

Creación de aplicaciones de ADOBE AIR®



Avisos legales

Para ver los avisos legales, consulte http://help.adobe.com/es_ES/legalnotices/index.html.

Contenidos

Capítulo 1: Acerca de Adobe AIR

Capítulo 2: Instalación de Adobe AIR

Instalación de Adobe AIR	3
Desinstalación de Adobe AIR	5
Instalación y ejecución de las aplicaciones de AIR de muestra	6
Actualizaciones de Adobe AIR	6

Capítulo 3: Trabajo con las API de AIR

Clases de ActionScript 3.0 específicas de AIR	8
Clases de Flash Player con funcionalidad específica de AIR	13
Componentes de Flex específicos de AIR	16

Capítulo 4: Herramientas de la plataforma de Adobe Flash para el desarrollo de AIR

Instalación del SDK de AIR	18
Configuración del SDK de Flex	20
Configuración de los SDK externos	21

Capítulo 5: Creación de su primera aplicación de AIR

Creación de su primera aplicación de AIR de Flex de escritorio en Flash Builder	22
Creación de la primera aplicación de AIR de escritorio con Flash Professional	26
Creación de su primera aplicación de AIR para Android en Flash Professional	28
Creación de su primera aplicación de AIR para iOS	29
Creación de la primera aplicación de AIR basada en HTML con Dreamweaver	33
Creación de la primera aplicación de AIR basada en HTML con el SDK de AIR	36
Creación de la primera aplicación de AIR de escritorio con el SDK de Flex	40
Creación de la primera aplicación de AIR para Android con el SDK de Flex	44

Capítulo 6: Desarrollo de aplicaciones de AIR para el escritorio

Flujo de trabajo para el desarrollo de una aplicación de escritorio de AIR	49
Configuración de las propiedades de una aplicación de escritorio	50
Depuración de una aplicación de AIR de escritorio	55
Empaquetado de un archivo de instalación de AIR de escritorio	57
Empaquetado de un instalador nativo de escritorio	60
Empaquetado de un paquete de motor de ejecución captador para equipos de escritorio	64
Distribución de paquetes de AIR para equipos de escritorio	67

Capítulo 7: Desarrollo de aplicaciones de AIR para dispositivos móviles

Configuración del entorno de desarrollo	70
Consideraciones sobre el diseño de aplicaciones móviles	71
Flujo de trabajo para crear aplicaciones de AIR para dispositivos móviles	75
Configuración de las propiedades de una aplicación móvil	76
Empaquetado de una aplicación de AIR móvil	99
Depuración de una aplicación de AIR móvil	106

Contenidos

Instalación de AIR y aplicaciones de AIR en dispositivos móviles	114
Actualización de aplicaciones móviles de AIR	118
Uso de notificaciones push	118
 Capítulo 8: Desarrollo de aplicaciones de AIR para dispositivos de televisión	
Funciones de AIR para TV	128
Consideraciones de diseño de las aplicaciones de AIR para TV	130
Flujo de trabajo para desarrollar una aplicación de AIR para TV	145
Propiedades del descriptor de la aplicación de AIR para TV	148
Empaquetado de una aplicación de AIR para TV	151
Depuración de aplicaciones de AIR para TV	153
 Capítulo 9: Uso de extensiones nativas para Adobe AIR	
Archivos de extensión nativa de AIR (ANE)	157
Extensiones nativas y la clase NativeProcess de ActionScript	158
Extensiones nativas y clases de biblioteca de ActionScript (archivos SWC)	158
Dispositivos admitidos	158
Perfiles de dispositivos admitidos	159
Lista de tareas para utilizar una extensión nativa	159
Declaración de la extensión en el archivo descriptor de la aplicación	159
Inclusión del archivo ANE en la ruta de la biblioteca de la aplicación	160
Empaquetado de una aplicación con extensiones nativas	161
 Capítulo 10: Compiladores de ActionScript	
Información sobre las herramientas de la línea de comandos de AIR en el SDK de Flex	163
Configuración del compilador	163
Compilación de archivos de origen MXML y ActionScript para AIR	164
Compilación de una biblioteca de código o componente de AIR (Flex)	166
 Capítulo 11: AIR Debug Launcher (ADL)	
Uso de ADL	168
Ejemplos con ADL	171
Códigos de error y de salida de ADL	172
 Capítulo 12: AIR Developer Tool (ADT)	
Comandos de ADT	174
Conjuntos de opciones de ADT	189
Mensajes de error de ADT	193
Variables del entorno de ADT	198
 Capítulo 13: Firma de aplicaciones de AIR	
Firma digital de archivos de AIR	199
Creación de archivos intermedios sin firmar de AIR con ADT	208
Firma de un archivo intermedio de AIR con ADT	208
Firma de una versión actualizada de una aplicación de AIR	209
Creación de certificados con firma automática con ADT	213

Capítulo 14: Archivos descriptores de las aplicaciones de AIR

Cambios en el descriptor de la aplicación	215
Estructura del archivo descriptor de la aplicación	217
Elementos del descriptor de la aplicación de AIR	218

Capítulo 15: Perfiles de dispositivo

Restricción de perfiles de destino en el archivo descriptor de la aplicación	257
Capacidades en diferentes perfiles	257

Capítulo 16: API en navegador AIR.SWF

Personalización del archivo badge.swf de instalación integrada	261
Utilización del archivo badge.swf para instalar una aplicación de AIR	261
Carga del archivo air.swf	265
Cómo comprobar si está instalado el motor de ejecución	265
Cómo comprobar desde una página web si una aplicación de AIR está instalada	266
Instalación de una aplicación de AIR desde el navegador	267
Inicio desde el navegador de una aplicación de AIR instalada	268

Capítulo 17: Actualización de aplicaciones de AIR

Actualización de aplicaciones	271
Presentación de una interfaz de usuario personalizada para las actualizaciones de la aplicación	273
Descarga de un archivo de AIR en el equipo del usuario	273
Comprobar si una aplicación se está ejecutando por primera vez	275
Utilización del marco de actualización	277

Capítulo 18: Visualización de código fuente

Carga, configuración y apertura del visor de código fuente	291
Interfaz de usuario del visor de código fuente	294

Capítulo 19: Depuración con el introspector HTML de AIR

Introspector de AIR	295
Carga del código del introspector de AIR	295
Inspección de un objeto en la ficha Console (Consola)	296
Configuración del introspector de AIR	298
Interfaz del introspector de AIR	298
Utilización del introspector de AIR con contenido en un entorno limitado ajeno a la aplicación	305

Capítulo 20: Localización de aplicaciones de AIR

Localización del nombre y la descripción en el instalador de aplicaciones de AIR	308
Localización de contenido HTML con la arquitectura de localización de HTML de AIR	308

Capítulo 21: Variables del entorno de ruta

Configuración de la ruta en Linux y Mac OS utilizando el shell Bash	319
Configuración de la ruta en Windows	320

Capítulo 1: Acerca de Adobe AIR

Adobe® AIR® es un motor de ejecución multipantalla válido para todos los sistemas operativos que le permite aprovechar sus habilidades de desarrollo web para crear e implementar aplicaciones enriquecidas de Internet (RIA) en el escritorio y dispositivos móviles. Las aplicaciones de AIR móviles, de televisión y escritorio se pueden crear con ActionScript 3.0 utilizando Adobe® Flex y Adobe® Flash® (basado en SWF). Las aplicaciones de AIR de escritorio también se pueden crear con HTML, JavaScript® y Ajax (basado en HTML).

Para obtener más información sobre el uso y una introducción a Adobe AIR, consulte Adobe AIR Developer Connection <http://www.adobe.com/devnet/air/> (Centro de desarrollo de Adobe AIR) (en inglés).

AIR permite el trabajo en entornos conocidos para aprovechar las herramientas y los procesos con los que se encuentra más cómodo. Al admitir Flash, Flex, HTML, JavaScript y Ajax, es posible obtener la mejor experiencia posible que se adapte a sus necesidades.

Por ejemplo, se pueden desarrollar aplicaciones utilizando una de las tecnologías siguientes o combinando varias de ellas:

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

Los usuarios interactúan con las aplicaciones de AIR del mismo modo que con las aplicaciones nativas. El motor de ejecución se instala una vez en el ordenador del usuario o dispositivo y después se instalan y ejecutan las aplicaciones de AIR como cualquier otra aplicación de escritorio. (En iOS, no se instala un motor de ejecución de AIR independiente; todas las aplicaciones de AIR de iOS son independientes.)

El motor de ejecución ofrece una arquitectura y plataforma compatibles con distintos sistemas operativos para la implementación de aplicaciones. La compatibilidad y constancia del funcionamiento y las interacciones en distintos escritorios obvia la necesidad de realizar pruebas en distintos navegadores. En lugar de desarrollar programas para un sistema operativo determinado, el desarrollador centra sus esfuerzos en el motor de ejecución, lo cual ofrece las siguientes ventajas:

- Las aplicaciones desarrolladas para AIR se ejecutan en varios sistemas operativos distintos sin suponer trabajo adicional para el desarrollador. El motor de ejecución asegura una presentación e interacciones constantes y predecibles en todos los sistemas operativos compatibles con AIR.
- Las aplicaciones se pueden crear de forma más rápida permitiendo el aprovechamiento de tecnologías web y patrones de diseño existentes. Las aplicaciones basadas en web se pueden ampliar al escritorio sin tener que aprender las tecnologías de desarrollo en escritorio tradicionales o la complejidad del código nativo.
- El desarrollo de aplicaciones resulta más fácil que cuando se utilizan lenguajes de nivel inferior como C y C++. No hace falta gestionar las complejas API de nivel inferior que son específicas para cada sistema operativo.

Al desarrollar aplicaciones para AIR se puede aprovechar un juego enriquecido de arquitecturas e interfaces API:

- API específicas para AIR proporcionadas por el motor de ejecución y la arquitectura de AIR
- API de ActionScript utilizadas en archivos SWF y la arquitectura de Flex (además de otras bibliotecas y arquitecturas basadas en ActionScript)
- HTML, CSS y JavaScript
- La mayoría de las arquitecturas de Ajax

- Las extensiones nativas para Adobe AIR proporcionan API de ActionScript que permiten acceder a funciones específicas de la plataforma programadas con código nativo. Las extensiones nativas también proporcionan acceso al código nativo heredado y el código nativo siempre aporta mayor rendimiento.

AIR es toda una novedad en la forma de crear, implementar y experimentar las aplicaciones. Puede obtener más control creativo y ampliar las aplicaciones basadas en Flash, Flex, HTML y Ajax en su escritorio, dispositivos móviles y televisiones.

Para obtener información sobre los elementos incluidos en cada nueva actualización de AIR, consulte las notas de la versión de Adobe AIR (http://www.adobe.com/go/learn_air_relnotes_es).

Capítulo 2: Instalación de Adobe AIR

El motor de ejecución de Adobe® AIR® permite ejecutar aplicaciones de AIR. El motor de ejecución se puede instalar de cualquiera de las formas siguientes:

- Mediante la instalación independiente del motor de ejecución (sin instalar además una aplicación de AIR).
- Con la instalación de una aplicación de AIR por primera vez mediante una instalación de página web (“instalación desde navegador”) (se le indicará que también instale el motor de ejecución).
- Mediante la creación de un archivo de instalación que instale tanto la aplicación como el motor de ejecución. Debe obtener la aprobación de Adobe para poder distribuir el motor de ejecución de AIR de esta manera. Puede solicitar la aprobación en la página [Licencias del motor de ejecución de Adobe](#) (en inglés). Tenga en cuenta que Adobe no proporciona herramientas para crear este archivo de instalación. No obstante, hay disponibles muchos kits de herramientas de archivos de instalación de otros fabricantes.
- Mediante la instalación de una aplicación de AIR que contenga AIR como motor de ejecución captador. El motor de ejecución captador lo utiliza solamente la aplicación creadora del paquete. No se utiliza para ejecutar otras aplicaciones de AIR. Crear un motor de ejecución es una opción en Mac y Windows. En iOS, todas las aplicaciones incluyen un paquete de motor de ejecución. A partir de AIR 3.7, todas las aplicaciones de Android incluyen un paquete de motor de ejecución por defecto (aunque tiene la opción de usar un motor por separado).
- Mediante la instalación de un entorno de desarrollo de AIR como el kit de desarrollo de software de AIR, Adobe® Flash® Builder™ o el kit de desarrollo de software (SDK) de Adobe Flex® (que incluye las herramientas de desarrollo de la línea de comandos de AIR). El motor de ejecución incluido en el SDK solo se utiliza al depurar las aplicaciones; no se emplea para ejecutar aplicaciones de AIR instaladas.

Los requisitos del sistema para instalar AIR y ejecutar aplicaciones de AIR se describen en: [Adobe AIR: Requisitos del sistema](#) (<http://www.adobe.com/es/products/air/systemreqs/>).

Tanto el instalador del motor de ejecución como el instalador de la aplicación de AIR, crean archivos de registro cuando instalan, actualizan o eliminan aplicaciones de AIR o el propio motor de ejecución de AIR. Puede consultar estos registros para ayudar a determinar la causa de cualquier problema de instalación. Consulte [Installation logs](#) (Registros de instalación; en inglés).

Instalación de Adobe AIR

Para instalar o actualizar el motor de ejecución, el usuario debe tener privilegios administrativos para el equipo.

Instalación del motor de ejecución en un ordenador con Windows

- 1 Descargue el archivo de instalación del motor de ejecución en <http://get.adobe.com/es/air>.
- 2 Haga doble clic en el archivo de instalación del motor de ejecución.
- 3 Siga las indicaciones que aparecen en la ventana de instalación para llevar a cabo la instalación.

Instalación del motor de ejecución en un ordenador con Mac

- 1 Descargue el archivo de instalación del motor de ejecución en <http://get.adobe.com/es/air>.
- 2 Haga doble clic en el archivo de instalación del motor de ejecución.
- 3 Siga las indicaciones que aparecen en la ventana de instalación para llevar a cabo la instalación.

- 4 Si el instalador presenta una ventana de autenticación, escriba el nombre de usuario y la contraseña que utiliza para Mac OS.

Instalación del motor de ejecución en un ordenador con Linux

Nota: de momento, AIR 2.7 y posterior no se admite en Linux. Las aplicaciones de AIR desarrolladas para Linux deben seguir utilizando el SDK de AIR 2.6.

Uso del instalador binario:

- 1 Localice el archivo binario de instalación en http://kb2.adobe.com/es/cps/853/cpsid_85304.html y realice la descarga.
- 2 Establezca los permisos de archivo para que se pueda ejecutar la aplicación de instalación: Desde una línea de comandos, se pueden establecer los permisos de archivo con:

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

Algunas versiones de Linux permiten establecer permisos de archivo en el cuadro de diálogo de propiedades que se abre mediante un menú contextual.

- 3 Ejecute el instalador desde la línea de comandos o haciendo doble clic en el archivo de instalación.
- 4 Siga las indicaciones que aparecen en la ventana de instalación para llevar a cabo la instalación.

Adobe AIR está instalado como paquete nativo. Es decir, como rpm en una distribución basada en rpm y deb en una distribución Debian. Actualmente AIR no admite ningún otro formato de paquete.

Uso de los instaladores de paquete:

- 1 Localice el paquete de AIR en http://kb2.adobe.com/es/cps/853/cpsid_85304.html. Descargue el paquete rpm o Debian, dependiendo de qué formato de paquete admita el sistema.
- 2 Si es necesario, haga doble clic en el archivo del paquete de AIR para instalar el paquete.

También puede realizar la instalación desde la línea de comandos:

- a En un sistema Debian:

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

- b En un sistema basado en rpm:

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

O bien, si está actualizando una versión existente (AIR 1.5.3 o posterior):

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

La instalación de aplicaciones de AIR 2 y AIR requiere que disponga de privilegios de administrador en su equipo.

Adobe AIR se encuentra instalada en la siguiente ubicación: /opt/Adobe AIR/Versions/1.0

AIR registra el tipo mime "application/vnd.adobe.air-application-installer-package+zip", lo que significa que los archivos .air son de este tipo mime y, por lo tanto, se registran con el motor de ejecución de AIR.

Instalación del motor de ejecución en un dispositivo de Android

Se puede instalar la versión más reciente del motor de ejecución de AIR desde Android Market.

Es posible instalar las versiones de desarrollo del motor de ejecución de AIR desde un vínculo en una página web utilizando el comando `-installRuntime` de ADT. Solo una versión del motor de ejecución de AIR se puede instalar a la vez; no se pueden tener las versiones de desarrollo y lanzamiento instaladas al mismo tiempo.

Consulte "Comando `installRuntime` de ADT" en la página 186 para obtener más información.

Instalación del motor de ejecución en un dispositivo de iOS

El código del motor de AIR necesario se integra con cada aplicación creada para los dispositivos iPhone, iPod touch y iPad. No instale un componente del motor de ejecución independiente.

Más temas de ayuda

“[AIR para iOS](#)” en la página 75

Desinstalación de Adobe AIR

Una vez instalado el motor de ejecución, se puede desinstalar siguiendo los procedimientos que se explican a continuación.

Desinstalación del motor de ejecución en un ordenador con Windows

- 1 En el menú Inicio de Windows, seleccione Configuración > Panel de control.
- 2 Abra los programas, programas y funciones o el panel de control para agregar o quitar programas (dependiendo de la versión de Windows que ejecute).
- 3 Seleccione “Adobe AIR” para desinstalar el motor de ejecución.
- 4 Haga clic en el botón Cambiar o quitar.

Desinstalación del motor de ejecución en un ordenador con Mac

- Haga doble clic en el archivo de desinstalación de Adobe AIR, que se encuentra en la carpeta /Aplicaciones/Utilidades.

Desinstalación del motor de ejecución en un ordenador con Linux

Realice uno de los siguientes pasos:

- Seleccione el comando “Desinstalador de Adobe AIR” en el menú Aplicaciones.
- Ejecute el instalador de AIR con la opción `-uninstall`.
- Elimine los paquetes de AIR (`adobeair` y `adobecerts`) con el administrador de paquetes.

Elimine el motor de ejecución del dispositivo de Android

- 1 Abra la aplicación de configuración en el dispositivo.
- 2 Puntee en la entrada de Adobe AIR en Aplicaciones > Administrar aplicaciones.
- 3 Puntee el botón Desinstalar.

También puede utilizar el comando `-uninstallRuntime` de ADT. Consulte el “[Comando uninstallRuntime de ADT](#)” en la página 187 para obtener más información.

Eliminación de un paquete de motor de ejecución

Para eliminar un motor de ejecución captador, debe eliminar la aplicación con la que se haya instalado. Recuerde que los motores de ejecución captadores solo se utilizan para ejecutar la aplicación de instalación.

Instalación y ejecución de las aplicaciones de AIR de muestra

Para instalar o actualizar una aplicación de AIR, un usuario debe disponer de privilegios administrativos para el equipo.

Hay algunas aplicaciones de muestra a disposición para demostrar las funciones de AIR. Para tener acceso a las mismas e instalarlas, siga estas instrucciones:

- 1 Descargue y ejecute las [aplicaciones de AIR de muestra](#). Están a disposición tanto las aplicaciones compiladas como el código fuente.
- 2 Para descargar y ejecutar una aplicación de muestra, haga clic en el botón Instalar ahora de la aplicación. Un mensaje indica instalar y ejecutar la aplicación.
- 3 Si opta por descargar aplicaciones de muestra y ejecutarlas más adelante, seleccione los vínculos de descarga. Las aplicaciones de AIR pueden ejecutarse en cualquier momento de la siguiente manera:
 - En Windows, haga doble clic en el icono de la aplicación que se encuentra en el escritorio o seleccione la aplicación en el menú Inicio.
 - En Mac OS, haga doble clic en el icono de la aplicación, que se instala por omisión en la carpeta Aplicaciones de su directorio de usuario (por ejemplo, en Macintosh HD/Usuarios/UsuarioFicticio/Aplicaciones/).

