtiyaçlarınıza en uygun şekilde nasıl kullanabileceğinizi açıklar.

Güncelleme çerçevesinin başka bir örneği için, Adobe AIR geliştirici merkezinde aşağıdaki örnek uygulamaya bakın:

- Flash Tabanlı bir Uygulamada Çerçeveyi Güncelleme (http://www.adobe.com/go/learn_air_qs_update_framework_flash_tr)

AIR 2.5'e güncelleme

Uygulamalara sürüm numarası atama kuralları AIR 2.5'te değiştiğinden, AIR 2 güncelleme çerçevesi bir AIR 2.5 uygulama tanımlayıcısındaki sürüm bilgilerini çözümleyemez. Bu uyumsuzluk, uygulamanızı AIR 2.5 SDK'yi kullanmak için güncellemeden ÖNCE yeni güncelleme çerçevesini kullanmak üzere güncellemeniz gerektiği anlamına gelir. Bu nedenle, 2.5'ten önceki herhangi bir AIR sürümünü, AIR 2.5 veya daha sonraki bir sürüme güncellemek İKİ güncelleme gerektirir. İlk güncelleme AIR 2 ad alanını kullanmalı ve AIR 2.5 güncelleme çerçevesi kütüphanesini içermelidir (uygulama paketini AIR 2.5 SDK'yi kullanarak oluşturmaya devam edebilirsiniz). İkinci güncelleme AIR 2.5 ad alanını kullanabilir ve uygulamanızın yeni özelliklerini içerebilir.

Ayrıca ara güncellemenin, doğrudan AIR Updater sınıfını kullanarak AIR 2.5 uygulamasına güncelleme dışında hiçbir şey yapmamasını sağlayabilirsiniz.

Aşağıdaki örnek bir uygulamanın nasıl sürüm 1.0'dan 2.0'a güncelleneceğini gösterir. Sürüm 1.0 eski 2.0 ad alanını kullanır. Sürüm 2.0, 2.5 ad alanını kullanır ve AIR 2.5 API'leri kullanılarak uygulanan yeni özelliklere sahiptir.

1 Uygulamanın 1.0 sürümüne bağlı olarak 1.0.1 ara sürümünü oluşturun.

a Uygulamayı oluştururken AIR 2.5 Uygulama Güncelleyici çerçevesini kullanın.

***Not:** Flash teknolojisine bağlı AIR uygulamaları için `applicationupdater.swc` veya `applicationupdater_ui.swc` öğesini ve HTML tabanlı AIR uygulamaları için `applicationupdater.swf` veya `applicationupdater_ui.swf` öğesini kullanın.*

b Aşağıda gösterildiği gibi eski ad alanını ve sürümü kullanarak sürüm 1.0.1 için güncelleme tanımlayıcısı dosyası oluşturun:

```
<?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 Uygulamanın AIR 2.5 API'lerini ve 2.5 ad alanını kullanan 2.0 sürümünü oluşturun.

3 Uygulamayı 1.0.1 sürümünden 2.0 sürümüne güncellemek için güncelleme tanımlayıcısı oluşturun.

```
<?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>
```

Güncelleme tanımlayıcı dosyalarını tanımlama ve AIR dosyasını web sunucunuza ekleme

AIR güncelleme çerçevesini kullandığınızda, web sunucunuzda saklanan güncelleme tanımlayıcı dosyalarında bulunan güncelleme hakkında temel bilgileri tanımlarsınız. Güncelleme tanımlayıcı dosyası, basit bir XML dosyasıdır. Uygulamada yer alan güncelleme çerçevesi, yeni bir sürümün yüklenip yüklenmediğini görmek için bu dosyayı kontrol eder.

Güncelleme tanımlayıcı dosyasının biçimi AIR 2.5 için değişmiştir. Yeni biçim farklı bir ad alanı kullanır. Orijinal ad alanı“http://ns.adobe.com/air/framework/update/description/1.0” şeklindeydi. AIR 2.5 ad alanı“http://ns.adobe.com/air/framework/update/description/2.5” şeklindedir.

AIR 2.5'ten önce oluşturulmuş AIR uygulamaları yalnızca sürüm 1.0 güncelleme tanımlayıcısını okuyabilir. AIR 2.5 veya sonraki sürümlere dahil güncelleyici çerçevesi kullanılarak oluşturulmuş AIR uygulamaları yalnızca sürüm 2.5 güncelleme tanımlayıcısını okuyabilir. Bu sürüm uyumsuzluğu nedeniyle genellikle iki güncelleme tanımlayıcı dosyası oluşturmanız gerekir. Uygulamanızın AIR 2.5 sürümlerindeki güncelleme mantığı yeni biçimi kullanan bir güncelleme tanımlayıcısını indirmelidir. AIR uygulamanızın önceki sürümleri aynı biçimi kullanmaya devam etmelidir. İki dosya da yayınladığınız her güncelleme için değiştirilmelidir (siz AIR 2.5'ten önce oluşturulan sürümleri desteklemeyi durdurana kadar)

Güncelleme tanımlayıcı dosyası aşağıdaki verileri içerir:

- `versionNumber`—AIR uygulamasının yeni sürümü. `versionNumber` öğesini AIR 2.5 uygulamalarını güncellemek için kullanılan güncelleme tanımlayıcılarında kullanın. Değer yeni AIR uygulama tanımlayıcı dosyasının `versionNumber` öğesinde kullanılanla aynı dize olmalıdır. Güncelleme tanımlayıcı dosyasının sürüm numarası, güncelleme AIR dosyasının sürümü numarasıyla eşleşmiyorsa, güncelleme çerçevesi istisna verir.
- `version`—AIR uygulamasının yeni sürümü. AIR 2.5'ten önce oluşturulmuş uygulamaları güncellemek için kullanılan uygulama tanımlayıcılarda `version` öğesini kullanın. Değer yeni AIR uygulama tanımlayıcı dosyasının `version` öğesinde kullanılanla aynı dize olmalıdır. Güncelleme tanımlayıcı dosyasının sürümü, güncelleme AIR dosyasının sürümüyle eşleşmiyorsa, güncelleme çerçevesi istisna verir.

- `versionLabel`—Kullanıcılara gösterilmesi planlanan kişiler tarafından okunabilen sürüm dizesi. `versionLabel` isteğe bağlıdır, ancak yalnızca 2.5 sürümü güncelleme tanımlayıcı dosyalarında belirtilebilir. Uygulama tanımlayıcısında bir `versionLabel` öğesi kullanıyorsanız kullanın ve aynı değere ayarlayın.
- `url`—Güncelleme AIR dosyasının konumu. Bu, AIR uygulamasının güncelleme sürümünü içeren dosyadır.
- `description`—Yeni sürüme ilişkin ayrıntılar. Bu bilgi, güncelleme işlemi sırasında kullanıcıya gösterilebilir.

`version` ve `url` öğeleri zorunludur. `description` öğesi isteğe bağlıdır.

Örnek bir 2.5 sürümü güncelleme tanımlayıcı dosyası:

```
<?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>
```

Ve örnek bir 1.0 sürümü güncelleme tanımlayıcı dosyası:

```
<?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>
```

Birden çok dil kullanarak `description` etiketini tanımlamak istiyorsanız, bir `lang` niteliği tanımlayan birden çok `text` öğesi kullanın:

```
<?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>
```

Güncelleme tanımlayıcı dosyasını web sunucunuzda güncelleme AIR dosyasıyla birlikte konumlandırın.

Güncelleme tanımlayıcısına dahil olan templates dizini örnek güncelleme tanımlayıcı dosyaları içerir. Bunlar tek dilli ve çok dilli sürümler içerir.

Updater nesnesini başlatma

AIR güncelleme çerçevesini kodunuza yükledikten sonra (bkz. “Flex geliştirme ortamınızı ayarlama” sayfa 262 ve “Çerçeve dosyalarını HTML tabanlı bir AIR uygulamasına dahil etme” sayfa 262), aşağıdaki örnekte olduğu gibi bir updater nesnesini örneklemeniz gerekir.

ActionScript örneği:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

JavaScript örneği:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

Önceki kod ApplicationUpdater sınıfını kullanır (kullanıcı arabirimi sağlamayan). ApplicationUpdaterUI sınıfını kullanmak istiyorsanız (kullanıcı arabirimi sağlayan), aşağıdakini kullanın.

ActionScript örneği:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript örneği:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Bu belgenin devamında yer alan kod, `appUpdater` adlı bir updater nesnesini başlattığınızı varsayar.

Güncelleme ayarlarını yapılandırma

Hem ApplicationUpdater hem de ApplicationUpdaterUI uygulamayla birlikte verilen bir yapılandırma dosyası yoluyla veya uygulamadaki ActionScript ya da JavaScript aracılığıyla yapılandırılabilir.

Güncelleme ayarlarını XML yapılandırma dosyasında tanımlama

Güncelleme yapılandırma dosyası, bir XML dosyasıdır. Aşağıdaki öğeleri içerebilir:

- `updateURL`— Bir Dize. Güncelleme tanımlayıcısının uzak sunucudaki konumunu temsil eder. Herhangi bir geçerli URLRequest konumu olanaklıdır. Konfigürasyon dosyası veya komut dosyası aracılığıyla `updateURL` özelliğini tanımlamalısınız (bkz. "Güncelleme tanımlayıcı dosyalarını tanımlama ve AIR dosyasını web sunucunuza ekleme" sayfa 265). Güncelleyiciyi kullanmadan önce bu özelliği tanımlamanız gerekir ("Güncelleme çerçevesini başlatma" sayfa 269) bölümünde tanımlandığı biçimde updater nesnesinin `initialize()` yöntemini çağırmadan önce).
- `delay`—Bir Sayı. Güncellemeleri kontrol etmek için gün cinsinden (`0.25` gibi değerler geçerlidir) verilen zaman aralığını temsil eder. 0 değeri (varsayılan değer), güncelleyicinin otomatik bir dönemsel kontrol gerçekleştirmediğini belirtir.

ApplicationUpdaterUI için yapılandırma dosyası, `updateURL` ve `delay` öğelerinin yanı sıra aşağıdaki öğeyi içerebilir:

- `defaultUI`: Bir `dialog` öğeleri listesi. Her `dialog` öğesi, kullanıcı arabiriminde iletişim kutusuna karşılık gelen bir `name` niteliği içerir. Her`dialog` öğesi, iletişim kutusunun görünür olup olmadığını tanımlayan bir `visible` niteliğine sahiptir. Varsayılan olarak `true` değerindedir. `name` niteliği için olası değerler şöyledir:
  - `"checkForUpdate"`—Güncellemeyi Kontrol Et, Güncelleme Yok ve Hatayı Güncelle iletişim kutularına karşılık gelir
  - `"downloadUpdate"`—Güncellemeyi İndir iletişim kutusuna karşılık gelir
  - `"downloadProgress"`—İndirme İlerlemesi ve İndirme Hatası iletişim kutularına karşılık gelir
  - `"installUpdate"`—Güncellemeyi Yükle iletişim kutusuna karşılık gelir
  - `"fileUpdate"`—Dosya Güncelleme, Dosya Güncellemesi Yok ve Dosya Hatası iletişim kutularına karşılık gelir
- `"unexpectedError"`—Beklenmeyen Hata iletişim kutusuna karşılık gelir

  `false` olarak ayarlandığında, karşılık gelen iletişim kutusu güncelleme yordamının bölümü olarak görünmez.

Burada, ApplicationUpdater çerçevesine ilişkin yapılandırma dosyasının bir örneği bulunur:

```
<?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>
```

Burada, `defaultUI` öğesi için tanım içeren ApplicationUpdaterUI çerçevesine ilişkin yapılandırma dosyasının bir örneği bulunur:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
      <defaultUI>
         <dialog name="checkForUpdate" visible="false" />
         <dialog name="downloadUpdate" visible="false" />
         <dialog name="downloadProgress" visible="false" />
      </defaultUI>
</configuration>
```

`configurationFile` özelliğini dosyanın konumuna yönlendirin:

ActionScript örneği:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

JavaScript örneği:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

Güncelleme çerçevesinin templates dizini, örnek bir yapılandırma dosyası olan config-template.xml dosyasını içerir.

Güncelleme ayarları ActionScript veya JavaScript kodunu tanımlama

Bu yapılandırma parametreleri, uygulamadaki kodu aşağıdaki şekilde kullanarak da ayarlanabilir:

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

Updater nesnesi `updateURL` ve `delay` özelliklerine sahiptir. Bu özellikler, yapılandırma dosyasındaki `updateURL` ve `delay` öğeleriyle aynı özellikleri tanımlar: güncelleme tanımlayıcı dosyasının URL'si ve güncellemeleri kontrol etmek için aralık. Kodda bir yapılandırma dosyası *ve* ayarlarını tanımladığınızda, kod kullanılarak ayarlanan tüm özellikler yapılandırma dosyasında karşılık gelen ayarların yanında önceliğe sahip olur.

Güncelleyiciyi kullanmadan önce (updater nesnesinin `initialize()` yöntemini "Güncelleme çerçevesini başlatma" sayfa 269 bölümündeki şekilde çağırmadan önce) konfigürasyon dosyası veya komut dosyası yoluyla `updateURL` özelliğini tanımlamalısınız (bkz. "Güncelleme tanımlayıcı dosyalarını tanımlama ve AIR dosyasını web sunucunuza ekleme" sayfa 265).

ApplicationUpdaterUI çerçevesi, updater nesnesinin bu ek özelliklerini tanımlar:

- `isCheckForUpdateVisible`—Güncellemeyi Kontrol Et, Güncelleme Yok ve Hatayı Güncelle iletişim kutularına karşılık gelir
- `isDownloadUpdateVisible`—Güncellemeyi İndir iletişim kutusuna karşılık gelir
- `isDownloadProgressVisible`—İndirme İlerlemesi ve İndirme Hatası iletişim kutularına karşılık gelir
- `isInstallUpdateVisible`—Güncellemeyi Yükle iletişim kutusuna karşılık gelir
- `isFileUpdateVisible`—Dosya Güncelleme, Dosya Güncellemesi Yok ve Dosya Hatası iletişim kutularına karşılık gelir
- `isUnexpectedErrorVisible`—Beklenmeyen Hata iletişim kutusuna karşılık gelir

Her özellik, ApplicationUpdaterUI kullanıcı arabiriminde bir veya daha çok iletişim kutusuna karşılık gelir. Her özellik varsayılan olarak `true` değerine sahip olan bir Boolean değeridir. `false` olarak ayarlandığında, karşılık gelen iletişim kutuları güncelleme yordamının parçası olarak görüntülenmez.

Bu iletişim kutusu özellikleri, güncelleme yapılandırma dosyasındaki ayarları geçersiz kılar.

Güncelleme işlemi

AIR güncelleme çerçevesi, güncelleme işlemlerini aşağıdaki adımlarla tamamlar:

1. Güncelleyicinin başlatılması, tanımlanan gecikme aralığı içinde güncelleme kontrolünün gerçekleştirilip gerçekleştirilmediğini kontrol eder (bkz. "Güncelleme ayarlarını yapılandırma" sayfa 267). Güncelleme kontrolünün zamanı geldiğinde, güncelleme işlemi devam eder.
2. Güncelleyici, güncelleme tanımlayıcı dosyasını indirir ve yorumlar.
3. Güncelleyici, güncelleme AIR dosyasını indirir.
4. Güncelleyici, uygulamanın güncel sürümünü yükler.

Updater nesnesi, bu adımların her biri tamamlandığında olaylar gönderir. ApplicationUpdater sürümünde, işlemdeki bir adımın başarıyla tamamlandığını gösteren olayları iptal edebilirsiniz. Bu olaylardan birini iptal ettiğinizde, işlemin sonraki adımı iptal edilir. Güncelleyici, ApplicationUpdaterUI sürümünde kullanıcının işlemin her adımında iptal etmesine veya devam etmesine olanak sağlayan bir iletişim kutusu sunar.

Olayı iptal ederseniz, işleme devam etmek için updater nesnesinin yöntemlerini çağırabilirsiniz.

Güncelleyicinin ApplicationUpdater sürümü güncelleme işlemi boyunca ilerledikçe, güncel durumunu `currentState` özelliğinde kaydeder. Bu özellik, aşağıdaki olası değerlere sahip bir dizeye ayarlanır:

- `"UNINITIALIZED"`—Güncelleyici başlatılmadı.
- `"INITIALIZING"`—Güncelleyici başlatılıyor.
- `"READY"`—Güncelleyici başlatıldı
- `"BEFORE_CHECKING"`—Güncelleyici, güncelleme tanımlayıcı dosyası için henüz kontrol edilmedi.
- `"CHECKING"`—Güncelleyici, güncelleme tanımlayıcı dosyası için kontrol ediliyor.
- `"AVAILABLE"`—Güncelleme tanımlayıcı dosyası mevcut.
- `"DOWNLOADING"`—Güncelleyici, AIR dosyasını indiriyor.
- `"DOWNLOADED"`—Güncelleyici, AIR dosyasını indirdi.
- `"INSTALLING"`—Güncelleyici, AIR dosyasını yüklüyor.
- `"PENDING_INSTALLING"`—Güncelleyici başlatıldı ve bekleyen güncellemeler mevcut.

Updater nesnesinin bazı yöntemleri yalnızca güncelleyici belirli bir durumda olduğunda yürütülür.

Güncelleme çerçevesini başlatma

Konfigürasyon özelliklerini ayarladıktan sonra (bkz. "Temel örnek: ApplicationUpdaterUI sürümünü kullanma" sayfa 263), güncellemeyi başlatmak için `initialize()` yöntemini çağırın:

```
appUpdater.initialize();
```

Bu yöntem şunu yapar:

- Senkronize olarak bekleyen güncellemeleri sessizce yüklerken, güncelleme çerçevesini başlatır. Bu yöntem, çağrıldığında uygulamayı yeniden başlatabileceğinden, uygulamanın başlatılması sırasında çağrılmalıdır.
- Ertelenen bir güncelleme olup olmadığını kontrol eder ve bunu yükler.
- Güncelleme işlemi sırasında hata oluştuğunda, uygulama depo alanından güncelleme dosyasını ve sürüm bilgisini temizler.
- Gecikmenin süresi dolduğunda güncelleme işlemini başlatır. Aksi halde, zamanlayıcıyı yeniden başlatır.

Bu yöntemin çağrılması, updater nesnesinin aşağıdaki olayları göndermesiyle sonuçlanabilir:

- `UpdateEvent.INITIALIZED`—Başlatma tamamlandığında gönderilir.
- `ErrorEvent.ERROR`—Başlatma sırasında hata olduğunda gönderilir.

`UpdateEvent.INITIALIZED` olayı gönderildiğinde, güncelleme işlemi tamamlanmıştır.

`initialize()` yöntemini çağırdığınızda, güncelleyici güncelleme işlemini başlatır ve zamanlayıcı gecikme ayarını temel alarak tüm adımları tamamlar. Ancak, her zaman updater nesnesinin `checkNow()` yöntemini çağırarak güncelleme işlemini başlatabilirsiniz.

```
appUpdater.checkNow();
```

Güncelleme işlemi zaten çalışıyorsa, bu yöntem etkisizdir. Aksi halde, güncelleme işlemini başlatır.

Updater nesnesi, `checkNow()` yönteminin çağrılmasının sonucu olarak aşağıdaki olayı gönderebilir:

- Güncelleme tanımlayıcı dosyasını indirmeye çalışmadan hemen önce `UpdateEvent.CHECK_FOR_UPDATE` olayı.

`checkForUpdate` olayını iptal ederseniz, updater nesnesinin `checkForUpdate()` yöntemini çağırabilirsiniz. (Bkz. sonraki bölüm.) Olayı iptal etmezseniz, güncelleme işlemi güncelleme tanımlayıcı dosyasını kontrol etmeye devam eder.

ApplicationUpdaterUI sürümünde güncelleme işlemini yönetme

Kullanıcı, ApplicationUpdaterUI sürümünde, kullanıcı arabiriminin iletişim kutularındaki İptal düğmeleri yoluyla işlemi iptal edebilir. Ayrıca, ApplicationUpdaterUI nesnesinin `cancelUpdate()` yöntemini çağırarak da güncelleme işlemini programlama yoluyla iptal edebilirsiniz.

Güncelleyicinin hangi iletişim kutusu teyitlerini görüntülediğini belirlemek için ApplicationUpdaterUI nesnesinin özelliklerini ayarlayabilir veya güncelleme yapılandırma dosyasındaki öğeleri tanımlayabilirsiniz. Ayrıntılar için bkz. “Güncelleme ayarlarını yapılandırma” sayfa 267.

ApplicationUpdater sürümünde güncelleme işlemini yönetme

Güncelleme işleminin adımlarını iptal etmek için ApplicationUpdater nesnesi tarafından gönderilen olay nesnelerinin `preventDefault()` yöntemini çağırabilirsiniz (bkz. “Güncelleme işlemi” sayfa 269). Varsayılan davranışın iptal edilmesi, uygulamanıza kullanıcının devam etmek isteyip istemediğini soran bir mesaj görüntüleme şansı verir.

Aşağıdaki adımlar, güncelleme işleminin bir adımı iptal edildiğinde işleme nasıl devam edileceğini açıklar.

Güncelleme tanımlayıcı dosyasını indirme ve yorumlama

ApplicationUpdater nesnesi, `checkForUpdate` olayını güncelleme işlemi başlamadan, güncelleyici güncelleme tanımlayıcı dosyasını indirmeye çalışmadan hemen önce gönderir. `checkForUpdate` olayının varsayılan davranışını iptal ederseniz, güncelleyici güncelleme tanımlayıcı dosyasını indirmez. Güncelleme işlemine devam etmek için `checkForUpdate()` yöntemini çağırabilirsiniz:

```
appUpdater.checkForUpdate();
```

`checkForUpdate()` yönteminin çağrılması güncelleyicinin güncelleme tanımlayıcı dosyasını senkronize olmayan şekilde indirmesine ve yorumlamasına neden olur. Güncelleyici, `checkForUpdate()` yönteminin çağrılmasının sonucu olarak aşağıdaki olayları gönderebilir:

- `StatusUpdateEvent.UPDATE_STATUS`—Güncelleyici güncelleme tanımlayıcı dosyasını başarıyla indirdi ve yorumladı. Bu olay şu özelliklere sahiptir:
  - `available`—Bir Boolean değeri. Güncel uygulamanın sürümünden farklı bir sürüm mevcut olduğunda `true` olarak; aksi halde (sürüm aynıysa) `false` olarak ayarlanır.

- `version`—Bir Dize. Güncelleme dosyasının uygulama tanımlayıcı dosyasından sürüm
- `details`—Bir Dizi. Tanımlamanın yerelleştirilmiş sürümleri mevcutsa, bu dizi ilk öğe olarak boş bir dize (`""`) ve ikinci öğe olarak tanımı içerir.

  Tanımın birden çok sürümü mevcutsa (güncelleme tanımlayıcı dosyasında), bu dizi birden çok alt dizi içerir. Her dizi iki öğeye sahiptir: ilk öğe dil kodu (örn. `"en"`) ve ikinci öğe bu dile karşılık gelen tanımdır (bir Dize). Bkz. “Güncelleme tanımlayıcı dosyalarını tanımlama ve AIR dosyasını web sunucunuza ekleme” sayfa 265.

- `StatusUpdateErrorEvent.UPDATE_ERROR`—Hata oluştu ve güncelleyici güncelleme tanımlayıcı dosyasını indiremedi veya yorumlayamadı.

Güncelleme AIR dosyasını indirme

Güncelleyici güncelleme tanımlayıcı dosyasını başarıyla indirip yorumladıktan sonra, ApplicationUpdater nesnesi `updateStatus` olayını gönderir. Varsayılan davranış, mevcutsa güncelleme dosyasını indirmeye başlamaktır. Varsayılan davranışı iptal ederseniz, güncelleme işlemine devam etmek için `downloadUpdate()` yöntemini çağırabilirsiniz:

```
appUpdater.downloadUpdate();
```

Bu yöntemin çağrılması, güncelleyicinin AIR dosyasının güncel sürümünü senkronize olmayan biçimde indirmesine neden olur.

`downloadUpdate()` yöntemi aşağıdaki olayları gönderebilir:

- `UpdateEvent.DOWNLOAD_START`—Sunucuya bağlantı kuruldu. ApplicationUpdaterUI kitaplığı kullanılırken, bu olay indirme ilerlemesini izleyen bir ilerleme çubuğu içeren bir iletişim penceresi görüntüler.
- `ProgressEvent.PROGRESS`—Dosya indirme ilerlerken düzenli aralıklarla gönderilir.
- `DownloadErrorEvent.DOWNLOAD_ERROR`—Güncelleme dosyasına bağlanma veya bu dosyayı indirme sırasında hata oluştuğunda gönderilir. Ayrıca geçersiz HTTP durumlarında gönderilir. (Örn. “404 - Dosya bulunamadı”.) Bu olay, ek hata bilgilerini tanımlayan bir tam sayı olan `errorID` özelliğini içerir. Ek `subErrorID` özelliği, daha fazla bilgi içerir.
- `UpdateEvent.DOWNLOAD_COMPLETE`—Güncelleyici güncelleme tanımlayıcı dosyasını başarıyla indirdi ve yorumladı. Bu olayı iptal etmezseniz, ApplicationUpdater sürümü, güncel sürümü yüklemeye devam eder. ApplicationUpdaterUI sürümünde, kullanıcıya devam etme ve etmeme seçeneklerini veren bir iletişim kutusu sunulur.

Uygulamayı güncelleme

ApplicationUpdater nesnesi, güncelleme dosyasının indirilmesi tamamlandığında `downloadComplete` olayını gönderir. Varsayılan davranışı iptal ederseniz, güncelleme işlemine devam etmek için `installUpdate()` yöntemini çağırabilirsiniz:

```
appUpdater.installUpdate(file);
```

Bu yöntemin çağrılması, güncelleyicinin AIR dosyasının güncel sürümünü yüklemesine neden olur. Bu yöntem, güncelleme olarak AIR dosyasına referans veren bir File nesnesi olan tek bir `file` parametresi içerir.

ApplicationUpdater nesnesi, `installUpdate()` yöntemini çağırmanın bir sonucu olarak `beforeInstall` olayını gönderebilir:

- `UpdateEvent.BEFORE_INSTALL`—Güncellemenin yüklenmesinden hemen önce gönderilir. Bazen, güncelleme devam etmeden kullanıcının geçerli çalışmayı tamamlayabilmesi için, bu sırada güncellemenin yüklenmesini önlemek yararlıdır. Event nesnesinin `preventDefault()` yönteminin çağrılması, yüklemeyi bir sonraki yeniden başlatmaya kadar erteler ve ek güncelleme işlemi başlatılamaz. (Bunlar, `checkNow()` yönteminin çağrılması veya dönemsel kontrol sonucu oluşan güncellemeleri kapsar.)

Rastgele bir AIR dosyasından yükleme

Kullanıcının bilgisayarındaki bir AIR dosyasından yüklemek üzere güncel sürümü yüklemek için `installFromAIRFile()` yöntemini çağırabilirsiniz:

```
appUpdater.installFromAIRFile();
```

Bu yöntem, güncelleyicinin AIR dosyasından uygulamanın güncel sürümünü yüklemesine neden olur.