Nota: revise las notas de versión de AIR por si hubiera alguna actualización de estas instrucciones. Puede encontrarlas en: http://www.adobe.com/go/learn_air_relnotes_es.

Actualizaciones de Adobe AIR

De forma periódica, Adobe actualiza Adobe AIR con nuevas funciones o soluciones para problemas menores. La función de actualización y notificación automática permite que Adobe avise automáticamente a los usuarios del momento en que está disponible una versión actualizada de Adobe AIR.

Las actualizaciones a Adobe AIR garantizan que la aplicación funcione adecuadamente y a menudo contienen cambios importantes para la seguridad. Adobe recomienda que los usuarios actualicen a la versión más reciente de Adobe AIR si existe una nueva versión disponible, especialmente cuando se implica la actualización de seguridad.

De forma predeterminada, cuando se inicia una aplicación de AIR, el motor de ejecución comprueba si una actualización está disponible. Esta comprobación se lleva a cabo si han pasado más de dos semanas desde la última búsqueda de actualizaciones. Si alguna actualización está disponible, AIR la descarga en segundo plano.

Los usuarios pueden desactivar la capacidad de actualización automática, utilizando la aplicación SettingsManager de AIR. La aplicación SettingsManager de AIR está disponible para descarga en <http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air>.

El proceso de instalación normal de Adobe AIR incluye la conexión a <http://airinstall.adobe.com> para enviar la información básica sobre el entorno de instalación como, por ejemplo, idioma y versión del sistema operativo. Esta información solo se transmite una vez por instalación y permite que Adobe pueda confirmar que la instalación se ha realizado correctamente. No se transmitirá ni se recopilará información de identificación personal.

Actualización de motores de ejecución captadores

Si distribuye la aplicación con un paquete de motor de ejecución captador, este no se actualizará automáticamente. Por seguridad de sus propios usuarios, debe supervisar las actualizaciones publicadas por Adobe y actualizar su aplicación con la nueva versión del motor de ejecución cuando se publiquen cambios de seguridad importantes.

Capítulo 3: Trabajo con las API de AIR

Adobe® AIR® incluye funcionalidad que no está disponible para el contenido SWF que se ejecuta en Adobe® Flash® Player.

Desarrolladores de ActionScript 3.0

Las API de Adobe AIR se encuentran documentadas en las siguientes referencias:

- [Guía del desarrollador de ActionScript 3.0](#)
- [Referencia de ActionScript 3.0 para la plataforma de Adobe Flash](#)

Desarrolladores de HTML

Si está creando aplicaciones de AIR basadas en HTML, las API que están disponibles en JavaScript mediante el archivo AIRAliases.js (consulte [Acceso a las clases de API de AIR desde JavaScript](#)) se documentan en las siguientes referencias:

- [HTML Developer's Guide for Adobe AIR \(en inglés\)](#)
- [Adobe AIR API Reference for HTML Developers](#)

Clases de ActionScript 3.0 específicas de AIR

Las siguiente tabla contiene clases del motor de ejecución son específicas de Adobe AIR. No están disponibles para el contenido SWF que se ejecuta en Adobe® Flash® Player en el navegador.

Desarrolladores de HTML

Las clases que están disponibles en JavaScript mediante el archivo AIRAliases.js se incluyen en [Adobe AIR API Reference for HTML Developers](#).

Clase	Paquete de ActionScript 3.0	Añadido en versión de AIR
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

Clase	Paquete de ActionScript 3.0	Añadido en versión de AIR
CameraUI	flash.media	2.5
CertificateStatus	flash.security	2.0
CompressionAlgorithm	flash.utils	1.0
DatagramSocket	flash.net	2.0
DatagramSocketDataEvent	flash.events	2.0
DNSResolver	flash.net.dns	2.0
DNSResolverEvent	flash.events	2.0
DockIcon	flash.desktop	1.0
DownloadErrorEvent	air.update.events	1.5
DRMAuthenticateEvent	flash.events	1.0
DRMDeviceGroup	flash.net.drm	3.0
DRMDeviceGroupErrorEvent	flash.net.drm	3.0
DRMDeviceGroupEvent	flash.net.drm	3.0
DRMManagerError	flash.errors	1.5
EncryptedLocalStore	flash.data	1.0
ExtensionContext	flash.external	2.5
File	flash.filesystem	1.0
FileListEvent	flash.events	1.0
FileMode	flash.filesystem	1.0
FileStream	flash.filesystem	1.0
FocusDirection	flash.display	1.0
GameInput	flash.ui	3.0
GameInputControl	flash.ui	3.0
GameInputControlType	flash.ui	3.6 y versiones anteriores; eliminados, desde 3.7
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 y versiones anteriores; eliminados, desde 3.7
GameInputHand	flash.ui	3.6 y versiones anteriores; eliminados, desde 3.7
Geolocation	flash.sensors	2.0
GeolocationEvent	flash.events	2.0
HTMLHistoryItem	flash.html	1.0
HTMLHost	flash.html	1.0
HTMLLoader	flash.html	1.0

Clase	Paquete de ActionScript 3.0	Añadido en versión de AIR
HTMLPDFCapability	flash.html	1.0
HTMLSWFCapability	flash.html	2.0
HTMLUncaughtScriptExceptionEvent	flash.events	1.0
HTMLWindowCreateOptions	flash.html	1.0
Icon	flash.desktop	1.0
IFilePromise	flash.desktop	2.0
ImageDecodingPolicy	flash.system	2.6
Interactivelcon	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

Clase	Paquete de ActionScript 3.0	Añadido en versión de AIR
NativeWindowResize	flash.display	1.0
NativeWindowSystemChrome	flash.display	1.0
NativeWindowType	flash.display	1.0
NetworkInfo	flash.net	2.0
NetworkInterface	flash.net	2.0
NotificationType	flash.desktop	1.0
OutputProgressEvent	flash.events	1.0
PaperSize	flash.printing	2.0
PrintMethod	flash.printing	2.0
PrintUIOptions	flash.printing	2.0
PTRRecord	flash.net.dns	2.0
ReferencesValidationSetting	flash.security	1.0
ResourceRecord	flash.net.dns	2.0
RevocationCheckSettings	flash.security	1.0
Screen	flash.display	1.0
ScreenMouseEvent	flash.events	1.0
SecureSocket	flash.net	2.0
SecureSocketMonitor	air.net	2.0
ServerSocket	flash.net	2.0
ServerSocketConnectEvent	flash.events	2.0
ServiceMonitor	air.net	1.0
SignatureStatus	flash.security	1.0
SignerTrustSettings	flash.security	1.0
SocketMonitor	air.net	1.0
SoftKeyboardType	flash.text	3.0
SQLCollationType	flash.data	1.0
SQLColumnNameStyle	flash.data	1.0
SQLColumnSchema	flash.data	1.0
SQLConnection	flash.data	1.0
SQLException	flash.errors	1.0
SQLExceptionEvent	flash.events	1.0
SQLExceptionOperation	flash.errors	1.0
SQLEvent	flash.events	1.0
SQLIndexSchema	flash.data	1.0

Clase	Paquete de ActionScript 3.0	Añadido en versión de AIR
SQLMode	flash.data	1.0
SQLResult	flash.data	1.0
SQLSchema	flash.data	1.0
SQLSchemaResult	flash.data	1.0
SQLStatement	flash.data	1.0
SQLTableSchema	flash.data	1.0
SQLTransactionLockType	flash.data	1.0
SQLTriggerSchema	flash.data	1.0
SQLUpdateEvent	flash.events	1.0
SQLViewSchema	flash.data	1.0
SRVRecord	flash.net.dns	2.0
StageAspectRatio	flash.display	2.0
StageOrientation	flash.display	2.0
StageOrientationEvent	flash.events	2.0
StageText	flash.text	3.0
StageTextInitOptions	flash.text	3.0
StageWebView	flash.media	2.5
StatusFileUpdateErrorEvent	air.update.events	1.5
StatusFileUpdateEvent	air.update.events	1.5
StatusUpdateErrorEvent	air.update.events	1.5
StatusUpdateEvent	air.update.events	1.5
StorageVolume	flash.filesystem	2.0
StorageVolumeChangeEvent	flash.events	2.0
StorageVolumeInfo	flash.filesystem	2.0
SystemIdleMode	flash.desktop	2.0
SystemTrayIcon	flash.desktop	1.0
TouchEventIntent	flash.events	3.0
UpdateEvent	air.update.events	1.5
Updater	flash.desktop	1.0
URLFilePromise	air.desktop	2.0
URLMonitor	air.net	1.0
URLRequestDefaults	flash.net	1.0
XMLSignatureValidator	flash.security	1.0

Clases de Flash Player con funcionalidad específica de AIR

Las siguientes clases están disponibles para el contenido SWF que se ejecuta en el navegador, pero AIR ofrece propiedades o métodos adicionales:

Paquete	Clase	Propiedad, método o evento	Añadido en versión de AIR
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
		Evento orientationChange	2.0
		Evento orientationChanging	2.0
		setAspectRatio	2.0
		setOrientation	2.0
		softKeyboardRect	2.6
		supportedOrientations	2.6
		supportsOrientationChange	2.0
		NativeWindow	owner
	listOwnedWindows		2.6
	NativeWindowInitOptions	owner	2.6

Paquete	Clase	Propiedad, método o evento	Añadido en versión de AIR
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
	responseHeaders		1.0
	responseURL		1.0
	KeyboardEvent	commandKey	1.0
		controlKey	1.0

Paquete	Clase	Propiedad, método o evento	Añadido en versión de AIR
flash.net	FileReference	extension	1.0
		httpResponseStatus event	1.0
		uploadUnencoded()	1.0
	NetStream	Evento drmAuthenticate	1.0
		Evento onDRMContentData	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	Evento httpResponseStatus	1.0

Paquete	Clase	Propiedad, método o evento	Añadido en versión de AIR
flash.printing	PrintJob	active	2.0
		copies	2.0
		firstPage	2.0
		isColor	2.0
		jobName	2.0
		lastPage	2.0
		maxPixelsPerInch	2.0
		paperArea	2.0
		printableArea	2.0
		printer	2.0
		printers	2.0
		selectPaperSize ()	2.0
		showPageSetupDialog ()	2.0
		start2 ()	2.0
		supportsPageSetupDialog	2.0
		terminate ()	2.0
	PrintJobOptions	pixelsPerInch	2.0
printMethod		2.0	
flash.system	Capabilities	languages	1.1
	LoaderContext	allowLoadBytesCodeExecution	1.0
	Security	APPLICATION	1.0
flash.ui	KeyLocation	D_PAD	2.5

La mayoría de estos nuevos métodos y propiedades solo están disponibles para contenido que se encuentra en el entorno limitado de seguridad de la aplicación de AIR. No obstante, los nuevos integrantes de las clases URLRequest también están disponibles para el contenido que se ejecuta en otros entornos limitados.

Los métodos `ByteArray.compress ()` y `ByteArray.uncompress ()` incluyen cada uno un nuevo parámetro, `algorithm`, que permite seleccionar entre la compresión deflate y zlib. Este parámetro está disponible únicamente en el contenido que se ejecuta en AIR.

Componentes de Flex específicos de AIR

Los siguientes componentes MX de Adobe® Flex™ están a disposición cuando se desarrolla contenido para Adobe AIR:

- [FileEvent](#)
- [FileSystemComboBox](#)
- [FileSystemDataGrid](#)

- [FileSystemEnumerationMode](#)
- [FileSystemHistoryButton](#)
- [FileSystemList](#)
- [FileSystemSizeDisplayMode](#)
- [FileSystemTree](#)
- [FlexNativeMenu](#)
- [HTML](#)
- [Window](#)
- [WindowedApplication](#)
- [WindowedSystemManager](#)

Asimismo, Flex 4 incluye los siguientes componentes spark de AIR:

- [Window](#)
- [WindowedApplication](#)

Para obtener más información sobre los componentes de Flex para AIR, consulte [Using the Flex AIR components](#) (Uso de componentes de Flex para AIR).

Capítulo 4: Herramientas de la plataforma de Adobe Flash para el desarrollo de AIR

Es posible desarrollar aplicaciones de AIR con las siguientes herramientas de desarrollo de la plataforma de Adobe Flash:

Para los desarrolladores de ActionScript 3.0 (Flash y Flex):

- Adobe Flash Professional (consulte [Publicación para AIR](#))
- SDK de Adobe Flex 3.x y 4.x (consulte “[Configuración del SDK de Flex](#)” en la página 20 y “[AIR Developer Tool \(ADT\)](#)” en la página 174)
- Adobe Flash Builder (consulte [Desarrollo de aplicaciones de AIR con Flash Builder](#))

Para desarrolladores de HTML y Ajax:

- SDK de Adobe AIR (consulte “[Instalación del SDK de AIR](#)” en la página 18 y “[AIR Developer Tool \(ADT\)](#)” en la página 174)
- Adobe Dreamweaver CS3, CS4, CS5 (consulte [Extensión de AIR para Dreamweaver](#))

Instalación del SDK de AIR

El SDK de Adobe AIR contiene las siguientes herramientas de la línea de comandos que se utilizan para iniciar y empaquetar aplicaciones:

AIR Debug Launcher (ADL) Permite ejecutar aplicaciones de AIR sin tener que instalarlas primero. Consulte “[AIR Debug Launcher \(ADL\)](#)” en la página 168.

AIR Development Tool (ADT) Empaqueta aplicaciones de AIR en paquetes de instalación distribuibles. Consulte “[AIR Developer Tool \(ADT\)](#)” en la página 174.

Las herramientas de la línea de comandos de AIR requieren Java para su instalación en el equipo. La máquina virtual Java se puede utilizar desde JRE o JDK (versión 1.5 o posterior). Java JRE y Java JDK se encuentran disponibles en <http://java.sun.com/>.

Se requieren 2 GB de memoria como mínimo para ejecutar la herramienta ADT.

***Nota:** Java no se requiere para los usuarios finales que ejecuten aplicaciones de AIR.*

Para obtener una breve descripción general sobre la creación de una aplicación de AIR con el SDK de AIR, consulte “[Creación de la primera aplicación de AIR basada en HTML con el SDK de AIR](#)” en la página 36.

Descarga e instalación del SDK de AIR

El SDK de AIR se puede descargar e instalar utilizando las siguientes instrucciones:

Instalación del SDK de AIR en Windows

- Descargue el archivo de instalación del SDK de AIR.

- El SDK de AIR se distribuye como archivo estándar. Para instalar AIR, extraiga el contenido del SDK en una carpeta del equipo (por ejemplo: C:\Archivos de programa\Adobe\AIRSDK o C:\AIRSDK).
- Las herramientas ADL y ADT se incluyen en la carpeta bin del SDK de AIR; añada la ruta a esta carpeta a la variable de entorno PATH.

Instalación del SDK de AIR en Mac OS X

- Descargue el archivo de instalación del SDK de AIR.
- El SDK de AIR se distribuye como archivo estándar. Para instalar AIR, extraiga el contenido del SDK en una carpeta del equipo (por ejemplo: /Users/<userName>/Applications/AIRSDK).
- Las herramientas ADL y ADT se incluyen en la carpeta bin del SDK de AIR; añada la ruta a esta carpeta a la variable de entorno PATH.

Instalación del SDK de AIR en Linux

- El SDK está disponible en el formato tbz2.
- Para instalar el SDK, cree una carpeta en la que desee descomprimirlo y, a continuación, utilice el siguiente comando: `tar -jxvf <path to AIR-SDK.tbz2>`

Para obtener información sobre instrucciones para comenzar a utilizar las herramientas del SDK de AIR, consulte Creación de aplicaciones de AIR con las herramientas de la línea de comandos.

Componentes del SDK de AIR

En la siguiente tabla se describen las funciones de los archivos incluidos en el SDK de AIR:

Carpeta del SDK	Descripción de las herramientas/archivos
bin	AIR Debug Launcher (ADL) permite ejecutar una aplicación de AIR sin empaquetarla e instalarla primero. Para obtener más información sobre el uso de esta herramienta, consulte " AIR Debug Launcher (ADL) " en la página 168. AIR Developer Tool (ADT) empaqueta la aplicación como archivo de AIR para distribución. Para obtener información sobre el uso de esta herramienta, consulte " AIR Developer Tool (ADT) " en la página 174.
frameworks	El directorio libs contiene bibliotecas de código para su uso en aplicaciones de AIR. El directorio projects contiene el código para las bibliotecas SWF y SWC compiladas.
include	El directorio include contiene el archivo de encabezado del lenguaje C para escribir extensiones nativas.
install	El directorio install contiene los controladores USB de Windows para dispositivos de Android. (Estos son los controladores proporcionados por Google en el SDK de Android.)
lib	Contiene el código de compatibilidad para las herramientas del SDK de AIR.

Carpeta del SDK	Descripción de las herramientas/archivos
runtimes	Los motores de ejecución de AIR para los dispositivos móviles y de escritorio. ADL utiliza el motor de ejecución para iniciar las aplicaciones de AIR antes de que se empaqueten o se instalen. Los motores de ejecución de AIR para Android (paquetes APK) se pueden instalar en emuladores o dispositivos de Android para desarrollo y prueba. Los paquetes APK independientes se utilizan para los dispositivos y emuladores. (El motor de ejecución público de AIR para Android está disponible en Android Market.)
samples	Esta carpeta contiene un archivo descriptor de la aplicación de ejemplo, un ejemplo de la función de instalación integrada (badge.swf) y los iconos de la aplicación de AIR predeterminados.
templates	descriptor-template.xml: una plantilla del archivo descriptor de la aplicación que es necesaria para todas las aplicaciones de AIR. Para ver una descripción detallada del archivo descriptor de la aplicación, consulte “Archivos descriptores de las aplicaciones de AIR” en la página 214. Los archivos de esquema de la estructura XML del descriptor de la aplicación para cada versión oficial de AIR también se encuentran en esta carpeta.

Configuración del SDK de Flex

Para desarrollar aplicaciones de Adobe® AIR® con Adobe® Flex™, dispone de las siguientes opciones:

- Puede descargar e instalar Adobe® Flash® Builder™, que proporciona herramientas integradas para crear proyectos de Adobe AIR y comprobar, depurar y empaquetar aplicaciones de AIR. Consulte “[Creación de su primera aplicación de AIR de Flex de escritorio en Flash Builder](#)” en la página 22.
- Puede descargar el SDK de Adobe® Flex™ y desarrollar aplicaciones de AIR de Flex utilizando su editor de texto favorito y las herramientas de la línea de comandos.

Para obtener una breve descripción general de la creación de una aplicación de AIR con el SDK de Flex, consulte “[Creación de la primera aplicación de AIR de escritorio con el SDK de Flex](#)” en la página 40.

Instalación del SDK de Flex

La creación de aplicaciones de AIR con las herramientas de la línea de comandos requiere que Java esté instalado en el equipo. La máquina virtual Java se puede utilizar desde JRE o JDK (versión 1.5 o posterior). El JRE y JDK de Java están disponibles en <http://java.sun.com/>.

Nota: Java no se requiere para los usuarios finales que ejecuten aplicaciones de AIR.