`installFromAIRFile()` yöntemi aşağıdaki olayları gönderebilir:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS`—ApplicationUpdater, dosyayı `installFromAIRFile()` yöntemini kullanarak başarıyla doğruladıktan sonra gönderilir. Bu olay, aşağıdaki özelliklere sahiptir.
  - `available`—Güncel uygulamanın sürümünden farklı bir sürüm mevcut olduğunda `true` olarak; aksi halde (sürüm aynıysa) `false` olarak ayarlıdır.
  - `version` —Yeni kullanılabilir sürümü temsil eden dize.
  - `path`—Güncelleme dosyasının yerel yolunu temsil eder.

  StatusFileUpdateEvent nesnesinin mevcut özelliği `true` olarak ayarlandığında bu olayı iptal edebilirsiniz. Bu olayın iptal edilmesi, güncellemenin ilerlemesini iptal eder. İptal edilen güncellemeye devam etmek için `installUpdate()` yöntemini çağırın.
- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR`—Hata oluştu ve güncelleyici AIR uygulamasını yükleyemedi.

Güncelleme işlemini iptal etme

Güncelleme işlemini iptal etmek için `cancelUpdate()` yöntemini çağırabilirsiniz:

```
appUpdater.cancelUpdate();
```

Bu yöntem, tüm eksik indirilen dosyaları silerek, bekleyen tüm indirmeleri iptal eder ve dönemsel kontrol zamanlayıcısını yeniden başlatır.

Updater nesnesi başlatılıyorsa, bu yöntem etkisizdir.

ApplicationUpdaterUI arabirimini yerelleştirme

ApplicationUpdaterUI sınıfı, güncelleme işlemi için varsayılan bir kullanıcı arabirimi sağlar. Bu, kullanıcının işlemi başlatmasına, işlemi iptal etmesine ve diğer ilgili işlemleri gerçekleştirmesine olanak sağlayan iletişim kutularını içerir.

Güncelleme tanımlayıcı dosyasının `description` öğesi, uygulama açıklamasını farklı dillerde tanımlamanızı sağlar. Aşağıdaki gibi `lang` niteliklerini tanımlayan birden çok `text` öğesi kullanın:

```
<?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>
```

Güncelleme çerçevesi, son kullanıcının yerelleştirme zincirine en çok uyan tanımı kullanır. Daha fazla bilgi için bkz. Güncelleme tanımlayıcı dosyasını tanımlama ve AIR dosyasını web sunucunuza ekleme.

Flex geliştiricileri, doğrudan `"ApplicationUpdaterDialogs"` paketine yeni bir dil ekleyebilir.

JavaScript geliştiricileri, updater nesnesinin `addResources()` yöntemini çağırabilir. Bu yöntem, bir dil için dinamik olarak yeni bir kaynak paketi ekler. Kaynak paketi, bir dile ilişkin yerelleştirilmiş dizeleri tanımlar. Bu dizeler, çeşitli iletişim kutusu metin alanlarında kullanılır.

JavaScript geliştiricileri, kullanıcı arabirimi tarafından kullanılan yerel ayar zincirini tanımlamak için ApplicationUpdaterUI sınıfının `localeChain` özelliğini kullanabilir. Genellikle yalnızca JavaScript (HTML) geliştiricileri bu özelliği kullanır. Flex geliştiricileri, yerel ayar zincirini yönetmek için ResourceManager kullanabilir.

Örneğin, aşağıdaki JavaScript kodu Rumence ve Macarca için kaynak paketlerini tanımlar:

```
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;
```

Ayrıntılar için bkz. dil başvurusundaki ApplicationUpdaterUI sınıfının `addResources()` yöntemi.

Bölüm 18: Kaynak Kodunu Görüntüleme

Bir web tarayıcısındaki bir HTML sayfasının kaynak kodunu yalnızca tek bir kullanıcı görüntüleyebilir, kullanıcılar HTML tabanlı AIR uygulamasının kaynak kodunu görüntüleyebilirler. Adobe® AIR® SDK, kaynak kodunuzu son kullanıcılara göstermek için uygulamanızda kullanabileceğiniz bir AIRSourceViewer.js JavaScript dosyası içerir.

Kaynak Görüntüleyicisi'ni yükleme, yapılandırma ve açma

Kaynak Görüntüleyicisi kodu AIR SDK'nin çerçeveler dizininde bulunan AIRSourceViewer.js adlı bir JavaScript dosyasına dahil edilir. Kaynak Görüntüleyicisi'ni uygulamanızda kullanmak için, AIRSourceViewer.js dosyasını uygulama proje dizininize kopyalayın ve dosyayı uygulamanızdaki ana HTML dosyası aracılığıyla yükleyin:

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

AIRSourceViewer.js dosyası `air.SourceViewer` öğesini çağırarak JavaScript kodundan erişebileceğiniz SourceViewer adlı bir sınıfı tanımlar.

SourceViewer sınıfı üç yöntem tanımlar: `getDefault()`, `setup()` ve `viewSource()`.

Yöntem	Açıklama
`getDefault()`	Statik bir yöntem. Başka yöntemler çağırmak için kullanabileceğiniz bir SourceViewer örneği gönderir.
`setup()`	Yapılandırma ayarlarını Kaynak Görüntüleyicisi'ne uygular. Ayrıntılı bilgi için, bkz. "Kaynak Görüntüleyicisi'ni yapılandırma" sayfa 274
`viewSource()`	Kullanıcının ana bilgisayar uygulamasının kaynak dosyalarını tarayıp açabileceği yeni bir pencere açar.

Not: *Kaynak Görüntüleyicisi'ni kullanan kod uygulama güvenlik sanal alanında olmalıdır (uygulama dizininde bulunan bir dosyada.)*

Örneğin, aşağıdaki JavaScript kodu bir Kaynak Görüntüleyicisi kodu başlatır ve tüm kaynak dosyalarını listeleyen bir Kaynak Görüntüleyicisi penceresi açar:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Kaynak Görüntüleyicisi'ni yapılandırma

`config()` yöntemi verilen ayarları Kaynak Görüntüleyicisi'ne uygular. Bu yöntem tek bir parametre alır: `configObject`. `configObject` nesnesi Kaynak Görüntüleyicisi'ne ilişkin ayarları tanımlayan özelliklere sahiptir. Bu özellikler, `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` ve `typesToAdd` özellikleridir.

default

Kaynak Görüntüleyicisi'nde görüntülenecek ilk dosyaya giden göreceli yolu belirleyen dize.

Örneğin, aşağıdaki JavaScript kodu Kaynak Görüntüleyicisi penceresini gösterilen ilk dosya olarak index.html dosyasıyla açar:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

hariç tut

Kaynak Görüntüleyicisi listesinden hariç tutulacak bir dizeler dizisi. Yollar uygulama diziniyle ilişkilidir. Joker karakterler desteklenmez.

Örneğin, aşağıdaki JavaScript kodu Kaynak Görüntüleyicisi penceresini açar ve AIRSourceViewer.js dosyası ve Görüntüler ve Sesler alt dizinlerindeki dosyalar dışındaki tüm kaynak dosyaları listeler:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

İki sayı içeren ve Kaynak Görüntüleyicisi penceresinin x ve y koordinatlarını belirleyen bir dizi.

Örneğin, aşağıdaki JavaScript kodu Kaynak Görüntüleyicisi penceresini [40, 60] (X = 40, Y = 60) ekran koordinatlarında açar:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

Kaynak Görüntüleyicisi'nin bir modal (true) pencere mi yoksa modal olmayan (false) bir pencere mi olması gerektiğini belirleyen bir Boolean değeri. Varsayılan olarak, Kaynak Görüntüleyicisi penceresi modsaldır.

Örneğin, aşağıdaki JavaScript kodu, Kaynak Görüntüleyicisi penceresini kullanıcının hem Kaynak Görüntüleyicisi penceresiyle hem de herhangi bir uygulama penceresiyle etkileşebileceği şekilde açar:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

Kaynak Görüntüleyicisi'nde bulunan varsayılan türlere ek olarak dahil edilecek dosya türlerini belirleyen bir dizeler dizisi.

Varsayılan olarak, Kaynak Görüntüleyicisi aşağıdaki dosya türlerini listeler:

- Metin dosyaları: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Görüntü dosyaları: JPG, JPEG, PNG, GIF

  Hiç bir değer belirli değilse, varsayılan tüm türler dahil edilir (`typesToExclude` özelliğinde belirtilenler hariç).

  Örneğin, aşağıdaki Kaynak Görüntüleyicisi penceresini açan JavaScript kodu VCF ve VCARD dosyalarını içerir:

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToAdd = ["text.vcf", "text.vcard"];
  viewer.viewSource(configObj);
  ```

Listelediğiniz her dosya türü için, "metin" (metin dosya türleri için) veya "görüntü" (görüntü dosya türleri için) dosyası olduğunu belirtmeniz gerekir.

typesToExclude
Kaynak Görüntüleyicisi'nden hariç tutulacak dosya türlerini belirleyen dizeler dizisi.

Varsayılan olarak, Kaynak Görüntüleyicisi aşağıdaki dosya türlerini listeler:

- Metin dosyaları: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Görüntü dosyaları: JPG, JPEG, PNG, GIF

  Örneğin, aşağıdaki JavaScript kodu, GIF veya XML dosyalarını listelemeden Kaynak Görüntüleyicisi penceresini açar:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

  Listelediğiniz her dosya türü için, `"metin"` (metin dosya türleri için) veya `"görüntü"` (görüntü dosya türleri için) dosyası olduğunu belirtmeniz gerekir.

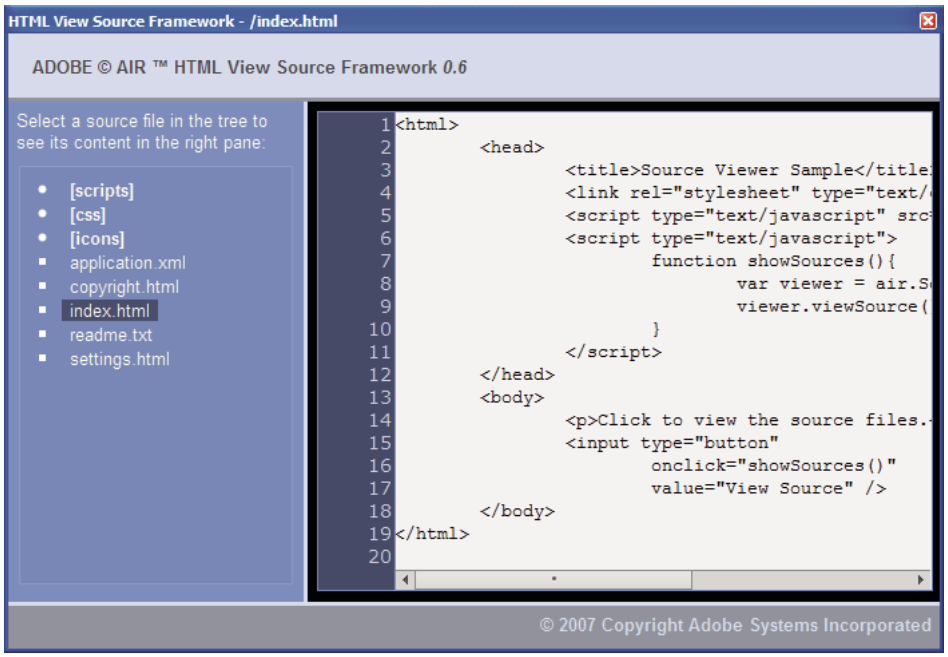
Kaynak Görüntüleyicisi'ni Açma

Kullanıcı onu seçtiğinde Kaynak Görüntüleyicisi kodunu çağıran bir bağlantı, düğme veya menü komutu gibi bir kullanıcı arabirim öğesini dahil etmelisiniz. Örneğin, aşağıdaki basit uygulama kullanıcı bir bağlantıyı tıklattığında Kaynak Görüntüleyicisi'ni açar:

```
<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>
```

Kaynak Görüntüleyicisi kullanıcı arabirimi

Uygulama SourceViewer nesnesinin `viewSource()` yöntemini çağırdığında, AIR uygulaması bir Kaynak Görüntüleyicisi penceresi açar. Pencere bir kaynak dosyalar ve dizinler listesi (solda) ve seçili dosyanın kaynak kodunu gösteren bir görüntüleme alanı (sağda) içerir:



Dizinler parantez içinde listelenir. Kullanıcı bir dizinin listesini genişletmek veya daraltmak için bir sol ayracı tıklatabilir.

Kaynak Görüntüleyicisi metin dosyalarına ilişkin kaynağı tanınan uzantılarla (HTML, JS, TXT, XML ve diğerleri gibi) veya tanınan görüntü uzantılarıyla (JPG, JPEG, PNG ve GIF) görüntüleyebilir. Kullanıcı tanınan bir dosya uzantısı olmayan bir dosya seçerse, bir hata mesajı görüntülenir (“Bu dosya türünden metin içeriği alınamıyor”).

`setup()` yöntemiyle dahil edilen herhangi bir kaynak dosyası listelenmez (bkz. “Kaynak Görüntüleyicisi'ni yükleme, yapılandırma ve açma” sayfa 274).

Bölüm 19: AIR HTML Introspector ile hata ayıklama

Adobe® AIR® SDK HTML tabanlı uygulamalarda hata ayıklamaya yardım etmek için uygulamanıza dahil edebileceğiniz bir AIRIntrospector.js JavaScript dosyası içerir.

AIR Introspector Hakkında

(AIR HTML Introspector adıyla kullanılan) Adobe AIR HTML/JavaScript Application Introspector HTML tabanlı uygulama geliştirme ve hata ayıklamaya yardımcı olan kullanışlı özellikler sağlar:

- Uygulamadaki bir kullanıcı arabirimini işaretlemenize ve onun işaretlemesi ve DOM özelliklerini görmenize olanak veren bir iç inceleyici araç içerir.
- İç inceleme için nesne başvuruları göndermeye yarayan bir konsol içerir ve özellik değerlerini ayarlayabilir ve JavaScript kodu çalıştırabilirsiniz. Ayrıca konsola nesneleri serileştirebilirsiniz, bu da veriyi düzenlemenizi sınırlar. Ayrıca konsoldan metin kopyalayabilir ve kaydedebilirsiniz.
- Ayrıca DOM özellik ve işlevleri için bir ağaç görünümü içerir.
- DOM öğelerine ilişkin nitelikleri ve metin düğümlerini düzenlemenize izin verir.
- Uygulamanızda yüklenen bağlantılar, CSS stilleri, görüntüler ve JavaScript dosyaları listeler.
- Kullanıcı arabirimine ilişkin ilk HTML kaynağı ve geçerli işaretleme kaynağını görüntülemenize olanak tanır.
- Uygulama dizinindeki dosyalara erişmenize olanak tanır. (Bu özellik yalnızca uygulama sanal alanı için açılan AIR HTML Introspector konsolu için kullanılabilir. Uygulama harici sanal alan içeriğine açık olan konsollar için kullanılabilir değildir.)
- XMLHttpRequest nesneleri ve özellikleri için bir görüntüleyici içerir, buna `responseText` ve `responseXML` (kullanılabilir olduğunda) dahildir.
- Eşleşen metni kaynak kod ve dosyalarda aratabilirsiniz.

AIR Introspector kodu yükleniyor.

AIR Introspector kodu AIR SDK'nin çerçeveler dizinine dahil edilen AIRSourceViewer.js adlı bir JavaScript dosyasına dahil edilir. AIR Introspector'ı uygulamanızda kullanmak için, AIRIntrospector.js dosyasını uygulama proje dizininize kopyalayın ve dosyayı uygulamanızdaki ana HTML dosyası aracılığıyla yükleyin:

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

Ayrıca dosyayı uygulamanızdaki farklı yerel pencerelere karşılık gelen her HTML dosyasında içerin.

Önemli: *AIRIntrospector.js dosyasını yalnızca uygulamayı geliştirirken ve onun hatalarını ayıklarken dahil edin. Onu dağıttığınız paketli AIR uygulamasıyla kaldırın.*

AIRIntrospector.js dosyası `air.SourceViewer` öğesini çağırarak JavaScript kodundan erişebileceğiniz Console adlı bir sınıfı tanımlar.

Not: *AIR Introspector'ı kullanan kod uygulama güvenlik sanal alanında olmalıdır (uygulama dizininde bulunan bir dosyada.)*

Bir nesneyi Konsol sekmesinde denetlemek.

Konsol sınıfı beş yöntem tanımlar: `log()`, `warn()`, `info()`, `error()` ve `dump()`.

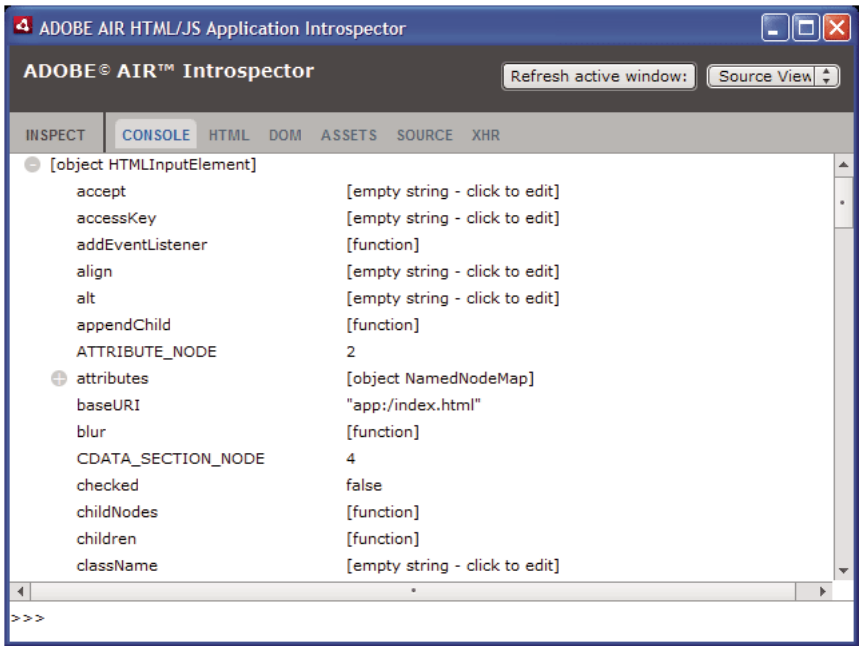
`log()`, `warn()`, `info()` ve `error()` yöntemlerinin hepsi Konsol sekmesine bir nesne göndermenize olanak tanır. Bu yöntemlerin en temel olanı `log()` yöntemidir. Aşağıdaki kod Konsol sekmesine `test` değişkeni tarafından temsil edilen basit bir nesne gönderir:

```
var test = "hello";
air.Introspector.Console.log(test);
```

Ancak, Konsol sekmesine bir karmaşık nesne göndermek daha kullanışlıdır. Örneğin, aşağıdaki HTML sayfası, button nesnesinin kendisini Konsol sekmesine gönderen (`btn1`) adlı bir düğme içerir:

```
<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>
```

Düğmeye bastığınızda, Konsol sekmesi btn1 nesnesini görüntüler ve nesnenin özelliklerini denetlemek için onun ağaç görünümünü genişletebilirsiniz:



Nesnenin bir özelliğini, özellik adının sağındaki listeyi tıklatarak ve metin listesini değiştirerek düzenleyebilirsiniz.

`info()`, `error()` ve `warn()` yöntemleri `log()` yöntemiyle aynıdır. Ancak, bu yöntemleri çağırdığınızda, Konsol satırın başında bir simge görüntüler:

Yöntem	Simge
`info()`	
`error()`	
`warn()`	

`log()`, `warn()`, `info()` ve `error()` yöntemleri yalnızca gerçek nesneye bir başvuru gönderir, bu nedenle de kullanılabilir olan özellikler görüntüleme sırasında olanlardır. Gerçek nesneyi serileştirmek istiyorsanız, `dump()` yöntemini kullanın. Bu yöntem iki parametreye sahiptir:

Parametre	Açıklama
`dumpObject`	Serileştirilecek nesne.
`düzeyler`	Nesne ağacında incelenecek maksimum düzey sayısı (kök düzeyine ek olarak). Varsayılan değer 1'dir (yani ağacın kök düzeyinin ötesindeki bir düzey gösterilir). Bu parametre isteğe bağlıdır.

`dump()` yöntemini çağırmak bir nesneyi Console sekmesine göndermeden önce onu serileştirir, böylece nesnenin özelliklerini düzenleyemezsiniz. Örneğin, şu kodu göz önünde bulundurun:

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

Bu kodu çalıştırdığınızda, Konsol `testObject` nesnesini ve onun özelliklerini görüntüler ancak Konsol'da özellik değerlerini düzenleyemezsiniz.

AIR Introspector'ı Yapılandırma

Konsolu küresel `AIRIntrospectorConfig` değişkeninin özelliklerini ayarlayarak yapılandırabilirsiniz. Örneğin, aşağıdaki JavaScript kodu AIR Introspector uygulaması sütunlarını 100 karakterde bir saracak şekilde yapılandırır:

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

AIRIntrospector.js dosyasını (bir `script` etiketiyle) yüklemeden önce `AIRIntrospectorConfig` değişkeninin özelliklerini ayarladığınızdan emin olun.

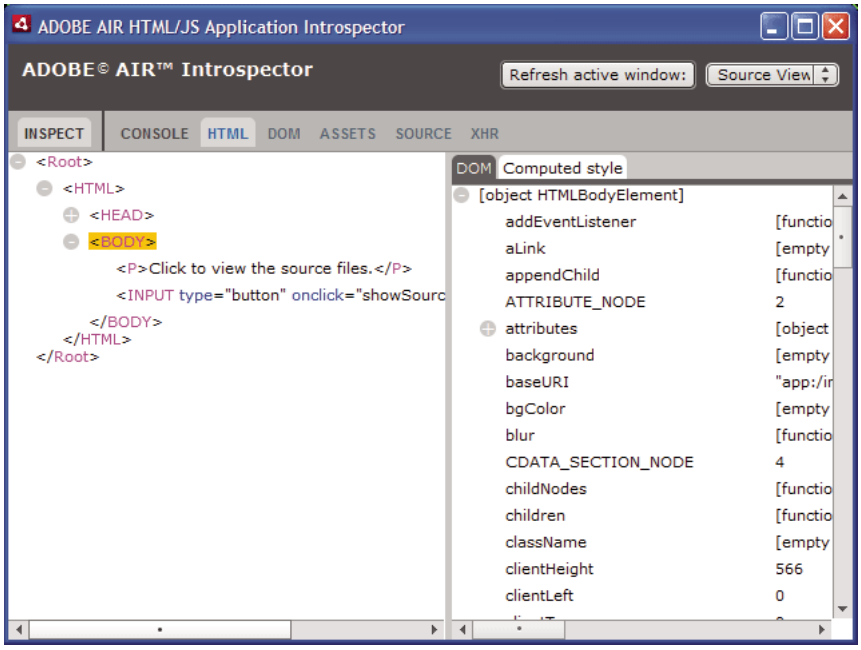
`AIRIntrospectorConfig` değişkeninin sekiz özelliği vardır:

Özellik	Varsayılan değer	Açıklama
`closeIntrospectorOnExit`	`true`	Denetleyici penceresini uygulamanın diğer tüm pencereleri kapandığında kapanacak şekilde ayarlar.
`debuggerKey`	123 (F12 tuşu)	AIR Introspector penceresini göstermek ve saklamak için kullanılan klavye kısayolunun tuş kodu.
`debugRuntimeObjects`	`true`	Introspector'ı JavaScript'te tanımlanan nesnelere ek olarak çalışma zamanı nesnelerini de genişletmeye ayarlar.
`flashTabLabels`	`true`	Konsol ve XMLHttpRequest sekmelerini onlarda bir değişiklik olduğunda haber vererek yanıp sönmeye ayarlar (örneğin, bu sekmelerde bir metin kaydolduğunda).
`introspectorKey`	122 (F11 tuşu)	Denetleme panelini açacak klavye kısayoluna ilişkin tuş kodu.
`showTimestamp`	`true`	Konsol sekmesini her satırın başında zaman damgaları görüntüleyecek şekilde ayarlar.
`showSender`	`true`	Mesajı her satırın başında göndererek nesne üzerinde bilgi görüntülemek için Konsol sekmesini ayarlar.
`wrapColumns`	2000	Kaynak dosyaların sarıldığı sütun sayısı.

AIR Introspector arabirimi

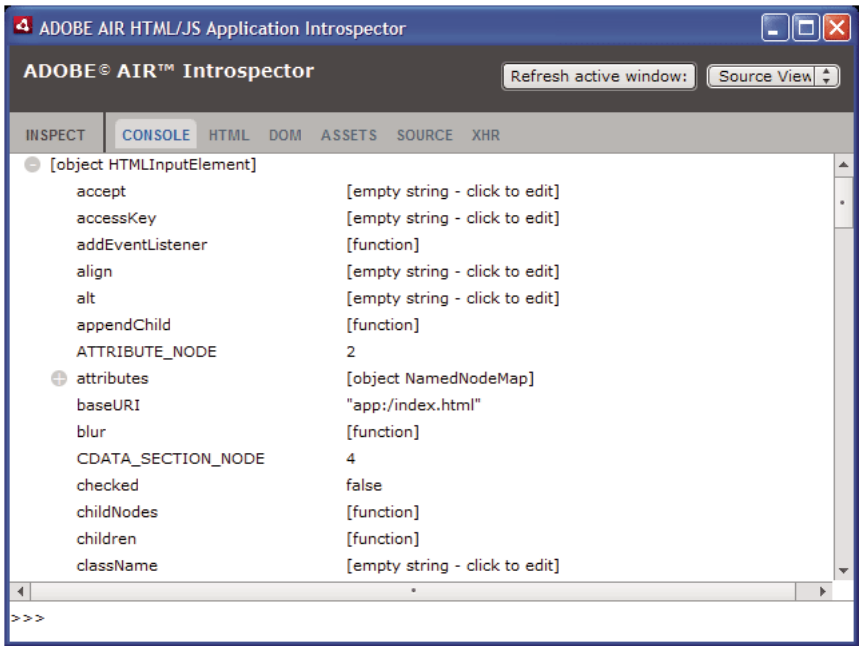
Uygulamanın hata ayıklamasını yaparken AIR iç inceleyici penceresini açmak için, F12 tuşuna basın veya Konsol sınıfının yöntemlerinden birini çağırın (bkz. “Bir nesneyi Konsol sekmesinde denetlemek.” sayfa 279). Kısayol tuşunu F12 tuşundan başka bir tuş olarak yapılandırabilirsiniz; bkz. “AIR Introspector'ı Yapılandırma” sayfa 281.

AIR Introspector penceresinde aşağıdaki örnekte gösterildiği gibi altı sekme vardır: Konsol, HTML, DOM, Varlıklar, Kaynak ve XHR:



Konsol sekmesi

Konsol sekmesi air.Introspector.Console sınıfının yöntemlerinden birine parametreler olarak geçen özellik değerlerini görüntüler. Ayrıntılı bilgi için, bkz. "Bir nesneyi Konsol sekmesinde denetlemek." sayfa 279.

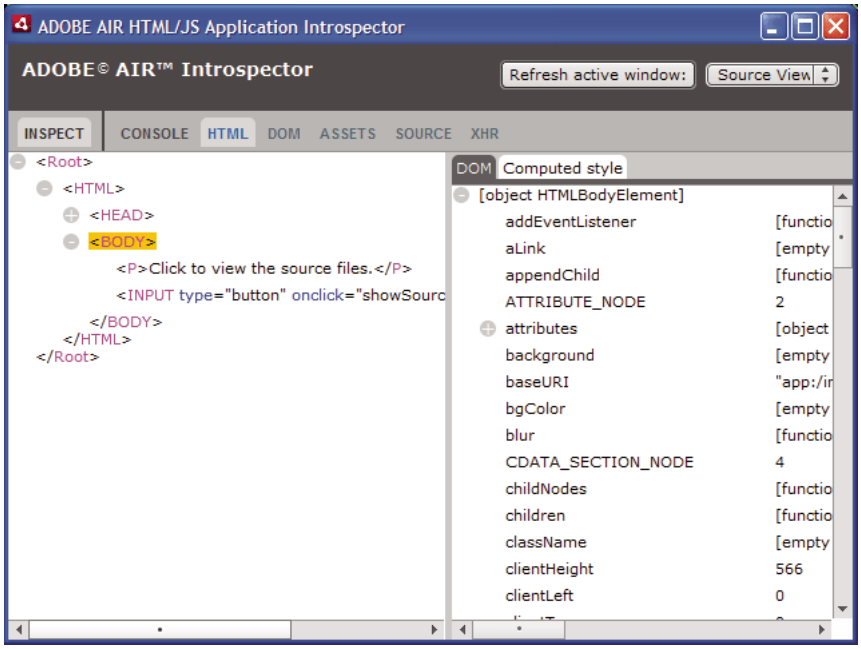


- Konsolu temizlemek için, metni sağ tıklayın ve Konsolu Temizle'yi tıklayın.

- Konsol sekmesindeki metni bir dosyaya kaydetmek için, Konsol sekmesini sağ tıklayın ve Konsolu Dosyaya Kaydet seçeneğini belirleyin.
- Konsol sekmesindeki metni panoya kaydetmek için, Konsol sekmesini sağ tıklayın ve Konsolu Dosyaya Kaydet seçeneğini belirleyin. Panoya yalnızca seçili metni kopyalamak için, metni sağ tıklayın ve Kopyala seçeneğini belirleyin.
- Konsol sekmesindeki metni bir dosyaya kaydetmek için, Konsol sekmesini sağ tıklayın ve Konsolu Dosyaya Kaydet seçeneğini belirleyin.
- Sekmede görüntülenen eşleşen metni aratmak için Windows'ta CTRL+F veya Mac OS'de Command+F tuşunu tıklatın. (Görünür olmayan ağaç düğümleri aratılmaz.)

HTML sekmesi

HTML sekmesi HTML DOM'un tamamını bir ağaç yapısında görüntülemenize olanak tanır. Özelliklerini sekmenin sağ kısmında görüntülemek için öğeyi tıklatın. Ağaçtaki bir düğümü genişletmek veya daraltmak için + ve - simgelerini tıklatın.



HTML sekmesindeki herhangi bir nitelik veya metin öğesini düzenleyebilirsiniz ve düzenlenen değer uygulamaya yansır.

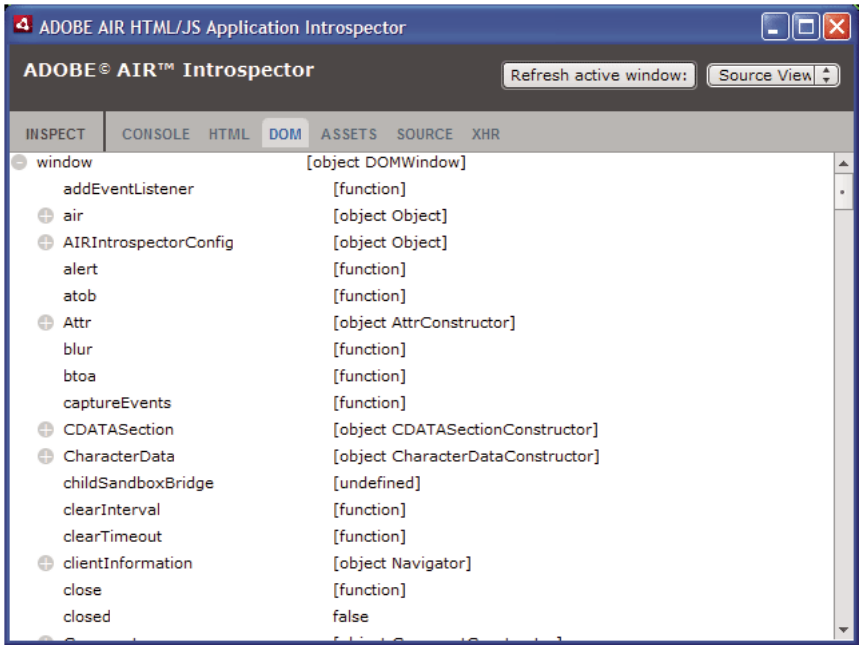
Denetle düğmesini tıklatın (AIR Introspector penceresindeki sekme listesinin solunda). Ana pencerenin HTML sayfasındaki herhangi bir öğeyi tıklatabilirsiniz ve ilişkili DOM nesnesi HTML sekmesinde görüntülenir. Ana pencere odağa sahip olduğunda, Denetleme düğmesini açıp kapatmak için de klavye kısayoluna basabilirsiniz. Klavye kısayolu varsayılan olarak F11'dir. Kısayol tuşunu F11 tuşundan başka bir tuş olarak yapılandırabilirsiniz; bkz. “AIR Introspector'ı Yapılandırma” sayfa 281.

HTML sekmesinde görüntülenen veriyi yenilemek için Etkin Pencereyi Yenile düğmesini (AIR Introspector penceresinin üst kısmında) tıklatın.

Sekmede görüntülenen metni eşleştirmek için Windows'ta CTRL+F veya Mac OS'de Command+F tuşunu tıklatın. (Görünür olmayan ağaç düğümleri aratılmaz.)

DOM sekmesi

DOM sekmesi window nesnesini bir ağaç yapısında gösterir. Herhangi bir dize ve sayısal özelliği düzenleyebilirsiniz ve düzenlenen değer uygulamaya yansır.

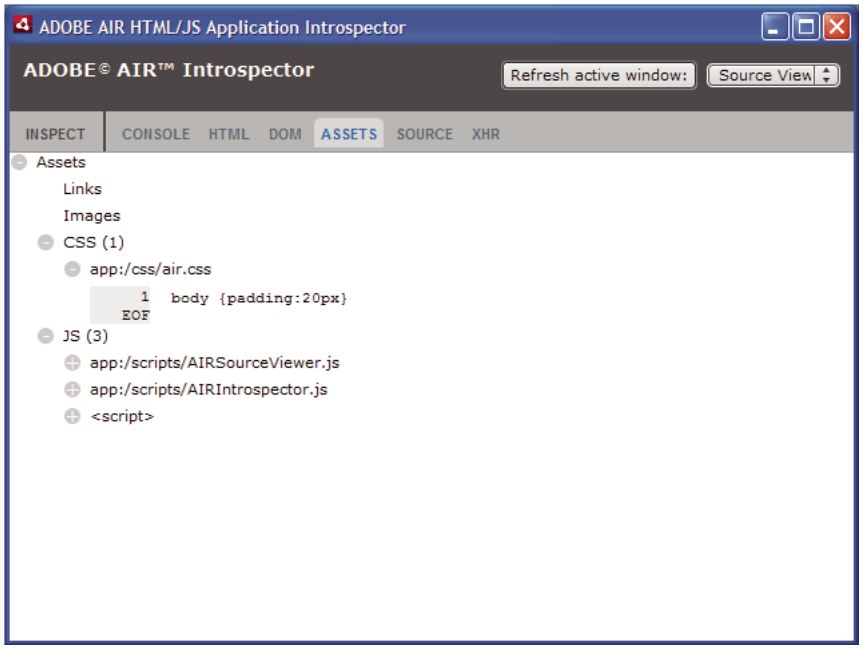


DOM sekmesinde görüntülenen veriyi yenilemek için Etkin Pencereyi Yenile düğmesini (AIR Introspector penceresinin üst kısmında) tıklatın.

Sekmede görüntülenen metni eşleştirmek için Windows'ta CTRL+F veya Mac OS'de Command+F tuşunu tıklatın. (Görünür olmayan ağaç düğümleri aratılmaz.)

Varlıklar sekmesi

Varlıklar sekmesi yerel pencerede yüklenen bağlantıları, görüntüleri, CSS'i ve JavaScript dosyalarını denetlemenize olanak tanır. Bu düğümlerden birini genişletmek dosyanın içeriğini gösterir veya kullanılan gerçek nesneyi görüntüler.



Varlıklar sekmesinde görüntülenen veriyi yenilemek için Etkin Pencereyi Yenile düğmesini (AIR Introspector penceresinin üst kısmında) tıklatın.

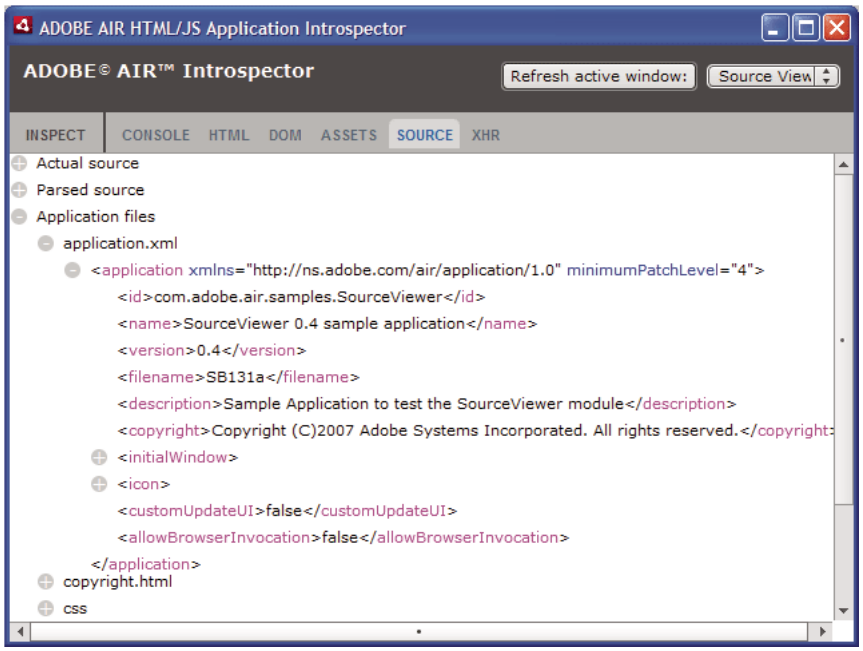
Sekmede görüntülenen metni eşleştirmek için Windows'ta CTRL+F veya Mac OS'de Command+F tuşunu tıklatın. (Görünür olmayan ağaç düğümleri aratılmaz.)

Kaynak sekmesi

Kaynak sekmesi üç bölüm içerir:

- Gerçek kaynak: Uygulama başladığında kök içerik olarak yüklenen sayfanın HTML kaynağını gösterir.
- Çağrılan kaynak: Uygulamanın işaretleme kodunu Ajax teknolojisini kullanarak anında oluşturmasından dolayı gerçek kaynaktan farklı olabilen uygulama arabirimini oluşturan geçerli işaretlemeyi gösterir.

- Uygulama dosyaları: Dosyaları uygulama dizininde listeler. Uygulama güvenlik sanal alanındaki içerikten başlatıldığında, bu liste yalnızca AIR Introspector için kullanılabilir. Bu bölümde, metin dosyalarının içeriğini veya görüntüleri inceleyebilirsiniz.

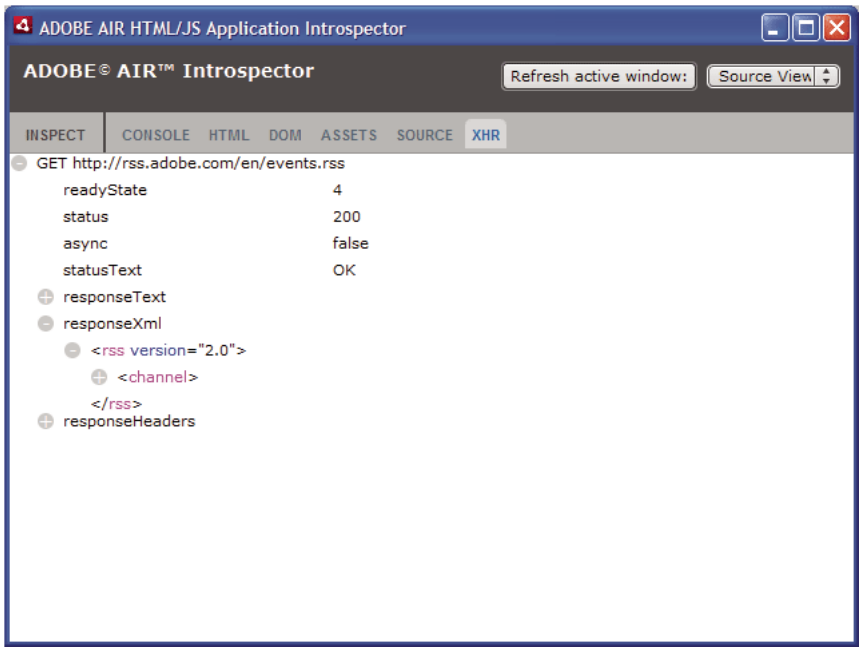


Kaynak sekmesinde görüntülenen veriyi yenilemek için Etkin Pencereyi Yenile düğmesini (AIR Introspector penceresinin üst kısmında) tıklatın.

Sekmede görüntülenen metni eşleştirmek için Windows'ta CTRL+F veya Mac OS'de Command+F tuşunu tıklatın. (Görünür olmayan ağaç düğümleri aratılmaz.)

XHR sekmesi

XHR sekmesi tüm XMLHttpRequest iletişimini uygulamada kesiştirir ve bilgiyi kaydeder. Bu, XMLHttpRequest özelliklerini `responseText` ve `responseXML` (kullanılabilir olduğunda) dahil olarak bir ağaç görünümünde görüntülemenize olanak tanır.



Sekmede görüntülenen metni eşleştirmek için Windows'ta CTRL+F veya Mac OS'de Command+F tuşunu tıklatın. (Görünür olmayan ağaç düğümleri aratılmaz.)

AIR Introspector'ı uygulama harici sanal alandaki içerikle kullanma

Uygulama dışı sanal alanıyla eşleşen bir iframe veya frame öğesine uygulama dizininden içerik yükleyebilirsiniz. (ActionScript geliştiricileri için Adobe AIR'de HTML güvenliği bölümüne veya HTML geliştiricileri için Adobe AIR'de HTML güvenliği bölümüne bakın). AIR introspector uygulamasını bu tip içerikle kullanabilirsiniz ancak aşağıdaki kuralları inceleyin:

- AIRIntrospector.js dosyası uygulama sanal alanı ve uygulama harici sanal alan (iframe) içeriklerinin ikisinde de bulunmalıdır.
- `parentSandboxBridge` özelliğinin üzerine yazmayın; AIR Introspector kodu bu özelliği kullanır. Gerektikçe özellikler ekleyin. O zaman aşağıdakileri yazmak yerine:

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)}} ;
  ```