El SDK de Flex proporciona la API de AIR y las herramientas de la línea de comandos que se utilizan para empaquetar, compilar y depurar las aplicaciones de AIR.

- 1 Si aún no lo ha hecho, descargue el SDK de Flex en <http://opensource.adobe.com/wiki/display/flexsdk/Downloads>.
- 2 Ubique el contenido del SDK en una carpeta (por ejemplo, Flex SDK).
- 3 Copie el contenido del SDK de AIR en los archivos del SDK de Flex.

Nota: en los equipos Mac, asegúrese de que copia o reemplaza los archivos independientes en las carpetas del SDK, no los directorios completos. De forma predeterminada, con la copia de un directorio en Mac en un directorio del mismo nombre se eliminan los archivos existentes en el directorio de destino; no se combina el contenido de los dos directorios. Puede utilizar el comando `ditto` en la ventana de terminal para combinar el SDK de AIR con el SDK de Flex: `ditto air_sdk_folder flex_sdk_folder`

- 4 Las utilidades de AIR de la línea de comandos se encuentran en la carpeta bin.

Configuración de los SDK externos

El desarrollo de aplicaciones para Android y iOS requiere que descargue los archivos de suministro, los SDK u otras herramientas de desarrollo de los creadores de plataformas.

Para obtener más información sobre la descarga e instalación del SDK de Android, consulte [Android Developers: Installing the SDK](#). Desde AIR 2.6, no es necesario descargar el SDK de Android. El SDK de AIR ahora incluye los componentes básicos necesarios para instalar e iniciar paquetes de APK. Aun así, el SDK de Android puede resultar útil para una serie de tareas de desarrollo, entre las que se incluyen la creación o ejecución de emuladores de software y capturas de pantalla del dispositivo.

Un SDK externo no es necesario para el desarrollo de iOS. Sin embargo, se requieren certificados especiales y perfiles de suministro. Para obtener más información, consulte [Obtención de archivos de desarrollo de Apple](#).

Capítulo 5: Creación de su primera aplicación de AIR

Creación de su primera aplicación de AIR de Flex de escritorio en Flash Builder

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear y empaquetar una sencilla aplicación “Hello World” de AIR basada en archivos SWF utilizando Adobe® Flash® Builder.

Si aún no lo ha hecho, descargue e instale Flash Builder. Asimismo, descargue e instale la versión más reciente de Adobe AIR, que se encuentra en: www.adobe.com/go/air_es.

Creación de un proyecto de AIR

Flash Builder incluye las herramientas necesarias para desarrollar y empaquetar aplicaciones de AIR.

La creación de aplicaciones de AIR en Flash Builder o Flex Builder comienza del mismo modo en que se crean otros proyectos de aplicaciones basadas en Flex: mediante la definición de un nuevo proyecto.

- 1 Abra Flash Builder.
- 2 Seleccione File (Archivo) > New (Nuevo) > Flex Project (Proyecto de Flex).
- 3 Indique el nombre del proyecto como AIRHelloWorld.
- 4 En Flex, las aplicaciones de AIR se consideran un tipo de aplicación. Se dispone de dos opciones:
 - Una aplicación web que se ejecuta en Adobe® Flash® Player.
 - Una aplicación de escritorio que se ejecuta en Adobe AIR.

Seleccione Escritorio como tipo de aplicación.

- 5 Haga clic en Finish (Finalizar) para crear el proyecto.

Los proyectos de AIR constan inicialmente de dos archivos: el archivo principal MXML y el archivo XML de la aplicación (al que se hace referencia como archivo descriptor de la aplicación). El último archivo especifica las propiedades de la aplicación.

Para obtener más información, consulte [Developing AIR applications with Flash Builder](#) (Desarrollo de aplicaciones de AIR con Flash Builder; en inglés).

Escritura del código de aplicaciones de AIR

Para escribir el código de la aplicación “Hello World”, se edita el archivo MXML de la aplicación (AIRHelloWorld.mxml), que se abre en el editor. (Si el archivo no se abre, utilice el navegador de proyectos para abrir el archivo.)

Las aplicaciones de AIR de Flex en el escritorio se incluyen en la etiqueta MXML WindowedApplication. La etiqueta MXML WindowedApplication crea una sencilla ventana que incluye controles básicos como, por ejemplo, una barra de título y un botón de cierre.

- 1 Añada un atributo `title` al componente `WindowedApplication` y asígnele el valor "Hello World":

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

- 2 Añada un componente `Label` a la aplicación (sitúelo dentro de la etiqueta `WindowedApplication`), establezca la propiedad `text` del componente `Label` en "Hello AIR" y defina restricciones de diseño para mantenerlo centrado, tal y como se muestra a continuación:

```
<?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 Añada el siguiente bloque de estilo inmediatamente después de la etiqueta inicial `WindowedApplication` y antes de la etiqueta del componente `label` introduzca:

```
<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>
```

Esta configuración de estilo se aplica a toda la aplicación y se procesa el fondo de la ventana con un gris ligeramente transparente.

El código de la aplicación presenta en este momento el siguiente aspecto:

```
<?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>
```

A continuación, se modificará parte de la configuración del descriptor de la aplicación para permitir que la aplicación sea transparente:

- 1 En el panel Flex Navigator (Navegador de Flex), sitúe el archivo descriptor de la aplicación en el directorio de origen del proyecto. Si se ha asignado el nombre AIRHelloWorld al proyecto, este archivo se denomina AIRHelloWorld-app.xml.
- 2 Haga doble clic en el archivo descriptor de la aplicación para editarlo en Flash Builder.
- 3 En el código XML, sitúe las líneas de comentarios para las propiedades `systemChrome` y `transparent` (de la propiedad `initialWindow`). Elimine los comentarios. (Elimine los delimitadores de comentarios "`<!--`" y "`-->`".)
- 4 Establezca el valor de texto de la propiedad `systemChrome` en `none`, tal y como se muestra a continuación:
`<systemChrome>none</systemChrome>`
- 5 Establezca el valor de texto de la propiedad `transparent` en `true`, tal y como se indica a continuación:
`<transparent>>true</transparent>`
- 6 Guarde el archivo.

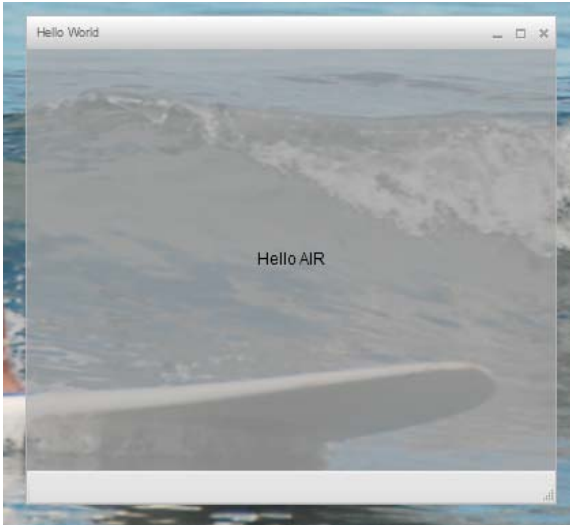
Prueba de la aplicación de AIR

Para probar el código de la aplicación que se ha escrito, ejecútelo en modo de depuración.

- 1 Haga clic en el botón Debug (Depurar)  de la barra de herramientas principal.

También puede seleccionar el comando Run (Ejecutar) > Debug (Depurar) > AIRHelloWorld.

La aplicación de AIR resultante debe ser similar al siguiente ejemplo:



- 2 Con el uso de las propiedades `horizontalCenter` y `verticalCenter` del control Label, el texto se sitúa en el centro de la ventana. Mueva o cambie el tamaño de la ventana tal y como lo haría en cualquier otra aplicación de escritorio.

Nota: si la aplicación no se compila, corrija la sintaxis o los errores ortográficos que se hayan podido introducir accidentalmente en el código. Los errores y advertencias se muestran en la vista *Problems (Problemas)* de Flash Builder.

Empaquetado, firma y ejecución de una aplicación de AIR

Ahora ya se puede empaquetar la aplicación "Hello World" en un archivo de AIR para su distribución. Un archivo de AIR es un archivo de almacenamiento que contiene los archivos de la aplicación, que son todos los archivos incluidos en la carpeta bin del proyecto. En este sencillo ejemplo, estos archivos son los archivos SWF y XML de la aplicación. El paquete de AIR se distribuye a los usuarios, que posteriormente lo utilizan para instalar la aplicación. Un paso necesario en este proceso consiste en firmarlo digitalmente.

- 1 Compruebe que la aplicación no presenta errores de compilación y que se ejecuta correctamente.
- 2 Seleccione Project (Proyecto) > Export Release Version (Exportar versión oficial).
- 3 Compruebe que el proyecto AIRHelloWorld y la aplicación AIRHelloWorld.mxml se incluyan para el proyecto y la aplicación.
- 4 Seleccione Exportar como opción de paquete de AIR firmado. A continuación, haga clic en Siguiente.
- 5 Si ya dispone de un certificado digital existente, haga clic en Examinar y selecciónelo.
- 6 Si debe crear un nuevo certificado digital con firma automática, seleccione Crear.
- 7 Introduzca la información necesaria y haga clic en Aceptar.
- 8 Haga clic en Finalizar para generar el paquete de AIR denominado AIRHelloWorld.air.

Ahora puede instalar y ejecutar la aplicación desde Project Navigator (Navegador de proyectos) en Flash Builder o desde el sistema de archivos haciendo doble clic en el archivo de AIR.

Creación de la primera aplicación de AIR de escritorio con Flash Professional

A continuación se resume la demostración del funcionamiento de Adobe® AIR®. Siga las instrucciones de este tema para crear y empaquetar una sencilla aplicación “Hello World” de AIR con Adobe® Flash® Professional.

Si aún no lo ha hecho, descargue e instale Adobe AIR, que se encuentra aquí: http://www.adobe.com/go/air_es.

Creación de la aplicación Hello World en Flash

Crear una aplicación de Adobe AIR en Flash es muy similar a crear cualquier otro archivo FLA. El siguiente procedimiento le guiará en el proceso de creación de una sencilla aplicación Hello World con Flash Professional.

Para crear la aplicación Hello World

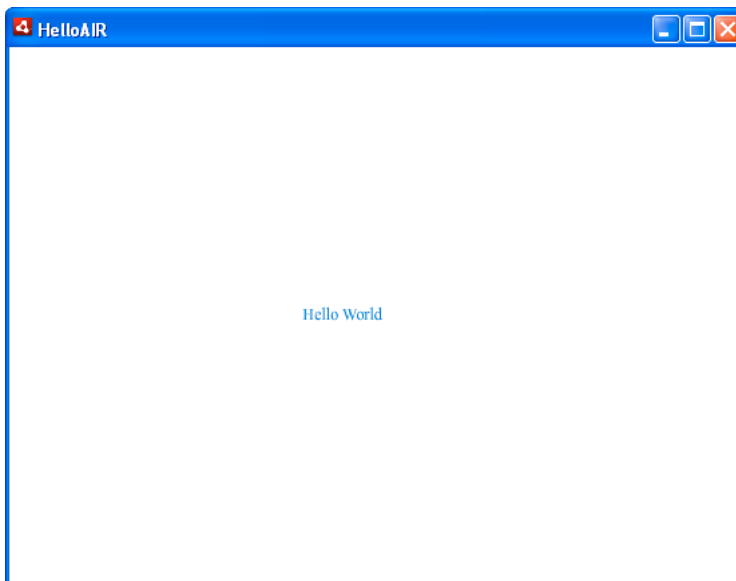
- 1 Inicie Flash.
- 2 En la pantalla de bienvenida, haga clic en AIR para crear un archivo FLA vacío con configuración de publicación de Adobe AIR.
- 3 Seleccione la herramienta Texto en el panel Herramientas y cree un campo de texto estático (valor predeterminado) en el centro del escenario. Dele una anchura suficiente para que pueda contener entre 15 y 20 caracteres.
- 4 Escriba el texto “Hello World” en el campo de texto.
- 5 Guarde el archivo y asígnele un nombre (por ejemplo, HelloAIR).

Prueba de la aplicación

- 1 Presione Ctrl + Intro o seleccione Control -> Probar película -> Probar para probar la aplicación en Adobe AIR.
- 2 Para utilizar la función Depurar película, añada primero código ActionScript a la aplicación. Puede intentarlo rápidamente añadiendo una sentencia trace como esta:

```
trace("Running AIR application using Debug Movie");
```
- 3 Presione Ctrl + Mayús + Intro o seleccione Depurar -> Depurar película para ejecutar la aplicación con Depurar película.

La aplicación Hello World se asemeja a la de la ilustración:



Empaquetado de la aplicación

- 1 Seleccione Archivo > Publicar.
- 2 Firme el paquete de Adobe AIR con un certificado digital existente o cree un certificado con firma automática utilizando los siguientes pasos:
 - a Haga clic en el botón Nuevo situado junto al campo Certificado.
 - b Rellene los campos Nombre del editor, Unidad de organización, Nombre de organización, Correo electrónico, País, Contraseña y Confirmar contraseña.
 - c Especifique el tipo de certificado. La opción Tipo de certificado hace referencia al nivel de seguridad: 1024-RSA utiliza una clave de 1.024 bits (menos segura) y 2048-RSA utiliza una clave de 2048 bits (más segura).
 - d Guarde la información en un archivo de certificado en la opción Guardar como o haciendo clic en el botón Examinar... para acceder a la ubicación de la carpeta. (Por ejemplo, *C:/Temp/mycert.pfx*). Cuando haya terminado, haga clic en Aceptar.
 - e Flash regresa al cuadro de diálogo Firma digital. La ruta y el nombre de archivo del certificado con firma automática creado aparece ahora en el cuadro de texto Certificado. Si no es así, introduzca la ruta y el nombre de archivo o haga clic en el botón Examinar para encontrarlo y seleccionarlo.
 - f Indique la misma contraseña en el campo de texto Contraseña del cuadro de diálogo Firma digital que la que se asignó en el paso b. Para obtener más información sobre la firma de las aplicaciones de Adobe AIR, consulte [“Firma digital de archivos de AIR”](#) en la página 199.
- 3 Para crear el archivo aplicación y el instalador, haga clic en el botón Publicar. (En Flash CS4 y CS5, haga clic en el botón Aceptar.) Debe ejecutar los comandos Probar película o Depurar película para crear los archivos SWF y application.xml antes de crear el archivo de AIR.
- 4 Para instalar la aplicación, haga doble clic en el archivo de AIR (*application.air*) en la misma carpeta en la que guardó la aplicación.
- 5 Haga clic en el botón Instalar del cuadro de diálogo Instalación de la aplicación.

- 6 Revise los parámetros de Preferencias de instalación y Ubicación y asegúrese de que la casilla de verificación 'Iniciar aplicación tras la instalación' está seleccionada. A continuación, haga clic en Continuar.
- 7 Haga clic en Finalizar cuando aparezca el mensaje Instalación completada.

Creación de su primera aplicación de AIR para Android en Flash Professional

Para desarrollar aplicaciones de AIR para Android, se debe descargar la extensión de Flash Professional CS5 para Android de [Adobe Labs](#).

También se debe descargar e instalar el SDK de Android en el sitio web de Android, tal y como se describe en: [Android Developers: Installing the SDK](#) (Desarrolladores de Android: Instalación del SDK; en inglés).

Creación de un proyecto

- 1 Abra Flash Professional CS5
- 2 Cree un nuevo proyecto de AIR para Android.
La pantalla de inicio de Flash Professional incluye un vínculo que crea una aplicación de AIR para Android. También puede seleccionar Archivo > Nuevo > y después seleccionar la plantilla de AIR para Android.
- 3 Guarde el documento como HelloWorld.fla.

Escritura del código

Debido a que el tutorial trata realmente sobre la escritura de código, utilice únicamente la herramienta Texto para escribir "Hello, World!" en el escenario.

Definición de las propiedades de la aplicación

- 1 Seleccione Archivo > Configuración de Android de AIR.
- 2 En la ficha General, establezca los siguientes ajustes:
 - Archivo de salida: HelloWorld.apk
 - Nombre de la aplicación: HelloWorld
 - ID de la aplicación: HelloWorld
 - Número de versión: 0.0.1
 - Proporción de aspecto: vertical
- 3 En la ficha Implementación, establezca los siguientes ajustes:
 - Certificado: hace referencia a un certificado de firma de código de AIR válido. Puede hacer clic en el botón Crear para crear un nuevo certificado. (Las aplicaciones de Android implementadas a través de Android Marketplace deben contar con certificados que sean válidos hasta el 2033 como mínimo.) Indique la contraseña del certificado en el campo Contraseña.
 - Tipo de implementación de Android: Depuración
 - Tras la publicación: Selección de ambas opciones
 - Introduzca la ruta a la herramienta ADB en el subdirectorio de herramientas del SDK de Android.
- 4 Cierre del cuadro de diálogo de ajustes de Android haciendo clic en Aceptar.

La aplicación no necesita iconos ni permisos en esta etapa de su desarrollo. La mayoría de las aplicaciones de AIR para Android requieren algunos permisos para poder acceder a funciones protegidas. Solo se deben establecer aquellos permisos que la aplicación requiera realmente debido a que los usuarios pueden rechazar la aplicación si esta solicita demasiados permisos.

- 5 Guarde el archivo.

Empaquetado e instalación de la aplicación en el dispositivo de Android

- 1 Asegúrese de que la depuración de USB esté activada en el dispositivo. La depuración USB se puede activar en Aplicaciones > Desarrollo.
- 2 Conecte el dispositivo al equipo mediante un cable USB.
- 3 Instale el motor de ejecución de AIR, si aún no lo ha hecho, dirigiéndose a Android Market y descargando Adobe AIR. (También puede instalar AIR localmente utilizando el comando “Comando [installRuntime de ADT](#)” en la página 186 de ADT. Los paquetes de Android para su uso en dispositivos de Android y los emuladores se incluyen en el SDK de AIR.)
- 4 Seleccione Archivo > Publicar.

Flash Professional crea el archivo APK, instala la aplicación en el dispositivo de Android conectado y lo inicia.

Creación de su primera aplicación de AIR para iOS

AIR 2.6 o posterior, iOS 4.2 o posterior

Puede codificar, crear y probar las funciones básicas de una aplicación de iOS utilizando solo herramientas de Adobe. Sin embargo, para instalar una aplicación de iOS en un dispositivo y distribuirla, debe unirse al programa Apple iOS Developer (servicio de pago). Una vez que se una al programa iOS Developer, puede acceder a iOS Provisioning Portal donde podrá obtener los siguientes elementos y archivos de Apple que son necesarios para instalar una aplicación en un dispositivo para prueba y para la distribución posterior. Entre estos elementos y archivos se incluyen:

- Certificados de distribución y desarrollo
- ID de la aplicación
- Archivos de suministro de distribución y desarrollo

Creación del contenido de la aplicación.

Cree un archivo SWF que muestre el texto “Hello world!” Esta tarea se puede realizar utilizando Flash Professional, Flash Builder u otro IDE. En este ejemplo simplemente se utiliza un editor de texto y el compilador SWF de la línea de comandos incluido en el SDK de Flex.

- 1 Cree un directorio en una ubicación adecuada para almacenar los archivos de la aplicación. Cree un archivo denominado *HelloWorld.as* y edítelo en su editor de código favorito.
- 2 Añada el siguiente código:

```
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 Compile la clase utilizando el compilador amxmlc:

```
amxmlc HelloWorld.as
```

Un archivo SWF, *HelloWorld.swf*, se crea en la misma carpeta.

Nota: en este ejemplo se da por sentado que ha configurado la variable de ruta de su entorno para que incluya el directorio que contiene amxmlc. Para obtener información sobre la configuración de la ruta, consulte [“Variables del entorno de ruta”](#) en la página 319. Como alternativa, es posible indicar la ruta completa en amxmlc y las otras herramientas de la línea de comandos utilizadas en este ejemplo.

Creación de gráficos de iconos y de la pantalla inicial de la aplicación

Todas las aplicaciones para iOS tienen iconos que aparecen en la interfaz de la aplicación iTunes y en la pantalla del dispositivo.

- 1 Cree un directorio en el directorio del proyecto y llámelo “icons”.
- 2 Cree tres archivos PNG en el directorio de iconos. Llámelos Icon_29.png, Icon_57.png e Icon_512.png.
- 3 Edite los archivos PNG para crear los gráficos que desee para la aplicación. Los archivos deben tener 29x29, 57x57 y 512x512 píxeles. Para esta prueba, basta con usar cuadrados de color sólido como gráficos.

Nota: cuando se envía una aplicación al App Store de Apple, se utiliza una versión JPG (no PNG) del archivo de 512 píxeles. Utilizará la versión PNG durante las versiones de desarrollo de pruebas de la aplicación.

Todas las aplicaciones para iPhone muestran una página inicial mientras se carga en el iPhone. Puede definir la imagen inicial en un archivo PNG:

- 1 En el directorio de desarrollo principal, cree un archivo PNG llamado Default.png. (No coloque este archivo en un subdirectorio de iconos.) Es importante que el nombre sea Default.png, con D mayúscula.)
- 2 Edite el archivo para que tenga 320 píxeles de ancho por 480 píxeles de alto. De momento, el contenido puede ser un rectángulo blanco sencillo. (Lo cambiaremos más adelante.)

Para obtener información detallada sobre estos gráficos, consulte [“Iconos de la aplicación”](#) en la página 93.

Creación del archivo descriptor de la aplicación

Cree un archivo descriptor de la aplicación que especifique las propiedades básicas de la aplicación. Esta tarea se puede completar utilizando un IDE como, por ejemplo, Flash Builder o un editor de texto.

- 1 En la carpeta del proyecto que contiene HelloWorld.as, cree un archivo XML denominado, *HelloWorld-app.xml*. Edite este archivo en el editor de XML favorito.
- 2 Añada el siguiente código XML:

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

Por motivos de simplicidad, en este ejemplo solo se establecen algunas de las propiedades disponibles.

Nota: si está utilizando AIR 2 o anterior, debe usar el elemento `<version>` en lugar de `<versionNumber>`.

- 3 Cambie el ID de la aplicación para que coincida con el ID especificado en iOS Provisioning Portal. (No incluya la parte raíz del paquete aleatoria al principio del ID).
- 4 Pruebe la aplicación utilizando ADL:

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

ADL debe abrir una ventana en el escritorio que muestre el texto: *Hello World!* Si no es así, compruebe el descriptor de la aplicación y el código fuente para localizar errores.

Compilación del archivo IPA

Ahora puede compilar el archivo de instalación IPA utilizando ADT. Debe contar con el certificado de desarrollador de Apple y la clave privada en formato de archivo P12 y el archivo de suministro de desarrollo de Apple.

Ejecute la utilidad ADT con las siguientes opciones, sustituyendo el almacén de claves, la contraseña para acceder al almacén y los valores del perfil de suministro con los suyos propios:

```
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
```

(Utilice una sola línea de comandos; los saltos de línea de este ejemplo solo se añaden para facilitar la lectura.)

ADT genera el archivo instalador de la aplicación de iOS, *HelloWorld.ipa*, en el directorio del proyecto. La compilación del archivo IPA puede tardar varios minutos.

Instalación de la aplicación en un dispositivo

Para instalar la aplicación de iOS para prueba:

- 1 Abra la aplicación iTunes.
- 2 Si aún no lo ha hecho, añada el archivo de suministro de esta aplicación a iTunes. En iTunes, seleccione Archivo > Añadir a la biblioteca. Posteriormente, seleccione el archivo de suministro (que tendrá *mobileprovision* como tipo de archivo provisional).

Por ahora, para probar la aplicación en su dispositivo de desarrollador, utilice el perfil de suministro de desarrollo.

Más adelante, cuando distribuya aplicaciones a iTunes Store, utilizará el perfil de distribución. Para distribuir la aplicación ad hoc (a varios dispositivos sin tener que pasar por iTunes Store), utilice el archivo de suministro ad hoc.

Para obtener más información sobre el suministro de perfiles, consulte [“Configuración de iOS”](#) en la página 71.

- 3 Algunas versiones de iTunes no sustituyen la aplicación si ya está instalada la misma versión. En ese caso, elimine la aplicación del dispositivo y de la lista de aplicaciones de iTunes.
- 4 Haga doble clic en el archivo IPA de su aplicación. Debe aparecer en la lista de aplicaciones en iTunes.
- 5 Conecte el dispositivo al puerto USB del equipo.
- 6 En iTunes, compruebe la ficha Aplicaciones del dispositivo y verifique que la aplicación aparece seleccionada en la lista de aplicaciones para instalar.
- 7 Seleccione el dispositivo en la lista de la izquierda de la aplicación iTunes. Haga clic en el botón Sincronizar. Cuando finalice la sincronización, la aplicación Hello World aparecerá en el iPhone.

Si no tiene instalada la versión más reciente, elimínela del dispositivo y de la lista de aplicaciones de iTunes y vuelva a realizar este procedimiento. Puede darse el caso si la versión instalada utiliza el mismo ID y versión que la aplicación existente.

Edición de los gráficos de la pantalla inicial

Recuerde que antes de compilar la aplicación, creó un archivo *Default.png* (consulte la sección [“Creación de gráficos de iconos y de la pantalla inicial de la aplicación”](#) en la página 30). Este archivo PNG sirve de imagen inicial cuando se carga la aplicación. Al probar la aplicación en el iPhone, probablemente se haya dado cuenta de que la pantalla inicial aparece en blanco.

Debe cambiar esta imagen para que coincida con la pantalla de inicio de la aplicación (“Hello World!”):

- 1 Abra la aplicación en el dispositivo. Cuando aparezca el primer texto “Hello World”, pulse el botón Inicio (debajo de la pantalla) y no lo suelte. Con el botón Inicio presionado, pulse el botón de encendido/reposo (en la parte superior del iPhone). De este modo hará una captura de pantalla y enviará la imagen al Carrete.
- 2 Transfiera la imagen al equipo de desarrollo desde iPhoto u otra aplicación de transferencia de fotografías. (En Mac OS, también puede utilizar la aplicación Instantánea.)

También puede enviar la fotografía al equipo de desarrollo por correo electrónico:

- Abra la aplicación Fotos.
- Abra el Carrete.
- Abra la captura de pantalla que realizó anteriormente.

- Toque la imagen y luego toque el botón “reenviar” (flecha) situado en la esquina inferior izquierda. Seguidamente, toque el botón Enviar foto y envíese la fotografía a usted mismo.
- 3 Reemplace el archivo Default.png (en su directorio de desarrollo) por una versión PNG de la imagen de la captura de pantalla.
 - 4 Vuelva a compilar la aplicación (consulte la sección “[Compilación del archivo IPA](#)” en la página 31) y reinstálela en el dispositivo.

La aplicación utilizará la nueva pantalla de inicio cuando se cargue.

Nota: puede crear cualquier gráfico que quiera para el archivo Default.png, siempre y cuando tenga las dimensiones correctas (320 x 480 píxeles). No obstante, suele ser mejor que la imagen del archivo Default.png coincida con el estado inicial de la aplicación.

Creación de la primera aplicación de AIR basada en HTML con Dreamweaver

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear y empaquetar una sencilla aplicación “Hello World” de AIR basada en HTML utilizando la extensión de Adobe® AIR® para Dreamweaver®.

Si aún no lo ha hecho, descargue e instale Adobe AIR, que se encuentra aquí: http://www.adobe.com/go/air_es.

Para obtener instrucciones sobre la instalación de la extensión de Adobe AIR para Dreamweaver, consulte [Instalación de la extensión de AIR para Dreamweaver](#).

Para obtener información general sobre la extensión, incluyendo los requisitos del sistema, consulte [Extensión de AIR para Dreamweaver](#).

Nota: las aplicaciones de AIR basadas en HTML solo se pueden desarrollar para los perfiles extendedDesktop y de escritorio. El perfil móvil no es compatible.

Preparación de los archivos de la aplicación

La aplicación de Adobe AIR debe contar con una página de inicio y todas sus páginas relacionadas definidas en un sitio de Dreamweaver.

- 1 Inicie Dreamweaver y asegúrese de disponer de un sitio definido.
- 2 Abra una nueva página HTML seleccionando Archivo > Nuevo, elija HTML en la columna Tipo de página, seleccione Ninguno en la columna Diseño y haga clic en Crear.
- 3 En la nueva página, escriba **Hello World!**
Este ejemplo es muy sencillo, pero, si lo desea, puede aplicar el estilo que desee al texto, añadir más contenido a la página, vincular otras páginas a esta página de inicio, etc.
- 4 Guarde la página (Archivo > Guardar) como hello_world.html. Compruebe que el archivo se guarda en un sitio de Dreamweaver.

(Para obtener más información sobre los sitios de Dreamweaver, consulte la ayuda de Dreamweaver.)

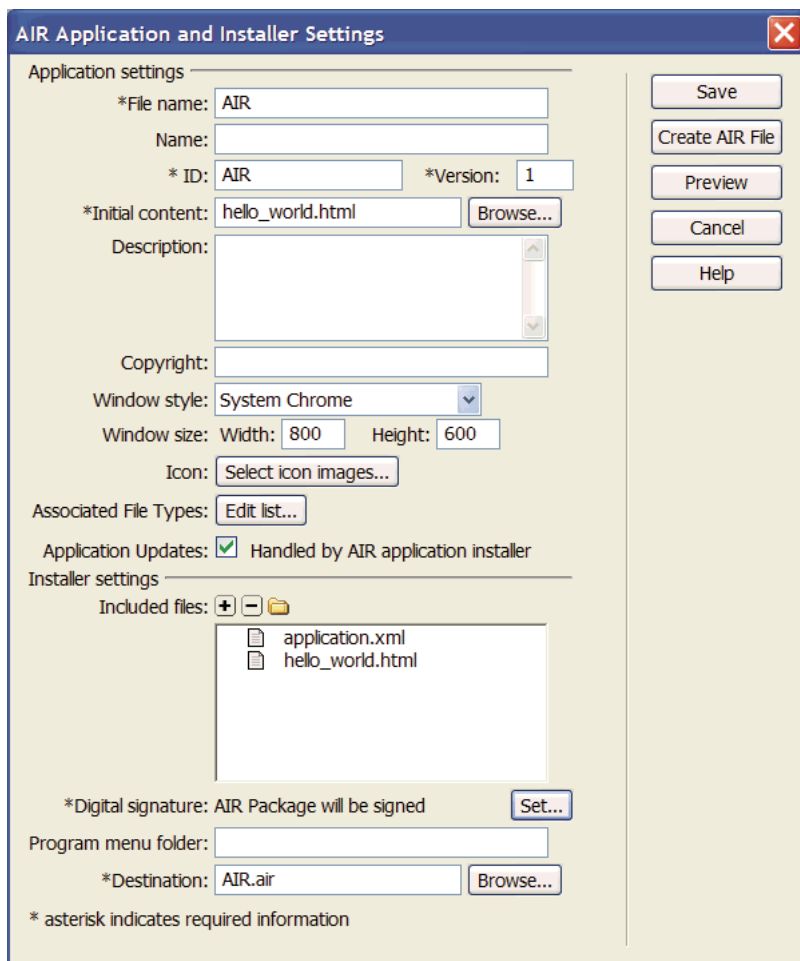
Creación de una aplicación de Adobe AIR

- 1 Compruebe que la página `hello_world.html` esté abierta en la ventana de documento de Dreamweaver. (Consulte la sección anterior para obtener instrucciones sobre cómo crearla.)
- 2 Seleccione Sitio > Configuración de la aplicación de Air.

La mayor parte de las opciones de configuración necesarias del cuadro de diálogo AIR - Configuración de aplicación e instalador se encuentran seleccionadas automáticamente. Sin embargo, se debe seleccionar el contenido inicial (o página de inicio) de la aplicación.
- 3 Haga clic en el botón Examinar situado junto a la opción Contenido inicial, desplácese a la página `hello_world.html` y selecciónela.
- 4 Junto a la opción Firma digital, haga clic en el botón Definir.

Con la firma digital se garantiza que el código de una aplicación no se ha alterado ni dañado desde su creación por parte del autor del software y es necesaria en todas las aplicaciones de Adobe AIR.
- 5 En el cuadro de diálogo Firma digital, seleccione Firmar el paquete de AIR con un certificado digital y haga clic en el botón Crear. (Si ya tiene acceso a un certificado digital, puede hacer clic en el botón Examinar para seleccionarlo.)
- 6 Complete los campos necesarios del cuadro de diálogo Certificado digital con firma automática. Debe indicar su nombre, una contraseña y su confirmación, así como un nombre para el archivo de certificado digital. Dreamweaver guarda el certificado digital en la raíz del sitio.
- 7 Haga clic en Aceptar para volver al cuadro de diálogo Firma digital.
- 8 En el cuadro de diálogo Firma digital, indique la contraseña especificada para el certificado digital y haga clic en Aceptar.

El cuadro de diálogo completo AIR - Configuración de aplicación e instalador puede presentar el siguiente aspecto:



Para obtener más información sobre todas las opciones del cuadro de diálogo y cómo modificarlas, consulte [Creación de una aplicación de AIR en Dreamweaver](#).

9 Haga clic en el botón Crear archivo de AIR.

Dreamweaver crea el archivo de la aplicación de Adobe AIR y lo guarda en la carpeta raíz del sitio. Dreamweaver también crea un archivo application.xml y lo guarda en el mismo lugar. Este archivo sirve como manifiesto y define distintas propiedades de la aplicación.

Instalación de la aplicación en el escritorio

Una vez creado el archivo de aplicación, puede instalarlo en cualquier escritorio.

1 Desplace el archivo de aplicación de Adobe AIR fuera del sitio de Dreamweaver y llévelo al escritorio deseado.

Este paso es opcional. Si lo prefiere, puede instalar la nueva aplicación en el equipo directamente desde el directorio del sitio de Dreamweaver.

2 Haga doble clic en el archivo ejecutable de la aplicación (archivo .air) para instalarla.

Vista previa de una aplicación de Adobe AIR

Se puede obtener una vista previa de las páginas que formarán parte de las aplicaciones de AIR en cualquier momento. Es decir, no necesariamente se debe empaquetar la aplicación antes de ver su aspecto cuando se instale.

- 1 Compruebe que la página `hello_world.html` esté abierta en la ventana de documento de Dreamweaver.
- 2 En la barra de herramientas Documento, haga clic en el botón de previsualización/depuración en el navegador y, a continuación, seleccione Vista previa en AIR.

También puede presionar `Ctrl+Mayús+F12` (Windows) o `Cmd+Mayús+F12` (Macintosh).

Al obtener una vista previa de esta página, fundamentalmente se está viendo lo que vería un usuario como página de inicio de la aplicación una vez instalada en un escritorio.

Creación de la primera aplicación de AIR basada en HTML con el SDK de AIR

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear y empaquetar una sencilla aplicación “Hello World” de AIR basada en HTML.

Para comenzar, debe tener instalado el motor de ejecución y configurar el SDK de AIR. En este tutorial utilizará *AIR Debug Launcher* (ADL) y *AIR Developer Tool* (ADT). ADL y ADT son programas de utilidades de línea de comandos y se pueden encontrar en el directorio `bin` del SDK de AIR (consulte “[Instalación del SDK de AIR](#)” en la página 18). En este tutorial se asume que está familiarizado con la ejecución de programas desde la línea de comandos y que conoce cómo configurar las variables del entorno de ruta necesarias para el sistema operativo.

Nota: Si es usuario de Adobe® Dreamweaver®, consulte “[Creación de la primera aplicación de AIR basada en HTML con Dreamweaver](#)” en la página 33.

Nota: las aplicaciones de AIR basadas en HTML solo se pueden desarrollar para los perfiles `extendedDesktop` y de escritorio. El perfil `móvil` no es compatible.

Creación de archivos del proyecto

Todos los proyectos de AIR basados en HTML deben incluir los siguientes archivos: un archivo descriptor de la aplicación, que especifica los metadatos de la aplicación y una página HTML de nivel superior. Además de estos archivos necesarios, este proyecto incluye un archivo de código JavaScript, `AIRAliases.js`, que define las variables de alias adecuadas para las clases de API de AIR.

- 1 Cree un directorio denominado `HelloWorld` para que incluya los archivos del proyecto.
- 2 Cree un archivo XML, denominado `HelloWorld-app.xml`.
- 3 Cree un archivo HTML denominado `HelloWorld.html`.
- 4 Copie `AIRAliases.js` de la carpeta `frameworks` del SDK de AIR al directorio `project`.

Creación del archivo descriptor de la aplicación de AIR

Para comenzar a crear la aplicación de AIR, cree un archivo descriptor de la aplicación XML con la siguiente estructura:

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

1 Abra HelloWorld-app.xml para la edición.

2 Añada el elemento raíz `<application>`, incluyendo el atributo de espacio de nombres de AIR:

`<application xmlns="http://ns.adobe.com/air/application/2.7">` El último segmento del espacio de nombres, "2.7", especifica la versión del motor de ejecución que requiere la aplicación.

3 Añada el elemento `<id>`:

`<id>examples.html.HelloWorld</id>` El ID de la aplicación la identifica de forma exclusiva junto con el ID de editor (que AIR obtiene del certificado utilizado para firmar el paquete de la aplicación). El ID de la aplicación se utiliza para la instalación, el acceso al directorio privado de almacenamiento del sistema de archivos de la aplicación, el acceso al almacenamiento cifrado privado y la comunicación entre aplicaciones.

4 Agregue el elemento `<versionNumber>`:

`<versionNumber>0.1</versionNumber>` Ayuda a los usuarios a determinar qué versión de la aplicación se está instalando.

Nota: si está utilizando AIR 2 o anterior, debe usar el elemento `<version>` en lugar de `<versionNumber>`.

5 Agregue el elemento `<filename>`:

`<filename>HelloWorld</filename>` Nombre utilizado para el ejecutable de la aplicación, el directorio de instalación y otras referencias a la aplicación en el sistema operativo.

6 Añada el elemento `<initialWindow>` que contiene los siguientes elementos secundarios para especificar las propiedades de la ventana de la aplicación inicial:

`<content>HelloWorld.html</content>` Identifica el archivo HTML raíz para que se cargue AIR.

`<visible>true</visible>` Hace visible a la ventana de forma inmediata.

`<width>400</width>` Establece la anchura de la ventana (en píxeles).

`<height>200</height>` Establece la altura de la ventana.

7 Guarde el archivo. El archivo descriptor de la aplicación completo debe presentar el siguiente aspecto:

```
<?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>
```

En este ejemplo solo se establecen unas cuantas de las posibles propiedades de la aplicación. Para obtener el conjunto completo de las propiedades de la aplicación, que permiten especificar determinados aspectos, como el tamaño y el fondo cromático de la ventana, la transparencia, el directorio de instalación predeterminado, los tipos de archivo asociados y los iconos de la aplicación, consulte [“Archivos descriptores de las aplicaciones de AIR”](#) en la página 214.