  Aşağıdaki gibi bir sözdizimi kullanın:

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- AIR Introspector'ı F12 tuşuna basarak veya air.Introspector.Console sınıfındaki yöntemlerden birini çağırarak uygulama harici sanal alan içeriğinden açamazsınız. Introspector penceresini yalnızca Introspector'ı Aç düğmesine basarak açabilirsiniz. Düğme varsayılan olarak iframe veya karenin sağ üst köşesinde eklenir. (Uygulama harici sanal alan içeriğine uygulanan güvenlik kısıtlamalarından dolayı, yeni bir pencere yalnızca bir düğmeyi tıklatmak gibi bir kullanıcı hareketi sonucu açılabilir.)
- Uygulama sanal alanı ve uygulama harici sanal alanı için ayrı AIR Introspector pencereleri açabilirsiniz. AIR Introspector pencerelerinde görüntülenen başlığı kullanarak ikisini ayırt edebilirsiniz.
- AIR Introspector bir uygulama harici sanal alanda çalıştırıldığında Kaynak sekmesi uygulama dosyalarını görüntülemez.
- AIR Introspector, yalnızca açıldığı sanal alandaki koda bakabilir.

Bölüm 20: AIR uygulamalarını yerelleştirme

Adobe AIR 1.1 ve sonrası

Adobe® AIR® birden çok dil desteği içerir.

Actionscript 3.0 ve Flex çerçevesindeki içeriği yerelleştirmeye genel bakış için, ActionScript 3.0 Geliştirici Kılavuzu'ndaki "Uygulamaları yerelleştirme" bölümünü inceleyin.

AIR'de desteklenen diller

AIR uygulamaları için aşağıdaki dillerdeki yerelleştirme desteği AIR 1.1 sürümünde piyasaya sürülmüştür:

- Basitleştirilmiş Çince
- Geleneksel Çince
- Fransızca
- Almanca
- İtalyanca
- Japonca
- Korece
- Brezilya Portekizcesi
- Rusça
- İspanyolca

AIR 1.5 sürümünde, aşağıdaki diller eklendi:

- Çek dili
- Felemenkçe
- Lehçe
- İsveççe
- Türkçe

Daha fazla Yardım konusu

Adobe AIR'de çok dilli Flex uygulamaları oluşturma

Çok dilli bir HTML tabanlı uygulama oluşturma

Uygulama yükleyicisindeki AIR uygulamasının adını ve açıklamasını yerelleştirme

Adobe AIR 1.1 ve sonrası

Uygulama tanımlayıcı dosyasındaki `name` ve `description` öğeleri için birden çok dil belirtebilirsiniz. Örneğin aşağıdaki örnek, uygulama adını üç dilde belirtir (İngilizce, Fransızca ve Almanca):

```
<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>
```

Her metin öğesinin `xml:lang` niteliği, RFC4646'da da (http://www.ietf.org/rfc/rfc4646.txt) tanımlandığı gibi bir dil kodu belirtir.

Ad öğesi, AIR uygulama yükleyicisinin görüntülediği uygulama adını tanımlar. AIR uygulama yükleyicisi, işletim sistemi ayarları tarafından belirtilen kullanıcı arabirimi dilleriyle en iyi eşleşen yerelleştirilmiş değeri kullanır.

Benzer şekilde uygulama tanımlayıcı dosyasındaki `description` öğesinin de çok dilli sürümlerini belirtebilirsiniz. Bu öğe, AIR uygulama yükleyicisinin görüntülediği açıklama metnini tanımlar.

Bu ayarlar yalnızca AIR uygulama yükleyicisinde kullanılabilir olan diller için geçerlidir. Çalışan, yüklü uygulama için kullanılabilir olan yerel ayarları belirtmez. AIR uygulamaları, AIR uygulama yükleyicisinde kullanılabilir olanlar da dahil olmak üzere ve bu dillere ek olarak birden çok dili destekleyen kullanıcı arabirimleri sağlayabilir.

Daha fazla bilgi için bkz. “AIR uygulama tanımlayıcısı öğeleri” sayfa 205.

Daha fazla Yardım konusu

Adobe AIR'de çok dilli Flex uygulamaları oluşturma

Çok dilli bir HTML tabanlı uygulama oluşturma

HTML içeriğini AIR HTML yerelleştirme çerçevesiyle yerelleştirme

Adobe AIR 1.1 ve sonrası

AIR 1.1 SDK, bir HTML yerelleştirme çerçevesi içerir. AIRLocalizer.js JavaScript dosyası çerçeveyi tanımlar. AIR SDK'nin çerçeveler dizini, AIRLocalizer.js dosyasını içerir. Bu dosya, birden çok yerelleştirilmiş sürümü destekleyen uygulamalar oluşturma konusuna yardımcı olmak için işlevler sağlayan air.Localizer sınıfını içerir.

AIR HTML yerelleştirme çerçevesi kodunu yükleme

Yerelleştirme çerçevesini kullanmak için, AIRLocalizer.js dosyasını projenize kopyalayın. Ardından bu dosyayı bir komut dosyası etiketi kullanarak uygulamanın ana HTML dosyasına ekleyin:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

Sonraki JavaScript `air.Localizer.localizer` nesnesini çağırabilir:

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

`air.Localizer.localizer` nesnesi, yerelleştirilmiş kaynakları kullanma ve yönetme için yöntemler ve özellikler tanımlayan tek nesnedir. Localizer sınıfı aşağıdaki yöntemleri içerir:

Yöntem	Açıklama
`getFile()`	Belirli bir yerel ayar için belirtilen kaynak paketinin metnini alır. Bkz. "Belirli yerel ayarlar için kaynak alma" sayfa 297.
`getLocaleChain()`	Dilleri yerel ayar zincirinde döndürür. Bkz. "Yerel ayar zincirini tanımlama" sayfa 296.
`getResourceBundle()`	Paket anahtarlarını ve karşılık gelen değerleri nesne olarak döndürür. Bkz. "Belirli yerel ayarlar için kaynak alma" sayfa 297.
`getString()`	Bir kaynak için tanımlı olan dizeyi alır. Bkz. "Belirli yerel ayarlar için kaynak alma" sayfa 297.
`setBundlesDirectory()`	Paket dizini konumunu ayarlar. Bkz. "AIR HTML Localizer ayarlarını özelleştirme" sayfa 295.
`setLocalAttributePrefix()`	HTML DOM öğelerinde kullanılan yerelleştirici nitelikleri tarafından kullanılan öneki ayarlar. Bkz. "AIR HTML Localizer ayarlarını özelleştirme" sayfa 295
`setLocaleChain()`	Yerel ayar zincirinde dillerin sırasını ayarlar. Bkz. "Yerel ayar zincirini tanımlama" sayfa 296.
`sortLanguagesByPreference()`	Yerel ayar zincirindeki yerel ayarları, işletim sistemi ayarlarındaki yerel ayar sırasına bağlı olarak sıralar. Bkz. "Yerel ayar zincirini tanımlama" sayfa 296.
`update()`	HTML DOM'yi (veya bir DOM öğesini) geçerli yerel ayar zincirindeki yerelleştirilmiş dizelerle günceller. Yerel ayar zinciriyle ilgili bilgi için bkz. "Yerel ayar zincirlerini yönetme" sayfa 293. `update()` yöntemi hakkında daha fazla bilgi için, bkz. "DOM öğelerini geçerli yerel ayarı kullanmak üzere güncelleme" sayfa 294.

Localizer sınıfı aşağıdaki statik özellikleri içerir:

Özellik	Açıklama
`localizer`	Uygulama için tek Localizer nesnesine bir başvuru döndürür.
`ultimateFallbackLocale`	Uygulama hiçbir kullanıcı tercihini desteklemediğinde kullanılan yerel ayar. Bkz. "Yerel ayar zincirini tanımlama" sayfa 296.

Desteklenen dilleri belirtme

Uygulama tarafından desteklenen dilleri tanımlamak için uygulama tanımlayıcı dosyasındaki `<supportedLanguages>` öğesini kullanın. Bu öğe yalnızca iOS, Mac sabit çalışma zamanı ve Android uygulamaları tarafından kullanılır ve diğer tüm uygulama türleri tarafından yoksayılır.

`<supportedLanguages>` öğesini belirtmezseniz, varsayılan olarak paketleyici, uygulama türüne bağlı olarak aşağıdaki eylemleri gerçekleştirir:

- iOS — AIR çalışma zamanı tarafından desteklenen tüm diller iOS app store'da uygulamanın desteklediği diller olarak listelenir.
- Mac sabit çalışma zamanı — Sabit paketle paketlenmiş olan uygulamada yerelleştirme bilgisi yoktur.
- Android — Uygulama paketinde AIR çalışma zamanı tarafından desteklenen tüm dillere yönelik kaynaklar bulunur.

Daha fazla bilgi için bkz. "supportedLanguages" sayfa 235.

Kaynak paketlerini tanımlama

HTML yerelleştirme çerçevesi *yerelleştirme* dosyalarından dizelerin yerelleştirilmiş sürümlerini okur. Yerelleştirme dosyası, bir metin dosyasında serileştirilmiş anahtar tabanlı değerlerin bir koleksiyonudur. Yerelleştirme dosyasına bazen *paket* de denir.

Uygulama projesi dizininizin locale (yerel ayar) adlı bir alt dizinini oluşturun. (Farklı bir ad da kullanabilirsiniz, bkz. "AIR HTML Localizer ayarlarını özelleştirme" sayfa 295). Bu dizin yerelleştirme dosyalarını içerir. Bu dizin *paketler dizini* olarak bilinir.

Uygulamanızın desteklediği her yerel ayar için, paketler dizininin bir alt dizinini oluşturun. Her alt dizini yerel ayar koduyla eşleşecek şekilde adlandırın. Örneğin French (Fransız) alt dizinini "fr" ve English (İngiliz) alt dizinini "en" olarak adlandırın. Dil ve ülke kodu olan bir yerel ayarı tanımlamak için alt çizgi (_) karakterini kullanabilirsiniz. Örneğin U.S. English (Amerikan İngilizcesi) dizinini "en_us" olarak adlandırın. (Alternatif olarak alt çizgi yerine, "en-us" örneğinde olduğu gibi tire de kullanabilirsiniz. HTML yerelleştirme çerçevesi her ikisini de destekler.)

Bir yerel ayar alt dizinine istediğiniz sayıda kaynak dosya ekleyebilirsiniz. Genellikle her dil için bir yerelleştirme dosyası oluşturursunuz (ve dosyayı ilgili dilin dizinine yerleştirirsiniz). HTML yerelleştirme çerçevesi, bir dosyanın içeriğini okumanızı sağlayan bir `getFile()` yöntemi içerir (bkz. "Belirli yerel ayarlar için kaynak alma" sayfa 297.

.properties dosya uzantısına sahip dosyalar, yerelleştirme özellik dosyaları olarak bilinir. Bir yerel ayar için anahtar değer çiftleri tanımlamak üzere onları kullanabilirsiniz. Özellikler dosyası, her satırda bir dize değeri tanımlar. Örneğin aşağıdaki örnekte dize değeri `"Hello in English."greeting` adlı anahtar için tanımlanır:

```
greeting=Hello in English.
```

Aşağıdaki metni içeren bir özellikler dosyası, altı anahtar değer çifti tanımlar:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Bu örnek özellikler dosyasının, en dizininde saklanacak olan İngilizce sürümünü gösterir.

Bu Özellikler dosyasının Fransızca sürümü fr dizinine konur.

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Farklı bilgi türleri için birden fazla kaynak dosya tanımlayabilirsiniz. Örneğin bir legal.properties dosyası yasal standart metin (telif hakkı bilgileri gibi) içerebilir. Bu kaynağı bir çok uygulamada tekrardan kullanabilirsiniz. Benzer şekilde, kullanıcı arabiriminin farklı kısımları için yerelleştirilmiş içerik tanımlayan ayrı dosyalar tanımlayabilirsiniz.

Bu dosyalarda, birden fazla dili desteklemek için UTF-8 kodlaması kullanın.

Yerel ayar zincirlerini yönetme

Uygulamanız AIRLocalizer.js dosyasını yüklediğinde, uygulamanızda tanımlı olan yerel ayarları inceler. Bu yerel ayarlar, paketler dizininin alt dizinlerine karşılık gelir (bkz. “Kaynak paketlerini tanımlama” sayfa 292). Kullanılabilir yerel ayarların listesi *yerel ayar zinciri* olarak bilinir. AIRLocalizer.js dosyası yerel ayar zincirini, işletim sistemi ayarları tarafından tanımlanmış tercih edilen sırayı temel alarak sıralar. (`Capabilities.languages` özelliği, işletim sistemi kullanıcı arabirimi dillerini tercih edilen sırada listeler.)