Creación de la página HTML de la aplicación

Es necesario crear una sencilla página HTML que sirva como archivo principal para la aplicación de AIR.

- 1 Abra el archivo `HelloWorld.html` para la edición. Añada el siguiente código HTML:

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

- 2 En la sección `<head>` del HTML, importe el archivo `AIRAliases.js`:

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

AIR define una propiedad denominada `runtime` en el objeto de la ventana HTML. La propiedad `runtime` proporciona acceso a las clases incorporadas de AIR, utilizando el nombre completo del paquete de la clase. Por ejemplo, para crear un objeto `File` de AIR se puede añadir la siguiente sentencia en JavaScript:

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

El archivo `AIRAliases.js` define los alias convenientes para las API de AIR más útiles. Con `AIRAliases.js` se puede reducir la referencia a la clase `File` del siguiente modo:

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

- 3 Bajo la etiqueta de `script AIRAliases`, añade otra etiqueta de `script` que contenga una función JavaScript para administrar el evento `onLoad`:

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

La función `appLoad()` simplemente llama a la función `air.trace()`. El mensaje de seguimiento se imprime en la consola de comandos cuando la aplicación se ejecuta utilizando ADL. Las sentencias `trace` pueden ser muy útiles en la depuración.

4 Guarde el archivo.

El archivo `HelloWorld.html` debe presentar ahora el siguiente aspecto:

```
<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>
```

Prueba de la aplicación

Para ejecutar y probar la aplicación desde la línea de comandos, emplee la utilidad AIR Debug Launcher (ADL). El ejecutable ADL se encuentra en el directorio `bin` del SDK de AIR. Si aún no ha configurado el SDK de AIR, consulte [“Instalación del SDK de AIR”](#) en la página 18.

- 1 Abra una consola de comandos o de shell. Cambie al directorio creado para este proyecto.
- 2 Ejecute el siguiente comando:

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

Se abrirá una ventana de AIR, mostrando la aplicación. Asimismo, la ventana de la consola muestra el mensaje resultante de la llamada a `air.trace()`.

Para obtener más información, consulte [“Archivos descriptores de las aplicaciones de AIR”](#) en la página 214.

Creación de un archivo de instalación de AIR

Cuando la aplicación se ejecute correctamente, puede emplear la utilidad ADT para empaquetar la aplicación en un archivo de instalación de AIR. Un archivo de instalación de AIR contiene todos los archivos de la aplicación, que se pueden distribuir a los usuarios. Se debe instalar Adobe AIR antes de instalar un archivo de AIR empaquetado.

Para garantizar la seguridad de la aplicación, todos los archivos de instalación de AIR se deben firmar digitalmente. Por motivos de desarrollo, se pueden generar certificados básicos con firma automática con ADT u otra herramienta de generación de certificados. También se puede adquirir un certificado con firma de código de una entidad comercial emisora de certificados como, por ejemplo, VeriSign o Thawte. Si los usuarios instalan un archivo de AIR con firma automática, el editor se muestra como “unknown” (desconocido) durante el proceso de instalación. Esto se debe a que el certificado con firma automática solo garantiza que el archivo de AIR no se ha modificado desde su creación original. No existe ningún método para evitar que alguien firme automáticamente un archivo de AIR de enmascaramiento y lo presente como su aplicación. Para los archivos de AIR distribuidos públicamente, se recomienda el uso de un certificado comercial verificable. Para obtener información general sobre los problemas de seguridad en AIR, consulte [Seguridad en AIR](#) (para desarrolladores de ActionScript) o [Seguridad en AIR](#) (para desarrolladores de HTML).

Generación de un certificado con firma automática y un par de claves

- ❖ Desde el símbolo del sistema, indique el siguiente comando (el ejecutable de ADT se ubica en el directorio `bin` del SDK de AIR):

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

ADT genera un archivo de almacén de claves denominado *sampleCert.pfx* que contiene un certificado y la clave privada relacionada.

En este ejemplo se utiliza el número mínimo de atributos que se pueden establecer para un certificado. El tipo de clave debe ser *1024-RSA* o *2048-RSA* (consulte “[Firma de aplicaciones de AIR](#)” en la página 199).

Creación de un archivo de instalación de AIR

- ❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

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

Se le solicitará la contraseña del archivo del almacén de claves.

El argumento `HelloWorld.air` es el archivo de AIR que genera ADT. `HelloWorld-app.xml` es el archivo descriptor de la aplicación. Los siguientes argumentos son los archivos utilizados por la aplicación. En este ejemplo solo se utilizan dos archivos, pero se puede incluir cualquier número de archivos y directorios. ADT comprueba el archivo de contenido principal, `HelloWorld.html` se incluye en el paquete, pero si se olvida incluir `AIRAliases.js`, la aplicación simplemente no funcionará.

Una vez creado el paquete de AIR, se puede instalar y ejecutar la aplicación haciendo doble clic en el archivo del paquete. También se puede escribir el nombre del archivo de AIR como comando en una ventana de comandos o de shell.

Pasos siguientes

En AIR, el código HTML y JavaScript se suele comportar tal y como lo haría en un navegador web típico. (De hecho, AIR utiliza el mismo motor de representación WebKit que se emplea en el navegador web Safari.) Sin embargo, existen algunas diferencias importantes que se deben conocer a la hora de desarrollar aplicaciones HTML en AIR. Para obtener más información sobre estas diferencias y otros temas importantes, consulte [Programming HTML and JavaScript](#) (Programación con HTML y JavaScript; en inglés).

Creación de la primera aplicación de AIR de escritorio con el SDK de Flex

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear una sencilla aplicación "Hello World" de AIR basada en SWF utilizando el SDK de Flex. Este tutorial muestra como compilar, probar y empaquetar una aplicación de AIR con las herramientas de la línea de comandos proporcionadas por el SDK de Flex (el SDK de Flex incluye el SDK de AIR).

Para comenzar, debe tener instalado el motor de ejecución y configurar Adobe® Flex™. En este tutorial se utiliza el compilador *AMXMLC*, *AIR Debug Launcher* (ADL) y *AIR Developer Tool* (ADT). Estos programas se pueden encontrar en el directorio `bin` del SDK de Flex (consulte “[Configuración del SDK de Flex](#)” en la página 20).

Creación del archivo descriptor de la aplicación de AIR

En esta sección se describe cómo crear el descriptor de la aplicación, que es un archivo XML con la siguiente estructura:

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

1 Cree un archivo XML denominado `HelloWorld-app.xml` y guárdelo en el directorio del proyecto.

2 Añada el elemento `<application>`, incluyendo el atributo de espacio de nombres de AIR:

`<application xmlns="http://ns.adobe.com/air/application/2.7">` El último segmento del espacio de nombres, "2.7", especifica la versión del motor de ejecución que requiere la aplicación.

3 Añada el elemento `<id>`:

`<id>samples.flex.HelloWorld</id>` El ID de la aplicación la identifica de forma exclusiva junto con el ID de editor (que AIR obtiene del certificado utilizado para firmar el paquete de la aplicación). La forma recomendada es una cadena de estilo DNS inversa delimitada por puntos como, por ejemplo, "com.company.AppName". El ID de la aplicación se utiliza para la instalación, el acceso al directorio privado de almacenamiento del sistema de archivos de la aplicación, el acceso al almacenamiento cifrado privado y la comunicación entre aplicaciones.

4 Agregue el elemento `<versionNumber>`:

`<versionNumber>1.0</versionNumber>` Ayuda a los usuarios a determinar qué versión de la aplicación se está instalando.

Nota: si está utilizando AIR 2 o anterior, debe usar el elemento `<version>` en lugar de `<versionNumber>`.

5 Agregue el elemento `<filename>`:

`<filename>HelloWorld</filename>` Nombre utilizado para el ejecutable de la aplicación, el directorio de instalación y otras referencias a la aplicación en el sistema operativo.

6 Añada el elemento `<initialWindow>` que contiene los siguientes elementos secundarios para especificar las propiedades de la ventana de la aplicación inicial:

`<content>HelloWorld.swf</content>` Identifica el archivo SWF raíz para que se cargue AIR.

`<visible>>true</visible>` Hace visible a la ventana de forma inmediata.

`<width>400</width>` Establece la anchura de la ventana (en píxeles).

`<height>200</height>` Establece la altura de la ventana.

7 Guarde el archivo. El archivo descriptor de la aplicación completo debe presentar el siguiente aspecto:

```
<?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>
```

En este ejemplo solo se establecen unas cuantas de las posibles propiedades de la aplicación. Para obtener el conjunto completo de las propiedades de la aplicación, que permiten especificar determinados aspectos, como el tamaño y el fondo cromático de la ventana, la transparencia, el directorio de instalación predeterminado, los tipos de archivo asociados y los iconos de la aplicación, consulte [“Archivos descriptores de las aplicaciones de AIR”](#) en la página 214.

Escritura del código de la aplicación

***Nota:** las aplicaciones de AIR basadas en SWF se pueden utilizar como clase principal definida con MXML o con Adobe® ActionScript® 3.0. En este ejemplo se utiliza un archivo MXML para definir su clase principal. El proceso para crear una aplicación de AIR con una clase ActionScript principal es similar. En lugar de compilar un archivo MXML en el archivo SWF, se compila el archivo de clase de ActionScript. Al utilizar ActionScript, la clase principal debe ampliar `flash.display.Sprite`.*

Al igual que sucede con todas las aplicaciones basadas en Flex, las aplicaciones de AIR creadas con la arquitectura de Flex contienen un archivo MXML principal. Las aplicaciones de AIR de escritorio utilizan el componente `WindowedApplication` como elemento raíz en lugar del componente `Application`. El componente `WindowedApplication` proporciona propiedades, métodos y eventos para controlar la aplicación y su ventana inicial. En las plataformas y perfiles para los que AIR no admite varias ventanas, continúe utilizando el componente `Application`. En las aplicaciones móviles de Flex, también se pueden usar los componentes `View` o `TabbedViewNavigatorApplication`.

El siguiente procedimiento crea la aplicación Hello World:

- 1 Con el uso de un editor de texto, cree un archivo denominado `HelloWorld.mxml` y añada el siguiente código MXML:

```
<?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 A continuación, añada un componente `Label` a la aplicación (sitúelo dentro de la etiqueta `WindowedApplication`).
- 3 Establezca la propiedad `text` del componente `Label` en `"Hello AIR"`.
- 4 Defina las restricciones de diseño para mantenerlo siempre centrado.

En el siguiente ejemplo se muestra el código hasta el momento:

```
<?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>
```

Compilación de la aplicación

Antes de que se pueda ejecutar y depurar la aplicación, compile el código MXML en un archivo SWF utilizando el compilador amxmlc. El compilador amxmlc se encuentra en el directorio bin del SDK de Flex. Si lo desea, el entorno de ruta del equipo se puede configurar para que incluya el directorio bin del SDK de Flex. Al establecer la ruta, se facilita la ejecución de las utilidades en la línea de comandos.

- 1 Abra un shell de comandos o una terminal y desplácese a la carpeta del proyecto de la aplicación de AIR.
- 2 Indique el siguiente comando:

```
amxmlc HelloWorld.mxml
```

Con la ejecución de amxmlc se genera HelloWorld.swf, que contiene el código compilado de la aplicación.

Nota: si la aplicación no se compila, corrija la sintaxis o los errores ortográficos. Los errores y los avisos se muestran en la ventana de la consola utilizada para ejecutar el compilador amxmlc.

Para obtener más información, consulte “[Compilación de archivos de origen MXML y ActionScript para AIR](#)” en la página 164.

Prueba de la aplicación

Para ejecutar y probar la aplicación desde la línea de comandos, utilice AIR Debug Launcher (ADL) para iniciar la aplicación utilizando su archivo descriptor. (ADL se encuentra en el directorio bin del SDK de Flex.)

- ❖ Desde el símbolo del sistema, indique el siguiente comando:

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

La aplicación de AIR resultante tiene un aspecto similar al de esta ilustración:



Con el uso de las propiedades horizontalCenter y verticalCenter del control Label, el texto se sitúa en el centro de la ventana. Mueva o cambie el tamaño de la ventana tal y como lo haría en cualquier otra aplicación de escritorio.

Para obtener más información, consulte “[AIR Debug Launcher \(ADL\)](#)” en la página 168.

Creación de un archivo de instalación de AIR

Cuando la aplicación se ejecute correctamente, puede emplear la utilidad ADT para empaquetar la aplicación en un archivo de instalación de AIR. Un archivo de instalación de AIR contiene todos los archivos de la aplicación, que se pueden distribuir a los usuarios. Se debe instalar Adobe AIR antes de instalar un archivo de AIR empaquetado.

Para garantizar la seguridad de la aplicación, todos los archivos de instalación de AIR se deben firmar digitalmente. Por motivos de desarrollo, se pueden generar certificados básicos con firma automática con ADT u otra herramienta de generación de certificados. También puede adquirir un certificado de firma de código comercial en una entidad emisora de certificados. Si los usuarios instalan un archivo de AIR con firma automática, el editor se muestra como “unknown” (desconocido) durante el proceso de instalación. Esto se debe a que el certificado con firma automática solo garantiza que el archivo de AIR no se ha modificado desde su creación original. No existe ningún método para evitar que alguien firme automáticamente un archivo de AIR de enmascaramiento y lo presente como su aplicación. Para los archivos de AIR distribuidos públicamente, se recomienda el uso de un certificado comercial verificable. Para obtener información general sobre los problemas de seguridad en AIR, consulte [Seguridad en AIR](#) (para desarrolladores de ActionScript) o [Seguridad en AIR](#) (para desarrolladores de HTML).

Generación de un certificado con firma automática y un par de claves

- ❖ Desde el símbolo del sistema, indique el siguiente comando (el ejecutable de ADT se ubica en el directorio bin del SDK de Flex):

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

En este ejemplo se utiliza el número mínimo de atributos que se pueden establecer para un certificado. El tipo de clave debe ser *1024-RSA* o *2048-RSA* (consulte “[Firma de aplicaciones de AIR](#)” en la página 199).

Creación del paquete de AIR

- ❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

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

Se le solicitará la contraseña del archivo del almacén de claves. Escriba la contraseña y presione Intro. Los caracteres de la contraseña no se muestran por razones de seguridad.

El argumento HelloWorld.air es el archivo de AIR que genera ADT. HelloWorld-app.xml es el archivo descriptor de la aplicación. Los siguientes argumentos son los archivos utilizados por la aplicación. En este ejemplo solo se utilizan tres archivos, pero se puede incluir cualquier número de archivos y directorios.

Una vez creado el paquete de AIR, se puede instalar y ejecutar la aplicación haciendo doble clic en el archivo del paquete. También se puede escribir el nombre del archivo de AIR como comando en una ventana de comandos o de shell.

Para obtener más información, consulte “[Empaquetado de un archivo de instalación de AIR de escritorio](#)” en la página 57.

Creación de la primera aplicación de AIR para Android con el SDK de Flex

Para comenzar, debe tener instalados y configurados los SDKs de AIR y Flex. Este tutorial utiliza el compilador *AMXMLC* del SDK de Flex, *AIR Debug Launcher* (ADL) y *AIR Developer Tool* (ADT) del SDK de AIR. Consulte “[Configuración del SDK de Flex](#)” en la página 20.

También debe descargar e instalar el SDK de Android desde el sitio web de Android, tal y como se describe en: [Android Developers: Installing the SDK](#) (Desarrolladores de Android: Instalación del SDK; en inglés).

Nota: para obtener más información sobre el desarrollo de iPhone, consulte [Creación de una aplicación Hello World para iPhone con Flash Professional CS5](#).

Creación del archivo descriptor de la aplicación de AIR

En esta sección se describe cómo crear el descriptor de la aplicación, que es un archivo XML con la siguiente estructura:

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

- 1 Cree un archivo XML denominado `HelloWorld-app.xml` y guárdelo en el directorio del proyecto.
- 2 Añada el elemento `<application>`, incluyendo el atributo de espacio de nombres de AIR:
`<application xmlns="http://ns.adobe.com/air/application/2.7">` El último segmento del espacio de nombres, "2.7", especifica la versión del motor de ejecución que requiere la aplicación.
- 3 Añada el elemento `<id>`:
`<id>samples.android.HelloWorld</id>` El ID de la aplicación la identifica de forma exclusiva junto con el ID de editor (que AIR obtiene del certificado utilizado para firmar el paquete de la aplicación). La forma recomendada es una cadena de estilo DNS inversa delimitada por puntos como, por ejemplo, "com.company.AppName".
- 4 Agregue el elemento `<versionNumber>`:
`<versionNumber>0.0.1</versionNumber>` Ayuda a los usuarios a determinar qué versión de la aplicación se está instalando.
- 5 Agregue el elemento `<filename>`:
`<filename>HelloWorld</filename>` Nombre utilizado para el ejecutable de la aplicación, el directorio de instalación y otras referencias a la aplicación en el sistema operativo.
- 6 Añada el elemento `<initialWindow>` que contiene los siguientes elementos secundarios para especificar las propiedades de la ventana de la aplicación inicial:
`<content>HelloWorld.swf</content>` Identifica el archivo HTML raíz para que se cargue AIR.
- 7 Agregue el elemento `<supportedProfiles>`.
`<supportedProfiles>mobileDevice</supportedProfiles>` Especifica que la aplicación solo se ejecuta en el perfil móvil.
- 8 Guarde el archivo. El archivo descriptor de la aplicación completo debe presentar el siguiente aspecto:

```
<?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>
```

En este ejemplo solo se establecen unas cuantas de las posibles propiedades de la aplicación. Existen otras configuraciones que se pueden usar en el archivo descriptor de la aplicación. Por ejemplo, puede añadir `<fullScreen>true</fullScreen>` al elemento `initialWindow` para crear una aplicación de pantalla completa. Para activar la depuración remota y las funciones controladas por acceso en Android, también se tendrán que añadir permisos de Android al descriptor de la aplicación. Los permisos no son necesarios para esta aplicación de ejemplo, por lo que no es necesario añadirlos ahora.

Para obtener más información, consulte “[Configuración de las propiedades de una aplicación móvil](#)” en la página 76.

Escritura del código de la aplicación

Cree un archivo denominado `HelloWorld.as` y añada el siguiente código utilizando un editor de texto:

```
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 );
        }
    }
}
```

Compilación de la aplicación

Antes de que se pueda ejecutar y depurar la aplicación, compile el código MXML en un archivo SWF utilizando el compilador `amxmlc`. El compilador `amxmlc` se encuentra en el directorio `bin` del SDK de Flex. Si lo desea, el entorno de ruta del equipo se puede configurar para que incluya el directorio `bin` del SDK de Flex. Al establecer la ruta, se facilita la ejecución de las utilidades en la línea de comandos.

- 1 Abra un shell de comandos o una terminal y desplácese a la carpeta del proyecto de la aplicación de AIR.
- 2 Indique el siguiente comando:

```
amxmlc HelloWorld.as
```

Con la ejecución de `amxmlc` se genera `HelloWorld.swf`, que contiene el código compilado de la aplicación.

Nota: si la aplicación no se compila, corrija la sintaxis o los errores ortográficos. Los errores y los avisos se muestran en la ventana de la consola utilizada para ejecutar el compilador `amxmlc`.

Para obtener más información, consulte [“Compilación de archivos de origen MXML y ActionScript para AIR”](#) en la página 164.

Prueba de la aplicación

Para ejecutar y probar la aplicación desde la línea de comandos, utilice AIR Debug Launcher (ADL) para iniciar la aplicación utilizando su archivo descriptor. (ADL se encuentra en el directorio bin del SDK de Flex y de AIR.)

- ❖ Desde el símbolo del sistema, indique el siguiente comando:

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

Para obtener más información, consulte [“Simulación del dispositivo utilizando ADL”](#) en la página 106.

Creación del archivo del paquete de APK

Cuando la aplicación se ejecute correctamente, puede emplear la utilidad ADT para empaquetar la aplicación en un archivo de paquete de APK. Un archivo del paquete de APK es el formato de archivo de la aplicación de Android nativo que se puede distribuir para los usuarios.

Todas las aplicaciones de Android se deben firmar. A diferencia de los archivos de AIR, se suelen firmar las aplicaciones de Android con un certificado de firma automática. El sistema operativo Android no intenta establecer la identidad del desarrollador de la aplicación. Se puede emplear un certificado generado por ADT para firmar los paquetes de Android. Los certificados usados para las aplicaciones enviadas a Android Market, deben tener un periodo de validez de al menos 25 años.

Generación de un certificado con firma automática y un par de claves

- ❖ Desde el símbolo del sistema, indique el siguiente comando (el ejecutable de ADT se ubica en el directorio bin del SDK de Flex):

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

En este ejemplo se utiliza el número mínimo de atributos que se pueden establecer para un certificado. El tipo de clave debe ser *1024-RSA* o *2048-RSA* (consulte el comando [“Comando certificate de ADT”](#) en la página 183).

Creación del paquete de AIR

- ❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

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

Se le solicitará la contraseña del archivo del almacén de claves. Escriba la contraseña y presione Intro.

Para obtener más información, consulte [“Empaquetado de una aplicación de AIR móvil”](#) en la página 99.

Instalación del motor de ejecución de AIR

Se puede instalar la versión más reciente del motor de ejecución de AIR en el dispositivo desde Android Market. También se puede instalar el motor de ejecución incluido en el SDK en un dispositivo o emulador de Android.

- ❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

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

Establezca el indicador `-platformsdk` en el directorio del SDK de Android (especifique el elemento principal de la carpeta de herramientas).

ADT instala Runtime.apk incluido en el SDK.

Para obtener más información, consulte [“Instalación de aplicaciones y el motor de ejecución de AIR para desarrollo”](#) en la página 115.

Instalación de la aplicación de AIR

- ❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

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

Establezca el indicador `-platformsdk` en el directorio del SDK de Android (especifique el elemento principal de la carpeta de herramientas).

Para obtener más información, consulte [“Instalación de aplicaciones y el motor de ejecución de AIR para desarrollo”](#) en la página 115.

La aplicación se puede iniciar punteando el icono de la aplicación en la pantalla del dispositivo o emulador.

Capítulo 6: Desarrollo de aplicaciones de AIR para el escritorio

Flujo de trabajo para el desarrollo de una aplicación de escritorio de AIR

El flujo de trabajo básico para el desarrollo de una aplicación de AIR es el mismo que la mayoría de modelos de desarrollo tradicionales: código, compilación, prueba y, al final del ciclo, empaquetado en un archivo instalador.

Se puede escribir el código de la aplicación utilizando Flash, Flex y ActionScript y realizar la compilación utilizando Flash Professional, Flash Builder o los compiladores de la línea de comandos mxmcl y compc. También se puede escribir código de la aplicación utilizando HTML y JavaScript y omitir el paso de compilación.

Las aplicaciones de AIR de escritorio se pueden probar con la herramienta ADL, que ejecuta una aplicación sin que sea necesario empaquetarse e instalarse en primer lugar. Flash Professional, Flash Builder, Dreamweaver, y el IDE de Aptana se integran con el depurador de Flash. También se puede iniciar la herramienta depuradora, FDB, manualmente al utilizar ADL desde la línea de comandos. ADL, por sí misma, muestra errores y sentencias trace.

Todas las aplicaciones de AIR se pueden empaquetar en un archivo de instalación. El formato de archivo de AIR multiplataforma se recomienda a no ser que:

- Necesite acceder a API dependientes de la plataforma, por ejemplo, la clase NativeProcess.
- Su aplicación utilice extensiones nativas.

En estos casos, se puede empaquetar una aplicación de AIR como un archivo de instalación nativo específico de la plataforma.

Aplicaciones basadas en SWF

- 1 Escriba el código MXML o ActionScript.
- 2 Cree los recursos necesarios como, por ejemplo, archivos de mapa de bits del icono.
- 3 Cree el descriptor de la aplicación.
- 4 Compile el código ActionScript.
- 5 Pruebe la aplicación.
- 6 Empaquete y firme como un archivo de AIR con el destino *air*.

Aplicaciones basadas en HTML

- 1 Escriba el código HTML y JavaScript.
- 2 Cree los recursos necesarios como, por ejemplo, archivos de mapa de bits del icono.
- 3 Cree el descriptor de la aplicación.
- 4 Pruebe la aplicación.
- 5 Empaquete y firme como un archivo de AIR con el destino *air*.

Creación de instaladores nativos para aplicaciones de AIR

- 1 Escriba el código (ActionScript o HTML y JavaScript).
- 2 Cree los recursos necesarios como, por ejemplo, archivos de mapa de bits del icono.
- 3 Cree el descriptor de la aplicación especificando el perfil *extendedDesktop*.
- 4 Compile cualquier código de ActionScript.
- 5 Pruebe la aplicación.
- 6 Empaque la aplicación en cada plataforma de destino con el destino *native*.

Nota: debe haber creado el archivo de instalación nativo para una plataforma de destino en dicha plataforma. No es posible, por ejemplo, crear un archivo de instalación de Windows en Mac. Puede utilizar una máquina virtual, como VMWare, para ejecutar varias plataformas en el mismo hardware del ordenador.

Creación de aplicaciones de AIR con un paquete de motor de ejecución captador

- 1 Escriba el código (ActionScript o HTML y JavaScript).
- 2 Cree los recursos necesarios como, por ejemplo, archivos de mapa de bits del icono.
- 3 Cree el descriptor de la aplicación especificando el perfil *extendedDesktop*.
- 4 Compile cualquier código de ActionScript.
- 5 Pruebe la aplicación.
- 6 Empaque la aplicación en cada plataforma de destino con el destino *bundle*.
- 7 Cree un programa de instalación con los archivos del paquete. (El SDK de AIR no incluye herramientas para crear este archivo de instalación, pero hay disponibles muchos kits de herramientas de otros fabricantes.)

Nota: debe haber creado el paquete para una plataforma de destino en dicha plataforma. No es posible, por ejemplo, crear un paquete de Windows en Mac. Puede utilizar una máquina virtual, como VMWare, para ejecutar varias plataformas en el mismo hardware del ordenador.

Configuración de las propiedades de una aplicación de escritorio

Defina las propiedades básicas de la aplicación en el archivo descriptor de la aplicación. En esta sección se analizan las propiedades relevantes en las aplicaciones de AIR de escritorio. Los elementos del archivo descriptor de la aplicación se describen detalladamente en “[Archivos descriptores de las aplicaciones de AIR](#)” en la página 214.

Versión necesaria del motor de ejecución de AIR

Especifique la versión del motor de ejecución de AIR que necesita la aplicación utilizando el espacio de nombres del archivo descriptor de la aplicación.

El espacio de nombres, asignado en el elemento `application`, determina, en gran parte, qué funciones puede utilizar la aplicación. Por ejemplo, si su aplicación utiliza el espacio de nombres de AIR 1.5 y el usuario tiene instalado AIR 3.0, la aplicación ve el comportamiento de AIR 1.5 (aunque el comportamiento se haya modificado en AIR 3.0). Solo cuando se cambia el espacio de nombres y se publica una actualización, la aplicación tendrá acceso a las nuevas funciones y características. Los cambios de WebKit y de seguridad son las principales excepciones a esta política.

Especifique el espacio de nombres utilizando el atributo `xmlns` del elemento raíz `application`:

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

Más temas de ayuda

[“application”](#) en la página 219

Identidad de la aplicación

Las distintas configuraciones deben ser exclusivas para cada aplicación que se publique. La configuración exclusiva incluye el ID, nombre y nombre del archivo.

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

Más temas de ayuda

[“id”](#) en la página 236

[“filename”](#) en la página 231

[“name”](#) en la página 244

Versión de la aplicación

En las versiones de AIR anteriores a AIR 2.5, especifique la aplicación en el elemento `version`. Se puede utilizar cualquier cadena. El motor de ejecución de AIR no interpreta la cadena; “2.0” no se considera versión superior a “1.0.”

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

En AIR 2.5 y posterior, especifique la versión de la aplicación en el elemento `versionNumber`. El elemento `version` ya no puede volver a utilizarse. Cuando se especifica un valor para `versionNumber`, se debe utilizar una secuencia de hasta tres nombres separados por puntos; por ejemplo, “0.1.2”. Cada segmento del número de versión puede tener hasta tres dígitos. (Es decir, “999.999.999” es el mayor número de versión permitido.) No es necesario incluir los tres segmentos en el número; “1” y “1.0” son también números de la versión legal.

También se puede especificar una etiqueta para la versión utilizando el elemento `versionLabel`. Cuando se añade una etiqueta de versión, se muestra en lugar del número de versión en estos lugares como cuadros de diálogo del instalador de aplicaciones de AIR.

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

Más temas de ayuda

[“version”](#) en la página 252

[“versionLabel”](#) en la página 253

[“versionNumber”](#) en la página 253

Propiedades de la ventana principal

Cuando AIR inicia una aplicación en el escritorio, crea una ventana y carga el archivo SWF principal o la página HTML en el mismo. AIR usa los elementos secundarios del elemento `initialWindow` para controlar el aspecto inicial y el comportamiento de esta ventana de la aplicación inicial.

- **content:** principal archivo SWF de la aplicación en el elemento secundario `content` de `initialWindow`. Cuando el objetivo son los dispositivos del perfil de escritorio, se puede emplear un archivo SWF o HTML.

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

El archivo se debe incluir en el paquete de AIR (utilizando ADT o su IDE). Simplemente hacer referencia al nombre en el descriptor de la aplicación no hace que el archivo se incluya en el paquete automáticamente.

- **depthAndStencil:** especifica el uso del búfer de estencil o de profundidad. Normalmente estos búferes se utilizan al trabajar con contenido en 3D.

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

- **height:** altura de la ventana inicial.
- **maximizable:** indica si se muestra el fondo cromático para maximizar la ventana.
- **maxSize:** tamaño máximo permitido.
- **minimizable:** indica si se muestra el fondo cromático para minimizar la ventana.
- **minSize:** tamaño mínimo permitido.
- **renderMode:** en AIR 3 o posterior, el modo de procesamiento se puede establecer en *auto*, *cpu*, *direct* o *gpu* para aplicaciones de escritorio. En versiones anteriores de AIR, este ajuste se omitía en las plataformas de escritorio. El ajuste de `renderMode` no se puede modificar en tiempo de ejecución.
 - *auto:* básicamente igual que el modo *cpu*.
 - *cpu:* los objetos de visualización se procesan y se copian en la memoria de visualización como software. `StageVideo` solo está disponible si una ventana está en modo de pantalla completa. `Stage3D` utiliza el procesador de software.
 - *direct:* los objetos de visualización se procesan con el software del motor de ejecución, pero al copiar el fotograma procesado en la memoria de visualización (blitting), se acelera por hardware. `StageVideo` está disponible. `Stage3D` usa aceleración por hardware, si es posible. Si la transparencia de la ventana se establece en *true*, la ventana “volverá” al procesamiento y blitting por software.

Nota: para poder aprovechar la aceleración de GPU del contenido de Flash con plataformas de AIR para móviles, Adobe recomienda utilizar `renderMode="direct"` (es decir, `Stage3D`) en vez de `renderMode="gpu"`. Adobe oficialmente admite y recomienda las siguientes arquitecturas basadas en `Stage3D`: `Starling (2D)` y `Away3D (3D)`. Para obtener más información sobre `Stage3D` y `Starling/Away3D`, consulte <http://gaming.adobe.com/getstarted/>.

- *gpu:* la aceleración de hardware se usa, si está disponible.
- **requestedDisplayResolution:** especifica si la aplicación debe usar resolución *estándar* o *alta* en ordenadores MacBook Pro con pantallas de alta resolución. En el resto de plataformas el valor se pasa por alto. Si el valor es *estándar*, cada píxel de escenario se representa como cuatro píxeles en pantalla. Si la resolución es *alta*, cada píxel de escenario corresponde a un solo píxel físico en pantalla. El valor especificado se usa para todas las ventanas de la aplicación. El uso de `requestedDisplayResolution` para aplicaciones de AIR de escritorio (como elemento secundario de `initialWindow`) está disponibles a partir de AIR 3.6.
- **resizable:** indica si se muestra el fondo cromático de la ventana.

- **systemChrome:** indica si se utiliza la apariencia de la ventana del sistema operativo estándar. La configuración de systemChrome de una ventana no se puede cambiar en tiempo de ejecución.
- **title:** título de la ventana.
- **transparent:** indica si la ventana es de mezcla alfa frente al fondo. La ventana no puede utilizar el fondo cromático del sistema si la transparencia está activada. La configuración transparente de una ventana no se puede cambiar en tiempo de ejecución.
- **visible:** indica si la ventana se puede ver tras su creación. De forma predeterminada, la ventana no se ve inicialmente para que la aplicación pueda dibujar su contenido antes de hacerse visible.
- **width:** anchura de la ventana.
- **x:** posición horizontal de la ventana.
- **y:** posición vertical de la ventana.

Más temas de ayuda

[“content”](#) en la página 225

[“depthAndStencil”](#) en la página 227

[“height”](#) en la página 235

[“maximizable”](#) en la página 243

[“maxSize”](#) en la página 243

[“minimizable”](#) en la página 244

[“minimizable”](#) en la página 244

[“minSize”](#) en la página 244

[“renderMode”](#) en la página 247

[“requestedDisplayResolution”](#) en la página 247

[“resizable”](#) en la página 248

[“systemChrome”](#) en la página 251

[“title”](#) en la página 252

[“transparent”](#) en la página 252

[“visible”](#) en la página 254

[“width”](#) en la página 254

[“x”](#) en la página 254

[“y”](#) en la página 255

Funciones de escritorio

Los siguientes elementos controlan las funciones de actualización y de instalación del escritorio.

- **customUpdateUI:** permite proporcionar cuadros de diálogos propios para actualizar una aplicación. Si se establece en `false` (valor predeterminado), se utilizan los cuadros de diálogo de AIR estándar.

- `fileTypes`: especifica los tipos de archivos que la aplicación desearía registrar como aplicación de apertura predeterminada. Si otra aplicación ya es el objeto opener predeterminado para un tipo de archivo, AIR no omite el registro existente. Sin embargo, la aplicación puede omitir el registro en tiempo de ejecución utilizando el método `setAsDefaultApplication()` del objeto `NativeApplication`. Resulta adecuado solicitar permiso del usuario antes de omitir sus asociaciones de tipo de archivo existentes.

***Nota:** el registro del tipo de archivo se omite al empaquetar una aplicación como paquete de motor de ejecución captador (con el destino `-bundle`). Para registrar un tipo de archivo determinado, se debe crear un programa de instalación que lleve a cabo el registro.*

- `installFolder`: especifica una ruta relativa a la carpeta de instalación de la aplicación estándar en la que se encuentra instalada la aplicación. Esta configuración se puede emplear para proporcionar un nombre de carpeta personalizado, así como para agrupar varias aplicaciones en una carpeta común.
- `programMenuFolder`: especifica la jerarquía de menús para el menú Todos los programas de Windows. Esta configuración se puede usar para agrupar varias aplicaciones en un menú común. Si no se especifica ninguna carpeta de menú, el método abreviado de la aplicación se añade directamente al menú principal.

Más temas de ayuda

[“customUpdateUI”](#) en la página 226

[“fileTypes”](#) en la página 232

[“installFolder”](#) en la página 240

[“programMenuFolder”](#) en la página 246

Perfiles admitidos

Si la aplicación solo es útil en el escritorio, su instalación se puede evitar en los dispositivos en otro perfil excluyendo dicho perfil de la lista de perfiles admitidos. Si la aplicación utiliza la clase `NativeProcess` o extensiones nativas, es necesario admitir el perfil `extendedDesktop`.

Si el elemento `supportedProfile` se deja fuera del descriptor de la aplicación, se da por sentado que la aplicación admite todos los perfiles definidos. Para limitar la aplicación a una lista específica de perfiles, incluya los perfiles, separados por un espacio en blanco:

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

Para obtener una lista de las clases de `ActionScript` admitidas en el perfil `desktop` y `extendedDesktop`, consulte [“Capacidades en diferentes perfiles”](#) en la página 257.

Más temas de ayuda

[“supportedProfiles”](#) en la página 250

Extensiones nativas necesarias

Las aplicaciones que admiten el perfil `extendedDesktop` pueden utilizar extensiones nativas.

Declare todas las extensiones nativas que la aplicación de AIR utiliza en el descriptor de la aplicación. El siguiente ejemplo ilustra la sintaxis para especificar dos extensiones nativas necesarias:

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

El elemento `extensionID` tiene el mismo valor que `id` en el archivo descriptor de la extensión. El archivo descriptor de la extensión es un archivo XML denominado `extension.xml`. Se empaqueta en el archivo ANE que se recibe del desarrollador de extensiones nativas.

Iconos de la aplicación

En el escritorio, los iconos especificados en el descriptor de la aplicación se utilizan como iconos de menú del programa, métodos abreviados y archivos de la aplicación. El icono de la aplicación se debe proporcionar como conjunto de imágenes PNG de 16x16, 32x32, 48x48 y 128x128 píxeles. Especifique la ruta a los archivos de icono en el elemento de icono del archivo descriptor de la aplicación:

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

Si no proporciona un icono de un tamaño determinado, el siguiente tamaño mayor se utiliza y se escala para ajustarse. Si no se proporciona ningún icono, se utilizará un icono del sistema predeterminado.

Más temas de ayuda

[“icon”](#) en la página 235

[“imageNxN”](#) en la página 237

Configuración omitida

Las aplicaciones del escritorio omiten la configuración de la aplicación que se aplica a funciones de perfil móvil. Los valores de configuración omitidos son:

- `android`
- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `iPhone`
- `renderMode` (anterior a AIR 3)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Depuración de una aplicación de AIR de escritorio

Si la aplicación se está desarrollando con un IDE como, por ejemplo, Flash Builder, Flash Professional o Dreamweaver, las herramientas de depuración están integradas normalmente. La aplicación se puede depurar simplemente iniciándola en modo de depuración. Si no se está utilizando ningún IDE que admita la depuración de forma directa, se puede emplear AIR Debug Launcher (ADL) y Flash Debugger (FDB) para ayudar en la depuración de la aplicación.

Más temas de ayuda

[De Monsters: Depurador Monster](#)

“[Depuración con el introspector HTML de AIR](#)” en la página 295

Ejecución de una aplicación con ADL

Es posible ejecutar una aplicación de AIR sin empaquetarla e instalarla utilizando ADL. Transmite el archivo descriptor de la aplicación a ADL como parámetro, tal y como se muestra en el siguiente ejemplo (el código ActionScript de la aplicación se debe compilar en primer lugar):

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

ADL imprime las sentencias trace, las excepciones en tiempo de ejecución y los errores de análisis HTML en la ventana de terminal. Si un proceso de FDB está esperando una conexión entrante, ADL se conectará al depurador.

También puede utilizar ADL para depurar una aplicación de AIR que utilice extensiones nativas. Por ejemplo:

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

Más temas de ayuda

“[AIR Debug Launcher \(ADL\)](#)” en la página 168

Impresión de sentencias trace

Para imprimir sentencias trace en la consola que se utiliza para ejecutar ADL, añada sentencias trace al código con la función `trace()`:

Nota: Si las sentencias `trace()` no se visualizan en la consola, asegúrese de que no ha especificado `ErrorReportingEnable` ni `TraceOutputFileEnable` en el archivo `mm.cfg`. Para obtener más información sobre la ubicación de este archivo en cada plataforma, consulte [Edición del archivo mm.cfg](#) (en inglés).

Ejemplo de ActionScript:

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

Ejemplo de JavaScript:

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

En el código JavaScript se pueden utilizar las funciones `alert()` y `confirm()` para mostrar los mensajes de depuración de la aplicación. Además, los números de línea para los errores de sintaxis, así como las excepciones de JavaScript sin capturar se imprimen en la consola.

Nota: para utilizar el prefijo `air` que muestra en el ejemplo de JavaScript, debe importar el archivo `AIRAliases.js` en la página. El archivo se ubica en el directorio `frameworks` del SDK de AIR.

Conexión a Flash Debugger (FDB)

Para depurar aplicaciones de AIR con Flash Debugger, inicie una sesión de FDB y posteriormente inicie la aplicación con ADL.

Nota: en las aplicaciones de AIR basadas en SWF los archivos de origen de ActionScript se deben compilar con el indicador `-debug`. (En Flash Professional, seleccione la opción Permitir depuración en el cuadro de diálogo Configuración de publicación.)

1 Inicie FDB. El programa FDB se encuentra en el directorio `bin` del SDK de Flex.

La consola presenta el indicador de FDB: `<fdb>`

2 Ejecute el comando `run`: `<fdb>run` [Enter]

3 En otra consola de comandos o de shell, inicie una versión de depuración de la aplicación:

```
adl myApp.xml
```

4 Utilice comandos de FDB para definir los puntos de corte según proceda.

5 Escriba: `continue` [Intro]

Si una aplicación de AIR se basa en SWF, el depurador solo controla la ejecución de código ActionScript. Si la aplicación de AIR se basa en HTML, el depurador solo controla la ejecución de código JavaScript.

Para ejecutar ADL sin conexión con el depurador, incluya la opción `-nodebug`:

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

Para obtener información básica sobre los comandos de FDB, ejecute el comando `help`:

```
<fdb>help
```

 [Enter]

Para obtener información sobre los comandos de FDB, consulte [Using the command-line debugger commands](#) (Uso de los comandos del depurador de la línea de comandos) en la documentación de Flex (en inglés).

Empaquetado de un archivo de instalación de AIR de escritorio

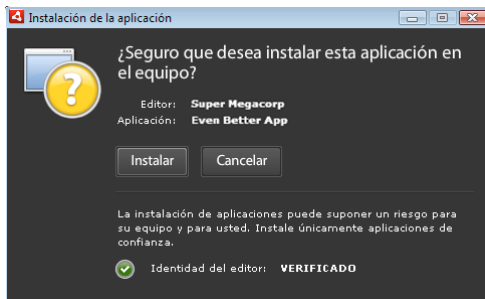
Cada aplicación de AIR debe incluir, como mínimo, un archivo descriptor de la aplicación y un archivo SWF o HTML principal. Cualquier otro recurso que se vaya a instalar con la aplicación se debe empaquetar también en el archivo de AIR.

En este artículo se analiza el empaquetado de una aplicación de AIR utilizando las herramientas de la línea de comandos incluidas con el SDK. Para obtener información sobre el empaquetado de una aplicación mediante una de las herramientas de edición de Adobe, consulte la siguiente referencia:

- Adobe® Flex® Builder™, consulte [Empaquetado de aplicaciones de AIR con Flex Builder](#).
- Adobe® Flash® Builder™, consulte [Empaquetado de aplicaciones de AIR con Flash Builder](#).
- Adobe® Flash® Professional, consulte [Publicación para Adobe AIR](#).
- Adobe® Dreamweaver®, consulte [Creación de una aplicación de AIR en Dreamweaver](#).

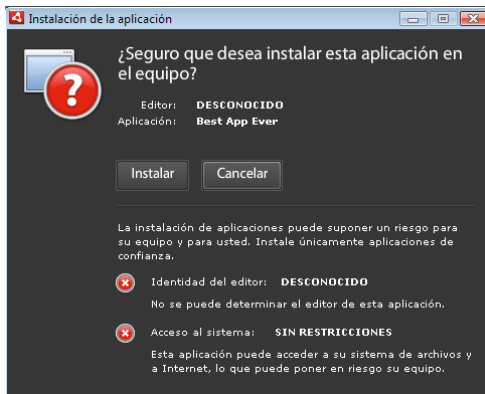
Todos los archivos instaladores de AIR deben firmarse con un certificado digital. El instalador de AIR utiliza la firma para verificar que el archivo de la aplicación no ha sido modificado desde que se firmó. Puede utilizar un certificado de firma de código de una entidad emisora de certificados o un certificado con firma automática.

Cuando se utiliza un certificado emitido por una autoridad de certificación de confianza, se ofrece a los usuarios de la aplicación cierta seguridad respecto de su identidad como editor de la aplicación. El cuadro de diálogo de instalación refleja el hecho de que su identidad se verifica mediante la entidad emisora de certificados:



Cuadro de diálogo de confirmación de instalación firmado por un certificado de confianza

Cuando se utiliza un certificado con firma automática, los usuarios no pueden verificar su identidad como firmante. Un certificado con firma automática también debilita la garantía de que el paquete no se haya modificado. (Esto se debe a que un archivo de instalación de confianza se puede sustituir por una falsificación antes de que llegue al usuario.) El cuadro de diálogo de instalación refleja que la identidad del editor no puede comprobarse. Los usuarios están arriesgando más su seguridad cuando instalan la aplicación:



Cuadro de diálogo de confirmación de instalación firmado por un certificado con firma automática

Se puede empaquetar y firmar un archivo de AIR en un solo paso con el comando `-package` de ADT. También se puede crear un paquete intermedio sin firmar con el comando `-prepare`, firmando después el paquete intermedio con el comando `-sign` en un paso separado.

Nota: las versiones de Java 1.5 y posteriores no aceptan caracteres ASCII superior en contraseñas utilizadas para proteger archivos de certificado PKCS12. Cuando se cree o se exporte el archivo de certificado de firma de código, utilice solamente caracteres ASCII normales en la contraseña.

Al firmar el paquete de instalación, ADT se pone en contacto automáticamente con el servidor de una autoridad de marcas de hora para verificar la hora. La marca de hora se incluye en el archivo de AIR. Un archivo de AIR que incluya una marca de hora verificada podrá instalarse en el futuro en cualquier momento. Si ADT no puede conectarse al servidor de marcas de hora, se cancela el empaquetado. La opción de marca de hora puede pasarse por alto, pero sin marca de hora la aplicación de AIR ya no podrá instalarse una vez caducado el certificado que se utilizó para firmar el archivo de instalación.

Si se está creando un paquete para actualizar una aplicación de AIR existente, el paquete se debe firmar con el mismo certificado que la aplicación original. Si el certificado original se ha renovado o ha caducado en los últimos 180 días, o bien, si se desea cambiar a un nuevo certificado, se puede aplicar una firma de migración. La firma de migración implica firmar el archivo de AIR de la aplicación tanto con el nuevo como con el antiguo certificado. Utilice el comando `-migrate` para aplicar la firma de migración tal y como se describe en “[Comando migrate de ADT](#)” en la página 182.

Importante: *existe un estricto periodo de gracia de 180 días para aplicar una firma de migración una vez caducado el certificado original. Sin la firma de migración, los usuarios existentes deben desinstalar su aplicación existente antes de instalar la nueva versión. El periodo de gracia de días solo se aplica a las aplicaciones que especifican la versión de AIR 1.5.3, o superior, en el espacio de nombres del descriptor de la aplicación. No hay periodo de gracia cuando se utilizan versiones anteriores del motor de ejecución de AIR.*

Antes de AIR 1.1, no se admitían las firmas de migración. La aplicación se debe empaquetar con un SDK de versión 1.1 o posterior para aplicar una firma de migración.

Las aplicaciones implementadas utilizando archivos de AIR se denominan aplicaciones de perfil de escritorio. No es posible utilizar ADT para empaquetar un instalador nativo para una aplicación de AIR si el archivo descriptor de la aplicación no admite el perfil de escritorio. Este perfil se puede restringir utilizando el elemento `supportedProfiles` en el archivo descriptor de la aplicación. Consulte “[Perfiles de dispositivo](#)” en la página 256 y “[supportedProfiles](#)” en la página 250.

Nota: *los valores definidos en el archivo descriptor de la aplicación determinan la identidad de la aplicación de AIR y su ruta de instalación predeterminada. Consulte “[Archivos descriptores de las aplicaciones de AIR](#)” en la página 214.*

ID de editor

A partir de AIR 1.5.3, los de editor quedan desfasados. Las nuevas aplicaciones (publicadas en un principio con AIR 1.5.3 o superior) no necesitan ni deben especificar un ID de editor.

Al actualizar las aplicaciones publicadas con versiones anteriores de AIR, se debe especificar el ID de editor original en el archivo descriptor de la aplicación. De lo contrario, la versión instalada de la aplicación y la versión de actualización se tratan como aplicaciones diferentes. Si se utiliza un ID distinto o se omite la etiqueta `publisherID`, un usuario debe desinstalar la versión anterior antes de instalar la nueva.

Para determinar el ID de editor original, localice el archivo `publisherid` en el subdirectorio META-INF/AIR donde se instaló la aplicación original. La cadena de este archivo es el ID de editor. El descriptor de la aplicación debe especificar el motor de ejecución de AIR 1.5.3 (o posterior) en la declaración del espacio de nombres del archivo descriptor de la aplicación con el fin de especificar el ID de editor manualmente.

Para las aplicaciones publicadas antes de AIR 1.5.3, o que se publican con el SDK de AIR 1.5.3, al especificar una versión anterior de AIR en el espacio de nombres del descriptor de la aplicación, se calcula un ID de editor en función del certificado de firma. Este ID se utiliza, junto con el ID de la aplicación, para determinar la identidad de una aplicación. El ID de editor, cuando se encuentra presente, se emplea para lo siguiente:

- Comprobar que un archivo de AIR es una actualización en lugar de una nueva aplicación para instalar.
- Como parte de la clave de cifrado para el almacén local cifrado.
- Como parte de la ruta para el directorio de almacenamiento de la aplicación.
- Como parte de la cadena de conexión para conexiones locales.
- Como parte de la cadena de identidad utilizada para invocar una aplicación la API en navegador de AIR.
- Como parte de OSID (utilizado al crear programas personalizados de instalación y desinstalación).

Antes de AIR 1.5.3, el ID de editor de una aplicación podía cambiar si se firmaba una actualización de la aplicación con una firma de migración utilizando un certificado nuevo o renovado. Cuando cambia un ID de editor, el comportamiento de cualquier función de AIR basada en el ID también cambia. Por ejemplo, ya no se puede acceder a los datos del almacén local cifrado existente y cualquier instancia de Flash o AIR que cree una conexión local con la aplicación debe utilizar el nuevo ID en la cadena de conexión.

En AIR 1.5.3 o posterior, el ID de editor no se basa en el certificado de firma y solo se asigna si la etiqueta `publisherID` se incluye en el descriptor de la aplicación. Una aplicación no puede actualizarse si el ID de editor especificado para el paquete de AIR de actualización no coincide con su ID de editor actual.

Empaquetado con ADT

La herramienta de la línea de comandos ADT de AIR se pueden emplear para empaquetar una aplicación de AIR. Antes del empaquetado, todo el código ActionScript, MXML y cualquier código de extensión se debe compilar. También se debe disponer de un certificado de firma de código.

Para obtener una referencia detallada sobre las opciones y los comandos de ADT, consulte “[AIR Developer Tool \(ADT\)](#)” en la página 174.

Creación de un paquete de AIR

Para crear un paquete de AIR, utilice el comando `package` de ADT, estableciendo el tipo de destino en `air` para las versiones oficiales.

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

En el ejemplo se da por sentado que la ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319 para obtener ayuda.)

Se debe ejecutar el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son `myApp-app.xml` (archivo descriptor de la aplicación), `myApp.swf` y un directorio de iconos.

Cuando se ejecuta el comando tal y como se muestra, ADT solicitará la contraseña del almacén de claves. (Los caracteres de la contraseña que se escriben no siempre se muestran; simplemente presione Intro cuando termine de introducirlos.)

Creación de un paquete de AIR desde un archivo de AIRI

Es posible crear un archivo de AIRI para crear un paquete de AIR que se pueda instalar:

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

Empaquetado de un instalador nativo de escritorio

Desde AIR 2, ADT puede utilizarse para crear instaladores de la aplicación nativos para distribuir aplicaciones de AIR. Por ejemplo, es posible crear un archivo de instalación EXE para la distribución de una aplicación de AIR en Windows. Se puede crear un archivo de instalación DMG para la distribución de una aplicación de AIR en Mac OS. En AIR 2.5 y AIR 2.6, es posible crear un archivo de instalación DEB o RPM para la distribución de una aplicación de AIR en Linux.

Las aplicaciones instaladas con un instalador de aplicación nativo se conocen como aplicaciones de perfil de escritorio ampliadas. No es posible utilizar ADT para empaquetar un instalador nativo para una aplicación de AIR si el archivo descriptor de la aplicación no admite el perfil ampliado de escritorio. Este perfil se puede restringir utilizando el elemento `supportedProfiles` en el archivo descriptor de la aplicación. Consulte “[Perfiles de dispositivo](#)” en la página 256 y “[supportedProfiles](#)” en la página 250.

Se puede crear una versión del instalador nativo de la aplicación de AIR de dos formas básicas:

- Se puede crear el instalador nativo basado en el archivo descriptor de la aplicación y otros archivos de origen: (Otros archivos de origen pueden incluir archivos SWF, HTML y otros recursos.)
- El instalador nativo se puede crear en función de un archivo de AIR o un archivo AIRI.

ADT se debe utilizar en el mismo sistema operativo que el del archivo de instalación nativo que se desea generar. Por lo tanto, para crear un archivo EXE para Windows, ejecute ADT en Windows. Para crear un archivo DMG para Mac OS, ejecute ADT en Mac OS. Para crear un archivo DEB o RPM para Linux, ejecute ADT en el SDK de AIR 2.6 en Linux.

Cuando se crea un instalador nativo para distribuir una aplicación de AIR, la aplicación gana estas capacidades:

- Puede iniciar e interactuar con los procesos nativos, utilizando la clase `NativeProcess`. Para obtener más información, consulte una de las siguientes referencias:
 - [Communicating with native processes in AIR](#) (Comunicación con procesos nativos en AIR; en inglés.) (Para desarrolladores de HTML)
 - [Communicating with native processes in AIR](#) (Comunicación con procesos nativos en AIR; en inglés.) (Para desarrolladores de HTML)
- Puede utilizar extensiones nativas.
- Puede utilizar el método `File.openWithDefaultApplication()` para abrir cualquier archivo con la aplicación del sistema predeterminada definida para abrirlo, independientemente de su tipo de archivo. (Existen limitaciones en las aplicaciones que *no* están instaladas con un instalador nativo. Para obtener más información, consulte la entrada para `File.openWithDefaultApplication()` en la referencia del lenguaje.)

Sin embargo, cuando el empaquetado se realiza como instalador nativo, la aplicación pierde algunas de las ventajas del formato de archivo de AIR. Un solo archivo ya no se puede distribuir en todos los equipos de escritorio. La función de actualización incorporada (así como el marco actualizador) no funciona.

Si el usuario hace doble clic en el archivo de instalación nativo, se instalará la aplicación de AIR. Si la versión necesaria de Adobe AIR aún no está instalada en el equipo, el instalador la descarga de la red y la instala en primer lugar. Si no hay conexión de red con la que obtener la versión correcta de Adobe AIR (si es necesaria), se produce un error de instalación. Asimismo, la instalación falla si el sistema operativo no se admite en Adobe AIR 2.

Nota: *si desea que un archivo sea ejecutable en su aplicación instalada, asegúrese de que lo es en el sistema de archivos en el momento de empaquetar la aplicación. (En Mac y Linux, puede utilizar `chmod` para establecer el indicador de ejecutable, si es necesario.)*

Creación de un instalador nativo a partir de los archivos de origen de la aplicación

Para crear un instalador nativo a partir de los archivos de origen para la aplicación, utilice el comando `-package` con la siguiente sintaxis (en una sola línea de comandos):

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

Esta sintaxis es similar a la sintaxis para empaquetar un archivo de AIR (sin un instalador nativo). Sin embargo, existen algunas diferencias:

- La opción `-target native` se añade al comando. (Si se especifica `-target air`, ADT genera un archivo de AIR en lugar de un archivo de instalación nativo.)
- El archivo DMG o EXE de destino se especifica como `installer_file`.
- De forma opcional, en Windows es posible añadir un segundo conjunto de opciones de firma, indicado como `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` en el listado de sintaxis. En Windows, además de firmar un archivo de AIR, se puede firmar el archivo de Windows Installer. Utilice el mismo tipo de certificado y sintaxis de opción de firma que se usaría para firmar el archivo de AIR (consulte “[Opciones de firma de código de ADT](#)” en la página 189). Se puede emplear el mismo certificado para firmar el archivo de AIR y el archivo de instalación, o bien, se pueden especificar certificados diferentes. Cuando un usuario descarga un archivo firmado de Windows Installer de la web, Windows identifica el origen del archivo, en función del certificado.

Para obtener más información sobre las opciones de ADT distintas de `-target`, consulte “[AIR Developer Tool \(ADT\)](#)” en la página 174.

En el siguiente ejemplo se crea un archivo DMG (un archivo de instalación nativo para Mac OS):

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

En el siguiente ejemplo se crea un archivo EXE (un archivo de instalación nativo para Windows):

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

En el siguiente ejemplo se crea un archivo EXE y se firma:

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

Creación de un archivo de instalación nativo para una aplicación que utiliza extensiones nativas

Puede crear un archivo de instalación nativo desde los archivos de origen para la aplicación y los paquetes de extensiones nativas que requiera la aplicación. Utilice el comando `-package` con la siguiente sintaxis (en una sola línea de comandos):

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

Esta sintaxis es igual que la utilizada para empaquetar un archivo de instalación nativo, con dos opciones adicionales. Utilice la opción `-extdir extension-directory` para especificar el directorio que contiene los archivos ANE (extensiones nativas) que utiliza la aplicación. Utilice el indicador `-migrate` opcional y parámetros `MIGRATION_SIGNING_OPTIONS` para firmar una actualización de una aplicación con una firma de migración, en casos en que el certificado de firma de código principal sea distinto del usado con la versión anterior. Para obtener más información, consulte [“Firma de una versión actualizada de una aplicación de AIR”](#) en la página 209.

Para obtener información sobre las opciones de ADT, consulte [“AIR Developer Tool \(ADT\)”](#) en la página 174.

El siguiente ejemplo crea un archivo DMG (un archivo de instalación nativo para Mac OS) para una aplicación que utiliza extensiones nativas:

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

Creación de un instalador nativo a partir de un archivo de AIR o un archivo de AIRI

ADT se puede utilizar para generar un archivo de instalación nativo basado en un archivo de AIR o AIRI. Para crear un instalador nativo basado en un archivo de AIR, utilice el comando `-package` de ADT con la siguiente sintaxis (en una sola línea de comandos):

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

Esta sintaxis es similar a la que se utiliza para crear un instalador nativo basado en los archivos de origen para la aplicación de AIR. Sin embargo, existen algunas diferencias:

- Como origen, se especifica un archivo de AIR, en lugar de un archivo descriptor de la aplicación y otros archivos de origen para la aplicación de AIR.
- No especifique opciones de firma para el archivo de AIR, ya que ya está firmado.

Para crear un instalador nativo basado en un archivo de AIRI, utilice el comando `-package` de ADT con la siguiente sintaxis (en una sola línea de comandos):

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

Esta sintaxis es similar a la que se emplea para crear un instalador nativo basado en un archivo de AIR. Sin embargo, existen unas cuantas diferencias:

- Como origen, se especifica un archivo de AIRI.
- Se especifican opciones de firma para la aplicación de AIR de destino.

En el siguiente ejemplo se crea un archivo DMG (un archivo de instalación nativo para Mac OS) basado en un archivo de AIR:

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

En el siguiente ejemplo se crea un archivo EXE (un archivo de instalación nativo para Windows) basado en un archivo de AIR:

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