Dolayısıyla bir uygulama "en", "en_US" ve "en_UK" yerel ayarları için kaynak tanımlarsa, AIR HTML Localizer çerçevesi yerel ayar zincirini uygun şekilde sıralar. Bir uygulama "en" öğesini birincil yerel ayar olarak bildiren bir sistemde başladığında, yerel ayar zinciri `["en", "en_US", "en_UK"]` şeklinde sıralanır. Bu durumda uygulama kaynakları önce "en" paketinde, sonra "en_US" paketinde arar.

Ancak sistem "en-US" öğesini birincil yerel ayar olarak bildirirse, sıralama `["en_US", "en", en_UK"]` şeklindedir. Bu durumda uygulama kaynakları önce "en_US" paketinde, sonra "en" paketinde arar.

Varsayılan olarak uygulama yerel ayar zincirindeki ilk yerel ayarı, kullanılacak varsayılan yerel ayar olarak tanımlar. Uygulamayı ilk kez çalıştırdıktan sonra kullanıcıdan bir yerel ayar seçmesini isteyebilirsiniz. Ardından seçimi bir tercihler dosyasında saklamayı ve uygulamanın bir sonraki başlatılışında bu yerel ayarı kullanmayı seçebilirsiniz.

Uygulamanız kaynak dizeleri yerel ayar zincirindeki herhangi bir yerel ayarda kullanabilir. Belirli bir yerel ayar bir kaynak dize tanımlamazsa, uygulama, yerel ayar zincirinde tanımlı olan diğer yerel ayarlar için bir sonraki eşleşen kaynak dizeyi kullanır.

Localizer nesnesinin `setLocaleChain()` yöntemini çağırarak yerel ayar zincirini özelleştirebilirsiniz. Bkz. “Yerel ayar zincirini tanımlama” sayfa 296.

DOM öğelerini yerelleştirilmiş içerikle güncelleme

Uygulamadaki bir öğe, yerelleştirme özellikleri dosyasındaki bir anahtar değere başvurabilir. Örneğin aşağıdaki örnekte bulunan `title` öğesi, bir `local_innerHTML` niteliğini belirtir. Yerelleştirme çerçevesi, yerelleştirilen bir değeri aramak için bu niteliği kullanır. Çerçeve varsayılan olarak `"local_"` ile başlayan nitelik adları arar. Çerçeve, `"local_"` ifadesinin arkasından gelen metinle eşleşen adlara sahip nitelikleri günceller. Bu durumda çerçeve, `title` öğesinin `innerHTML` niteliğini ayarlar. `innerHTML` niteliği, varsayılan özellikler dosyasındaki (default.properties) `mainWindowTitle` anahtarı için tanımlı olan değeri kullanır:

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Geçerli yerel ayar eşleşen bir değer tanımlamazsa, yerelleştirici çerçevesi yerel ayar zincirinin kalanını arar. Yerel ayar zincirinde bir değerin tanımlı olduğu bir sonraki yerel ayarı kullanır.

Aşağıdaki örnekte `p` öğesinin metni (`innerHTML` niteliği), varsayılan özellikler dosyasında tanımlanan `greeting` anahtarının değerini kullanır:

```
<p local_innerHTML="default.greeting" />
```

Aşağıdaki örnekte, `input` öğesinin değer niteliği (ve görüntülenen metin), varsayılan özellikler dosyasında tanımlanan `btnBlue` anahtarının değerini kullanır:

```
<input type="button" local_value="default.btnBlue" />
```

HTML DOM'yi geçerli yerel ayar zincirinde tanımlı olan dizeleri kullanmak üzere güncellemek için, Localizer nesnesinin `update()` yöntemini çağırın. `update()` yöntemini çağırmak Localizer nesnesinin DOM'yi ayrıştırmasına ve yerelleştirme (`"local_..."`) niteliklerini bulduğu konumda işlemeleri uygulamasına neden olur:

```
air.Localizer.localizer.update();
```

Hem bir nitelik ("innerHTML" gibi) hem de karşılık gelen yerelleştirme niteliği için ("local_innerHTML" gibi) değer tanımlayabilirsiniz. Bu durumda yerelleştirme çerçevesi, yerelleştirme zincirinde karşılık gelen bir değer bulursa, yalnızca nitelik değerinin üzerine yazar. Örneğin aşağıdaki öğe hem `value` hem de `local_value` niteliklerini tanımlar:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

Yalnızca belirli bir DOM öğesini de güncelleyebilirsiniz. Bir sonraki bölüm olan "DOM öğelerini geçerli yerel ayarı kullanmak üzere güncelleme" sayfa 294 bölümüne bakın.

AIR HTML Localizer `"local_"` ifadesini varsayılan olarak bir öğenin yerelleştirme ayarlarını tanımlayan nitelikler için önek olarak kullanır. Örneğin `local_innerHTML` niteliği varsayılan olarak bir öğenin `innerHTML` değeri için kullanılan paket ve kaynak adını tanımlar. Ayrıca `local_value` niteliği varsayılan olarak bir öğenin `value` niteliği için kullanılan paket ve kaynak adını da tanımlar. Localizer öğesini `"local_"` dışında bir nitelik öneki kullanmak üzere yapılandırabilirsiniz. Bkz. "AIR HTML Localizer ayarlarını özelleştirme" sayfa 295.

DOM öğelerini geçerli yerel ayarı kullanmak üzere güncelleme

Localizer nesnesi HTML DOM'yi güncellediğinde, işaretli öğelerin, geçerli yerel ayar zincirinde tanımlı dizelere dayalı olan nitelik değerlerini kullanmasına neden olur. HTML yerelleştiricisinin HTML DOM'yi güncellemesi için, `Localizer` nesnesinin `update()` yöntemini çağırın:

```
air.Localizer.localizer.update();
```

Yalnızca belirtilen bir DOM öğesini güncellemek için, `update()` yöntemine bir parametre olarak iletin. `update()` yöntemi yalnızca `parentNode` adlı, isteğe bağlı bir parametreye sahiptir. `parentNode` parametresi belirtildiğinde, yerelleştirilecek DOM öğesini tanımlar. `update()` yöntemini çağırmak ve bir `parentNode` parametresi belirtmek, yerelleştirme niteliklerini belirten tüm alt öğeler için yerelleştirilen değerleri ayarlar.

Örneğin aşağıdaki `div` öğesine bakın:

```
<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>
```

Bu öğeyi geçerli yerel ayar zincirinde tanımlı yerelleştirilmiş dizeleri kullanmak üzere güncellemek için, aşağıdaki JavaScript kodunu kullanın:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Yerel ayar zincirinde bir anahtar değer bulunmazsa, yerelleştirme çerçevesi nitelik değerini `"local_"` niteliğinin değerine ayarlar. Örneğin bir önceki örnekte, yerelleştirme çerçevesinin `lblColors` anahtarı için bir değer bulamadığını varsayın (yerel ayar zincirindeki hiçbir default.properties dosyasında). Bu durumda `innerHTML` değeri olarak `"default.lblColors"` öğesini kullanır. Bu değeri kullanmak eksik kaynaklar olduğunu (geliştiriciye) belirtir.

`update()` yöntemi, yerel ayar zincirinde kaynak bulamadığında bir `resourceNotFound` olayı gönderir. `air.Localizer.RESOURCE_NOT_FOUND` sabiti, `"resourceNotFound"` dizesini tanımlar. Olayın üç özelliği vardır: `bundleName`, `resourceName` ve `locale`. `bundleName` özelliği, kaynak bulunamayan paketin adıdır. `resourceName` özelliği, kaynak bulunamayan paketin adıdır. `locale` özelliği, kaynak bulunamayan yerel ayarın adıdır.

`update()` yöntemi, belirtilen paketi bulamadığında bir `bundleNotFound` olayı gönderir. `air.Localizer.BUNDLE_NOT_FOUND` sabiti, `"bundleNotFound"` dizesini tanımlar. Olayın iki özelliği vardır: `bundleName` ve `locale`. `bundleName` özelliği, kaynak bulunamayan paketin adıdır. `locale` özelliği, kaynak bulunamayan yerel ayarın adıdır.

`update()` yöntemi senkronize olmayan bir şekilde çalışır (ve `resourceNotFound` ve `bundleNotFound` olaylarını senkronize olmayan bir şekilde gönderir). Aşağıdaki kod `resourceNotFound` ve `bundleNotFound` olayları için olay dinleyicilerini ayarlar:

```
air.Localizer.localizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.update();
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

AIR HTML Localizer ayarlarını özelleştirme

Localizer nesnesinin `setBundlesDirectory()` yöntemi sayesinde paketlerin dizin yolunu özelleştirebilirsiniz. Localizer nesnesinin `setLocalAttributePrefix()` yöntemi sayesinde paketlerin dizin yolunu ve Localizer tarafından kullanılan nitelik değerini özelleştirebilirsiniz.

Varsayılan paketler dizini, uygulama dizininin yerel ayar alt dizini olarak tanımlanır. Localizer nesnesinin `setBundlesDirectory()` yöntemini çağırarak başka bir dizin de belirtebilirsiniz. Bu yöntem için `path` adlı bir parametre gereklidir, bu parametre, bir dize olarak istenilen paketler dizinine giden yoldur. `path` parametresinin değeri şunlardan biri olabilir:

- `"locales"` gibi, uygulama dizinine göreceli bir yol tanımlayan bir Dize
- `"app://languages"` gibi, `app`, `app-storage` veya `file` URL şemalarını kullanan geçerli bir URL tanımlayan bir Dize (`http` URL şemasını *kullanmayın*)
- Bir File nesnesi

URL'ler ve dizin yolları hakkında bilgi almak için bkz:

- File nesnelerinin yolları (ActionScript geliştiricileri için)
- File nesnelerinin yolları (HTML geliştiricileri için)

Örneğin aşağıdaki kod paketler dizinini, uygulama depolama dizininin (uygulama dizininin değil) diller alt dizinine ayarlar:

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

`path` parametresi olarak geçerli bir yol iletin. Aksi takdirde yöntem bir BundlePathNotFoundError istisnası atar. Hatanın `name` özelliği `"BundlePathNotFoundError"` öğesidir ve `message` özelliği geçersiz yolu belirtir.

AIR HTML Localizer `"local_"` ifadesini varsayılan olarak bir öğenin yerelleştirme ayarlarını tanımlayan nitelikler için önek olarak kullanır. Örneğin `local_innerHTML` niteliği, aşağıdaki `input` öğesinin `innerHTML` değeri için kullanılan paket ve kaynak adını tanımlar:

```
<p local_innerHTML="default.greeting" />
```

Localizer nesnesinin `setLocalAttributePrefix()` yöntemi sayesinde, `"local_"` dışında bir nitelik öneki kullanabilirsiniz. Bu statik yöntem için, nitelik öneki olarak kullanmak istediğiniz dize olan bir parametre gereklidir. Örneğin aşağıdaki kod, yerelleştirme çerçevesini, "loc_" ifadesini nitelik öneki olarak kullanmak üzere ayarlar:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Yerelleştirme çerçevesinin kullandığı nitelik önekini özelleştirebilirsiniz. Varsayılan değer (`"local_"`), kodunuzun kullandığı başka bir nitelikle çakışıyorsa, öneki özelleştirmek isteyebilirsiniz. Bu yöntemi çağırırken, HTML nitelikleri için geçerli karakterler kullandığınızdan emin olun. (Örneğin değer boşluk karakteri içermemelidir.)

Yerelleştirme niteliklerini HTML öğelerinde kullanma hakkında bilgi için, bkz. "DOM öğelerini yerelleştirilmiş içerikle güncelleme" sayfa 293.

Paketler dizini ve nitelik öneki ayarları, farklı uygulama oturumları arasında kalıcı değildir. Özel paketler dizini veya nitelik öneki ayarı kullanıyorsanız, uygulama her başladığında ayarı yeniden yaptığınızdan emin olun.

Yerel ayar zincirini tanımlama

AIRLocalizer.js kodunu yüklediğinizde, varsayılan olarak varsayılan yerel ayar zincirini ayarlar. Paketler dizininde ve işletim sistemi dil ayarlarında bulunan yerel ayarlar, bu yerel ayar zincirini tanımlar. (Ayrıntılar için bkz. "Yerel ayar zincirlerini yönetme" sayfa 293.)

Localizer nesnesinin statik `setLocaleChain()` yöntemini çağırarak, yerel ayar zincirini değiştirebilirsiniz. Örneğin, kullanıcı belirli bir dil için bir tercih belirtirse, bu yöntemi çağırmak isteyebilirsiniz. `setLocaleChain()` yöntemi için `chain` adlı bir parametre gereklidir, bu parametre `["fr_FR","fr","fr_CA"]` gibi bir yerel ayarlar dizisidir. Yerel ayarların dizideki sırası, çerçevenin kaynak arama sırasını belirler (sonraki işlemlerde). Zincirdeki ilk yerel ayar için kaynak bulunmazsa, diğer yerel ayarların kaynaklarına bakmaya devam eder. `chain` argümanı eksikse, bir dizi değilse veya boş bir diziyse, işlev başarısız olur ve bir IllegalArgumentsError istisnası atar.

Localizer nesnesinin statik `getLocaleChain()` yöntemi, geçerli yerel ayar zincirindeki yerel ayarları listeleyen bir Dizi döndürür.

Aşağıdaki kod geçerli yerel ayar zincirini okur ve zincirin başına iki Fransız yerel ayar ekler:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

`setLocaleChain()` yöntemi yerel ayar zincirini güncellediğinde bir `"change"` olayı gönderir. `air.Localizer.LOCALE_CHANGE` sabiti, `"change"` dizesini tanımlar. Olayın `localeChain` adlı bir özelliği vardır, bu, yeni yerel ayar zincirindeki yerel ayar kodlarının bir dizisidir. Aşağıdaki kod bu olay için bir olay dinleyicisi ayarlar:

```
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);
}
```

Statik `air.Localizer.ultimateFallbackLocale` özelliği, uygulama hiçbir kullanıcı tercihini desteklemediğinde kullanılan yerel ayarı temsil eder. Varsayılan olarak `"en"` değerindedir. Aşağıdaki kodda gösterildiği gibi, başka bir yerel ayara ayarlayabilirsiniz:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Belirli yerel ayarlar için kaynak alma

Localizer nesnesinin `getString()` yöntemi, belirli bir yerel ayardaki kaynak için tanımlı olan dizeyi döndürür. Yöntemi çağırırken bir `locale` değeri belirtmeniz gerekmez. Bu durumda yöntem yerel ayar zincirinin tamamına bakar ve belirli kaynak adını sağlayan ilk yerel ayardaki dizeyi döndürür. Bu yöntem aşağıdaki parametrelere sahiptir:

Parametre	Açıklama
`bundleName`	Kaynağı içeren paket. Bu, .properties uzantısı olmadan özellikler dosyasının dosya adıdır. (Örneğin bu parametre `"alerts"` olarak ayarlanırsa, Localizer kodu alerts.properties adlı yerelleştirme dosyalarına bakar.
`resourceName`	Kaynak adı.
`templateArgs`	İsteğe bağlı. Değiştirme dizesindeki numaralı etiketleri değiştirmek için dize dizisi. Örneğin `templateArgs` parametresinin `["Raúl", "4"]` ve eşleşen kaynak dizenin `"Hello, {0}. You have {1} new messages."`. Bu durumda işlev şunu döndürür: `"Hello, Raúl. You have 4 new messages."`. Bu ayarı yok saymak için, bir `null` değeri iletin.
`locale`	İsteğe bağlı. Kullanılacak yerel ayar kodu (`"en"`, `"en_us"` veya `"fr"` gibi). Bir yerel ayar sağlanmışsa ve eşleşen değer bulunmazsa, yöntem değerleri yerel ayar zincirindeki diğer yerel ayarlarda aramaya devam etmez. Yerel kod belirtilmemişse işlev, belirli kaynak adı için bir değer sağlayan yerel ayar zincirindeki ilk yerel ayardaki dizeyi döndürür.

Yerelleştirme çerçevesi işaretli HTML DOM niteliklerini güncelleyebilir. Ancak yerelleştirilmiş dizeleri başka şekillerde kullanabilirsiniz. Örneğin bir dizeyi dinamik olarak oluşturulmuş bir HTML'de veya bir işlev çağrısında parametre değeri olarak kullanabilirsiniz. Örneğin aşağıdaki kod `alert()` işlevini, fr_FR yerel ayarının varsayılan özellikler dosyasındaki `error114` kaynağında tanımlı olan dizeyle çağırır:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

`getString()` yöntemi, kaynağı belirtilen pakette bulamadığında bir `resourceNotFound` olayı gönderir. `air.Localizer.RESOURCE_NOT_FOUND` sabiti, `"resourceNotFound"` dizesini tanımlar. Olayın üç özelliği vardır: `bundleName`, `resourceName` ve `locale`. `bundleName` özelliği, kaynak bulunamayan paketin adıdır. `resourceName` özelliği, kaynak bulunamayan paketin adıdır. `locale` özelliği, kaynak bulunamayan yerel ayarın adıdır.

`getString()` yöntemi, belirtilen paketi bulamadığında bir `bundleNotFound` olayı gönderir. `air.Localizer.BUNDLE_NOT_FOUND` sabiti, `"bundleNotFound"` dizesini tanımlar. Olayın iki özelliği vardır: `bundleName` ve `locale`. `bundleName` özelliği, kaynak bulunamayan paketin adıdır. `locale` özelliği, kaynak bulunamayan yerel ayarın adıdır.

`getString()` yöntemi senkronize olmayan bir şekilde çalışır (ve `resourceNotFound` ve `bundleNotFound` olaylarını senkronize olmayan bir şekilde gönderir). Aşağıdaki kod `resourceNotFound` ve `bundleNotFound` olayları için olay dinleyicilerini ayarlar:

```
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 nesnesinin `getResourceBundle()` yöntemi, belirli bir yerel ayar için belirli bir paket döndürür. Yöntemin dönüş değeri, paketteki anahtarlarla eşleşen özelliklere sahip bir nesnedir. (Uygulama belirtilen paketi bulamazsa, yöntem `null` değerini döndürür.)

Yöntem iki parametre alır—`locale` ve `bundleName`.

Parametre	Açıklama
`locale`	Yerel ayar (örn. `"fr"`).
`bundleName`	Paket adı.

Örneğin, aşağıdaki kod fr yerel ayarı için varsayılan paketi yüklemek için `document.write()` yöntemini çağırır. Daha sonra, bu paketteki `str1` ve `str2` anahtarlarının değerlerini yazmak için `document.write()` yöntemini çağırır:

```
var aboutWin = window.open();
var bundle = localizer.getResourceBundle("fr", "default");
aboutWin.document.write(bundle.str1);
aboutWin.document.write("<br/>");
aboutWin.document.write(bundle.str2);
aboutWin.document.write("<br/>");
```

`getResourceBundle()` yöntemi, belirtilen paketi bulamadığında bir `bundleNotFound` olayı gönderir. `air.Localizer.BUNDLE_NOT_FOUND` sabiti, `"bundleNotFound"` dizesini tanımlar. Olayın iki özelliği vardır: `bundleName` ve `locale`. `bundleName` özelliği, kaynak bulunamayan paketin adıdır. `locale` özelliği, kaynak bulunamayan yerel ayarın adıdır.

Localizer nesnesinin `getFile()` yöntemi, belirli bir yerel ayar için bir paketin içeriğini dize olarak döndürür. Paket dosyası UTF-8 dosyası olarak okunur. Bu yöntem aşağıdaki parametreleri içerir:

Parametre	Açıklama
`resourceFileName`	Kaynak dosyanın dosya adı (`"about.html"` gibi).

`templateArgs`	İsteğe bağlı. Değiştirme dizesindeki numaralı etiketleri değiştirmek için dize dizisi. Örneğin `templateArgs` parametresinin `["Raúl", "4"]` olduğu ve eşleşen kaynak dizenin iki satır içerdiği bir işleve yapılacak çağrıyı düşünün:
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	Bu durumda işlev iki satırlı bir dize döndürür:
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

`locale`	Kullanılacak yerel ayar, örneğin `"en_GB"` gibi. Bir yerel ayar sağlanmışsa ve eşleşen dosya bulunmazsa, yöntem yerel ayar zincirindeki diğer yerel ayarlarda aramaya devam etmez. *Hiçbir* yerel kod belirtilmemişse işlev, `resourceFileName` öğesiyle eşleşen bir dosyaya sahip olan yerel ayar zincirindeki ilk yerel ayardaki metni döndürür.

Örneğin aşağıdaki kod, fr yerel ayarının about.html dosyasının içeriğini kullanarak `document.write()` yöntemini çağırır:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

`getFile()` yöntemi, yerel ayar zincirinde kaynak bulamadığında bir `fileNotFound` olayı gönderir. `air.Localizer.FILE_NOT_FOUND` sabiti, `"resourceNotFound"` dizesini tanımlar. `getFile()` yöntemi senkronize olmayan bir şekilde çalışır (ve `fileNotFound` olayını senkronize olmayan bir şekilde gönderir). Olayın iki özelliği vardır: `fileName` ve `locale`. `fileName` özelliği, bulunamayan dosyanın adıdır. `locale` özelliği, kaynak bulunamayan yerel ayarın adıdır. Aşağıdaki kod bu olay için bir olay dinleyicisi ayarlar:

```
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);
}
```

Daha fazla Yardım konusu

Çok dilli bir HTML tabanlı uygulama oluşturma

Bölüm 21: Path ortam değişkenleri

AIR SDK bir komut satırından veya terminalden başlatılabilecek birkaç program içerir. Bu programları SDK bin dizini yolu, path ortam değişkenine dahil olduğunda çalıştırmak genelde daha uygun olabilir.

Burada sunulan bilgiler Windows, Mac ve Linux'ta yolun nasıl ayarlanacağını açıklar ve kullanışlı bir kılavuzdur. Ancak, bilgisayar konfigürasyonları çok fazla çeşitlilik gösterir. Bu nedenle bu yordam her sistem için uygun olmayabilir. Bu durumda, gerekli bilgileri işletim sistemi belgelerinizden veya İnternet'ten bulabilmeniz gerekir.

Bash kabuğunu kullanarak Linux ve Mac OS'de PATH değişkenini ayarlama

Bir terminal penceresine kodu yazdığınızda, yazdığınızı okuyan ve uygun bir şekilde yanıt vermeye çalışan program olan kabuk, önce dosya sisteminde komut programını bulmalıdır. Kabuk, $PATH adlı ortam değişkeninde saklanan dizinler listesinde komutları arar. Yolda o anda nelerin listelenmiş olduğunu görmek için şunu yazın:

```
echo $PATH
```

Bu aşağıdaki gibi görünen, iki nokta üst üste işaretiyle ayrılmış dizin listesini döndürür:

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

Amaç kabuğun ADT ve ADL araçlarını bulabilmesi için yolu AIR SDK bin dizini listesine eklemektir. AIR SDK'yi `/Users/fred/SDKs/AIR` konumuna yerleştirdiğinizi varsayarsak, aşağıdaki komut yola gerekli dizinleri ekleyecektir:

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Not: *Yolunuz boşluk karakterleri içeriyorsa şurada olduğu gibi bu karakterlerin önüne ters eğik çizgi yerleştirin:*

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

İşe yaradığından emin olmak için `echo` komutunu tekrar kullanabilirsiniz:

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDK
s/android/tools
```

Şimdilik her şey yolunda. Artık aşağıdaki komutları yazdığınızda umut verici yanıtlar alabilmeniz gerekir:

```
adt -version
```

$PATH değişkenini doğru şekilde değiştirdiyseniz komut ADT sürümünü bildirmelidir.

Ancak hala bir sorun var; bir daha yeni bir terminal penceresi açışınızda, yoldaki yeni girişlerin orada olmadığını göreceksiniz. Her yeni terminal açışınızda yolu ayarlama komutu çalıştırılmalıdır.

Bu sorunun yaygın çözümlerinden biri komutu kabuğunuzun kullandığı başlangıç komut dosyalarından birine eklemektir. Mac OS'de, ~/username dizininde .bash_profile dosyasını oluşturabilirsiniz. Her yeni terminal penceresi açışınızda çalıştırılır. Ubuntu'da, yeni bir terminal penceresi açtığınızda çalışan başlangıç komut dosyası .bashrc dosyasıdır. Diğer Linux dağıtımları ve kabuk programlarının benzer kuralları vardır.

Komutu kabuk başlangıç komut dosyasına eklemek için:

1 Giriş dizininize değiştirin:

```
cd
```

2 Kabuk konfigürasyon profilini (gerekirse) oluşturun ve yazdığınız metni "`cat >>`" ile dosyanın sonuna yeniden yönlendirin. İşletim sisteminiz ve kabuğunuz için uygun olan dosyayı kullanın. Örneğin Mac OS'de `.bash_profile` dosyasını ve Ubuntu'da `.bashrc` dosyasını kullanabilirsiniz.

```
cat >> .bash_profile
```

3 Dosyaya eklenecek metni girin:

```
export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
```

4 Klavyede `CTRL-SHIFT-D` tuşlarına basarak metin yeniden yönlendirmesini sonlandırın.

5 Her şeyin doğru olduğundan emin olmak için dosyayı görüntüleyin:

```
cat .bash_profile
```

6 Yolu kontrol etmek için yeni bir terminal penceresi açın:

```
echo $PATH
```

Eklediğiniz yollar listelenmelidir.

Daha sonra farklı bir dizinde SDK'lerden birinin yeni bir sürümünü oluşturursanız, konfigürasyon dosyasında path komutunu güncellediğinizden emin olun. Aksi takdirde, kabuk eski sürümü kullanmaya devam eder.

Windows'ta Path değişkenini ayarlama

Windows'ta bir komut penceresi açtığınızda, o pencere sistem özelliklerinde tanımlanan global ortam değişkenlerini devralır. Önemli değişkenlerden biri, çalıştırmak için bir programın adını yazdığınızda komut programının arama yaptığı dizin listesi olan path değişkenidir. Bir komut penceresini kullanırken path değişkenine geçerli olarak nelerin dahil olduğunu görmek için şunu yazabilirsiniz:

```
set path
```

Bu aşağıdaki gibi görünen noktalı virgülle ayrılmış dizin listesini gösterir:

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

Amaç komut programının ADT ve ADL araçlarını bulabilmesi için path değişkenini AIR SDK bin dizini listesine eklemektir. AIR SDK'sini `C:\SDKs\AIR` konumuna yerleştirdiğinizi varsayarsak, doğru yol girişini aşağıdaki yordamla ekleyebilirsiniz:

1 Denetim Masasından veya Bilgisayarım simgesini sağ tıklatıp menüden Özellikler öğesini seçerek Sistem Özellikleri iletişim kutusunu açın.

2 Gelişmiş sekmesinde, Ortam Değişkenleri düğmesini tıklatın.

3 Ortam Değişkenleri iletişim kutusunun Sistem değişkenleri bölümünde Path girişini seçin

4 Düzen'i tıklatın.

5 Değişken değeri alanında metnin sonuna kaydırın.

6 Geçerli değerin en sonuna aşağıdaki metni girin:

```
;C:\SDKs\AIR\bin
```

7 Yolu kaydetmek için tüm iletişim kutularında Tamam'ı tıklatın.

Açık komut penceresi varsa ortamların güncellenmediğinin farkına varın. Yeni bir komut penceresi açın ve yolların doğru olarak ayarlandığından emin olmak için şu komutu yazın:

```
adt -version
```

AIR SDK'nin konumunu daha sonra değiştirirseniz veya yeni bir sürüm eklerseniz, path değişkenini güncellemeyi unutmayın.