En el siguiente ejemplo se crea un archivo EXE (basado en un archivo de AIR) y se firma:

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

En el siguiente ejemplo se crea un archivo DMG (un archivo de instalación nativo para Mac OS) basado en un archivo de AIRI:

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

En el siguiente ejemplo se crea un archivo EXE (un archivo de instalación nativo para Windows) basado en un archivo de AIRI:

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

En el siguiente ejemplo se crea un archivo EXE (basado en un archivo de AIRI) y se firma con una firma nativa de Windows y de AIR:

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

Empaquetado de un paquete de motor de ejecución captador para equipos de escritorio

Un paquete de motor de ejecución captador es un paquete que incluye el código de la aplicación junto con una versión dedicada del motor de ejecución. Una aplicación empaquetada de esta forma utiliza el motor de ejecución del paquete, no el motor de ejecución compartido que esté instalado en el ordenador del usuario.

El paquete producido es una carpeta con archivos de aplicación en Windows y un paquete .app en Mac OS. Debe generar el paquete para un sistema operativo de destino mientras esté en dicho sistema operativo. (Se puede utilizar una máquina virtual, como VMWare, para ejecutar varios sistemas operativos en un solo ordenador.)

La aplicación se puede ejecutar desde dicha carpeta o paquete sin necesidad de instalación.

Ventajas

- Produce una aplicación ya llena
- No se requiere acceso a Internet para la instalación
- La aplicación está aislada de actualizaciones del motor de ejecución
- Las empresas pueden certificar la combinación específica de aplicación y motor de ejecución
- Admite el modelo tradicional de desarrollo de software
- No se requiere redistribución independiente de motor de ejecución
- Puede utilizar la API NativeProcess
- Puede utilizar extensiones nativas
- Puede usar la función `File.openWithDefaultApplication()` sin restricciones
- Puede ejecutarse desde un USB o disco óptico sin necesidad de instalación

Desventajas

- Las soluciones críticas de seguridad no están disponibles para los usuarios de forma automática cuando Adobe publica parches de seguridad.
- No se puede usar el formato de archivo `.air`
- Debe crear su propio archivo de instalación, si es necesario
- No se admite la API de actualización de AIR ni el marco
- No se admite la API de navegador de AIR para instalar e iniciar una aplicación de AIR desde una página web
- En Windows, el registro debe asumirlo el archivo de instalación
- Mayor huella de la aplicación en el disco

Creación de un paquete de motor de ejecución captador en Windows

Para crear un paquete de motor de ejecución captador para Windows, debe empaquetar la aplicación mientras esté en el sistema operativo Windows. Empaquete la aplicación con el destino ADT *bundle*:

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

Este comando crea el paquete en un directorio llamado `myApp`. El directorio contiene los archivos para la aplicación, así como los archivos del motor de ejecución. Puede ejecutar el programa directamente desde la carpeta. No obstante, para crear una entrada de menú para el programa, registrar tipos de archivo o controladores de esquemas URI, debe crear un programa de instalación que establezca las entradas del registro de requisitos. El SDK de AIR no incluye herramientas para crear estos archivos de instalación, pero muchos otros fabricantes sí ofrecen (gratis y previo pago) kits de herramientas de archivos de instalación con código abierto.

Puede firmar el código ejecutable nativo en Windows especificando un segundo conjunto de opciones de firma tras la entrada `-target bundle` en la línea de comandos. Estas opciones de firma identifican la clave privada y el certificado asociado que deben utilizarse para aplicar la firma nativa de Windows. (Se suele utilizar un certificado de firma de código de AIR.) Solo se firma el código ejecutable principal. El resto de ejecutables se empaquetan con la aplicación y no se firman en este proceso.

Asociación de tipos de archivo

Para asociar la aplicación a tipos de archivo públicos o personalizados en Windows, el programa de instalación debe ajustar las entradas correspondientes del registro. Los tipos de archivo se enumeran también en el elemento fileTypes del archivo descriptor de la aplicación.

Para obtener más información sobre los tipos de archivo de Windows, consulte [MSDN Library: File Types and File Associations](#) (en inglés)

Registro del controlador de URI

Para que la aplicación pueda controlar la apertura de una dirección URL con un esquema URI dado, el archivo de instalación debe definir las entradas del registro de requisitos.

Para obtener más información sobre el registro de una aplicación para controlar un esquema URI, consulte [MSDN Library: Registering an Application to a URL Protocol](#) (en inglés)

Creación de un paquete de motor de ejecución captador en Mac OS X

Para crear un paquete de motor de ejecución captador para Mac OS X, debe empaquetar la aplicación mientras esté en el sistema operativo Mac OS X. Empaquete la aplicación con el destino ADT *bundle*:

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

Este comando crea el paquete de la aplicación llamado myApp.app. El paquete contiene los archivos para la aplicación, así como los archivos del motor de ejecución. Puede ejecutar la aplicación haciendo doble clic en el icono myApp.app e instalarla arrastrando a una ubicación adecuada, como la carpeta Aplicaciones. Sin embargo, para registrar tipos de archivo o controladores de esquemas URI, debe editar el archivo de la lista de propiedades dentro del paquete de la aplicación.

Para distribuir el archivo, puede crear un archivo de imagen de disco (.dmg). El SDK de Adobe AIR no proporciona herramientas para crear un archivo dmg para un paquete de motor de ejecución captador.

Asociación de tipos de archivo

Para asociar la aplicación a tipos de archivo personalizados o públicos en Mac OS X, debe editar el archivo info.plist del paquete para definir la propiedad CFBundleDocumentTypes. Consulte [Mac OS X Developer Library: Information Property List Key Reference, CFBundleURLTypes](#) (en inglés).

Registro del controlador de URI

Para que la aplicación pueda controlar la apertura de una dirección URL con un esquema URI determinado, debe editar el archivo info.plist del paquete para definir la propiedad CFBundleURLTypes. Consulte [Mac OS X Developer Library: Information Property List Key Reference, CFBundleDocumentTypes](#) (en inglés).

Distribución de paquetes de AIR para equipos de escritorio

Las aplicaciones de AIR se pueden distribuir como paquete de AIR, que contiene el código de la aplicación y todos los recursos. Este paquete puede distribuirse por cualquiera de las vías tradicionales, por ejemplo descargándolo, por correo electrónico o en soportes físicos como CD-ROM. El usuario pueden instalar la aplicación haciendo doble clic en el archivo de AIR. La API en navegador de AIR (una biblioteca de ActionScript basada en la web) permite a los usuarios instalar la aplicación de AIR (y Adobe® AIR®, si es necesario) haciendo clic en un solo vínculo de una página web.

Las aplicaciones de AIR también se pueden empaquetar y distribuir como instaladores nativos (es decir, como archivos EXE en Windows, archivos DMG en Mac y archivos DEB o RPM en Linux). Los paquetes de instalación nativos se pueden distribuir e instalar según las convenciones de la plataforma relevantes. Cuando la aplicación se distribuye como paquete nativo, se pierden algunas ventajas del formato de archivo de AIR. Concretamente, un solo archivo de instalación ya no se puede utilizar en todas las plataformas, el marco de actualización de AIR ya no se puede usar y la API en navegador no se podrá volver a emplear.

Instalación y ejecución de una aplicación de AIR en el escritorio

Se puede simplemente enviar el archivo de AIR al destinatario. Por ejemplo: podría enviar el archivo de AIR como adjunto a un correo electrónico o como vínculo en una página web.

Una vez que descargó la aplicación de AIR, el usuario sigue estas instrucciones para instalarla:

- 1 Haga doble clic en el archivo de AIR.

Adobe AIR ya debe estar instalado en el ordenador.

- 2 En la ventana Instalación, deje seleccionada la configuración predeterminada y haga clic en Continuar.

En Windows, AIR realiza automáticamente lo siguiente:

- Instala la aplicación en el directorio Archivos de programa.
- Crea un acceso directo para la aplicación en el escritorio
- Crea un acceso directo en el menú Inicio
- Añade una entrada para la aplicación en Agregar o quitar programas, en el Panel de control

En Mac OS, la aplicación se añade de forma predeterminada al directorio Aplicaciones.

Si la aplicación ya está instalada, el instalador ofrece al usuario la opción de abrir la versión existente de la misma o actualizarla a la versión del archivo de AIR descargado. El instalador identifica la aplicación utilizando para ello el ID de la aplicación y el ID del editor que figuran en el archivo de AIR.

- 3 Una vez concluida la instalación, haga clic en Finalizar.

En Mac OS, para instalar una versión actualizada de una aplicación el usuario debe contar con privilegios del sistema adecuados para instalar programas en el directorio de aplicaciones. En Windows y Linux, el usuario debe contar con privilegios de administrador.

Una aplicación también puede instalar una nueva versión mediante ActionScript o JavaScript. Para obtener más información, consulte [“Actualización de aplicaciones de AIR”](#) en la página 270.

Una vez instalada la aplicación de AIR, basta con que el usuario haga doble clic en la aplicación para ejecutarla, al igual que con cualquier otra aplicación del escritorio.

- En Windows, haga doble clic en el icono de la aplicación (que está instalada en el escritorio o en una carpeta) o seleccione la aplicación en el menú Inicio.

- En Linux, haga doble clic en el icono de la aplicación (que está instalada en el escritorio o en una carpeta) o seleccione la aplicación en el menú de aplicaciones.
- En Mac OS, haga doble clic en la aplicación en la carpeta en la que se instaló. El directorio de instalación predeterminado es /Aplicaciones.

Nota: únicamente las aplicaciones de AIR desarrolladas para AIR 2.6 o anterior se pueden instalar en Linux.

La función de *instalación integrada* permite al usuario instalar una aplicación de AIR haciendo clic en un vínculo en una página web. La función de *invocación desde el navegador* permite al usuario ejecutar una aplicación de AIR instalada haciendo clic en un vínculo en una página web. Estas funciones se describen en la siguiente sección.

Instalación y ejecución de aplicaciones de AIR de escritorio desde una página web

La API en navegador de AIR permite instalar y ejecutar las aplicaciones de AIR desde una página web. La API en navegador de AIR se proporciona en una biblioteca SWF, *air.swf*, alojada por Adobe. El SDK de AIR incluye una aplicación de “identificación” (“badge”) de ejemplo que utiliza esta biblioteca para instalar, actualizar o iniciar una aplicación de AIR (y el motor de ejecución, si es necesario). Es posible modificar el identificador de ejemplo proporcionado o crear una aplicación web de identificación propia que utilice la biblioteca *air.swf* en línea directamente.

Cualquier aplicación de AIR se puede instalar mediante un identificador de página web. Sin embargo, solo las aplicaciones que incluyen el elemento `<allowBrowserInvocation>true</allowBrowserInvocation>` en sus archivos descriptores de aplicaciones se pueden iniciar mediante un identificador web.

Más temas de ayuda

“[API en navegador AIR.SWF](#)” en la página 261

Implementación en la empresa en equipos de escritorio

Los administradores de TI pueden instalar el motor de ejecución de Adobe AIR y aplicaciones de AIR de forma silenciosa con herramientas de implementación estándar. Los administradores de TI pueden llevar a cabo las siguientes tareas:

- Realizar una instalación silenciosa del motor de ejecución de Adobe AIR empleando herramientas como Microsoft SMS, IBM Tivoli o cualquier herramienta de implementación que permita las instalaciones silenciosas que utilizan un arrancador.
- Efectuar una instalación silenciosa de la aplicación de AIR con las mismas herramientas que se utilizan para implementar el motor de ejecución.

Para obtener más información, consulte la [Guía del administrador de Adobe AIR](#) (http://www.adobe.com/go/learn_air_admin_guide_es).

Registros de instalación en equipos de escritorio

Los registros de instalación se crean cuando se instala el propio motor de ejecución de AIR o una aplicación de AIR. Los archivos de registro se pueden examinar para ayudar a determinar la causa de cualquier problema de actualización o instalación que se produzca.

Los archivos de registro se crean en las siguientes ubicaciones:

- Mac: registro del sistema estándar(/private/var/log/system.log)

El registro del sistema Mac se puede ver abriendo la aplicación Console (se suele ubicar en la carpeta de utilidades).

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

Nota: estos archivos de registro no se crearon en versiones de AIR anteriores a AIR 2.

Capítulo 7: Desarrollo de aplicaciones de AIR para dispositivos móviles

Las aplicaciones de AIR en los dispositivos móviles se implementan como aplicaciones nativas. Utilizan el formato de la aplicación del dispositivo y no el formato de archivo de AIR. Actualmente AIR admite paquetes APK de Android y paquetes IPA de iOS. Una vez creada la versión oficial del paquete de la aplicación, esta puede distribuirse mediante el mecanismo de plataforma estándar. Para Android, esto suele significar Android Market; para iOS, App Store de Apple.

Se puede emplear el [SDK de AIR](#) y Flash Professional, Flash Builder y otras herramientas de desarrollo de ActionScript para crear aplicaciones de AIR para dispositivos móviles. Las aplicaciones de AIR móviles basadas en HTML no se admiten en este momento.

***Nota:** Research In Motion (RIM) BlackBerry Playbook proporciona su propio SDK para el desarrollo de AIR. Para obtener más información sobre el desarrollo de Playbook, consulte [RIM: BlackBerry Tablet OS Development](#) (RIM: Desarrollo de BlackBerry Tablet OS; en inglés).*

***Nota:** este documento describe cómo desarrollar aplicaciones de iOS utilizando el SDK de AIR 2.6 o versiones posteriores. Las aplicaciones creadas con AIR 2.6+ se pueden instalar en dispositivos iPhone 3Gs, iPhone 4 e iPad con iOS 4 o versión posterior. Para desarrollar aplicaciones de AIR para versiones anteriores de iOS, se debe usar AIR 3 Packager for iPhone tal y como se describe en [Creación de aplicaciones de iPhone](#).*

Para obtener más información sobre las prácticas de privacidad recomendadas, consulte la [guía de privacidad de SDK de Adobe AIR](#).

Para conocer los requisitos completos del sistema para ejecutar aplicaciones de AIR, consulte [Requisitos del sistema de Adobe AIR](#).

Configuración del entorno de desarrollo

Las plataformas móviles tienen requisitos de configuración adicionales además de la configuración del entorno de desarrollo normal de AIR, Flex y Flash. (Para obtener más información sobre la configuración del entorno de desarrollo básico de AIR, consulte “[Herramientas de la plataforma de Adobe Flash para el desarrollo de AIR](#)” en la página 18.)

Configuración de Android

Por norma general, no se necesita ninguna configuración especial en Android con AIR 2.6+. La herramienta ADB de Android se incluye en el SDK de AIR (en la carpeta lib/android/bin). El SDK de AIR utiliza la herramienta ADB para instalar, desinstalar y ejecutar paquetes de la aplicación en el dispositivo. ADB también se puede utilizar para ver registros del sistema. Para crear y ejecutar un emulador de Android, se debe descargar el SDK de Android independiente.

Si la aplicación añade elementos al elemento `<manifestAdditions>` del descriptor de la aplicación que la versión actual de AIR no reconoce como válido, deberá instalar una versión más reciente del SDK de Android. Defina la variable de entorno `AIR_ANDROID_SDK_HOME` o el parámetros de línea de comandos `-platformsdk` como la ruta de archivo del SDK. La herramienta de empaquetado de AIR, ADT, utiliza este SDK para validar las entradas en el elemento `<manifestAdditions>`.

En AIR 2.5, es necesario descargar una copia independiente del SDK de Android desde Google. Se puede establecer la variable del entorno AIR_ANDROID_SDK_HOME para hacer referencia a la carpeta del SDK de Android. Si no se establece esta variable de entorno, se debe especificar la ruta al SDK de Android en el argumento `-platformsdk` en la línea de comandos de ADT.

Más temas de ayuda

[“Variables del entorno de ADT”](#) en la página 198

[“Variables del entorno de ruta”](#) en la página 319

Configuración de iOS

Para instalar y probar una aplicación de iOS en un dispositivo y distribuir dicha aplicación, se debe incorporar al programa iOS Developer de Apple (servicio gratuito). Una vez que se una al programa iOS Developer, puede acceder a iOS Provisioning Portal donde podrá obtener los siguientes elementos y archivos de Apple que son necesarios para instalar una aplicación en un dispositivo para prueba y para la distribución posterior. Entre estos elementos y archivos se incluyen:

- Certificados de distribución y desarrollo
- ID de la aplicación
- Archivos de suministro de distribución y desarrollo

Consideraciones sobre el diseño de aplicaciones móviles

El contexto de funcionamiento y las características físicas de los dispositivos móviles requieren un diseño y una codificación cuidadosos. Por ejemplo, la simplificación del código para agilizar su ejecución resulta fundamental. Obviamente la optimización del código solo puede llegar hasta un punto; el diseño inteligente que funciona en las limitaciones del dispositivo también puede ayudar a evitar que la presentación visual sobrecargue el sistema de procesamiento.

Código

Aunque la agilización de la ejecución del código siempre resulta beneficiosa, la velocidad más baja del procesador de la mayoría de los dispositivos móviles aumenta las recompensas del tiempo empleado al escribir código de apoyo. Asimismo, los dispositivos móviles siempre se ejecutan con batería. En la obtención del mismo resultado con menos trabajo se utiliza menos batería.

Diseño

Factores como el tamaño pequeño de la pantalla, el modo de interacción con la pantalla táctil e incluso el entorno de cambio constante de un usuario móvil se deben tener en cuenta al diseñar la experiencia de usuario de la aplicación.

Código y diseño de forma conjunta

Si la aplicación utiliza animación, la optimización de la representación es muy importante. Sin embargo, la optimización de código por sí misma a veces es insuficiente. Se deben diseñar los aspectos visuales de la aplicación para que el código los pueda representar con eficacia.

Las técnicas de optimización importantes se analizan en la guía [Optimización del rendimiento para la plataforma de Flash](#) de Adobe. Las técnicas incluidas en esta guía se aplican a todo el contenido de Flash y AIR, pero son esenciales para desarrollar aplicaciones que se ejecutan bien en los dispositivos móviles.

- [Paul Trani: Tips and Tricks for Mobile Flash Development](#) (Sugerencias y trucos para desarrollo de Flash en dispositivos móviles, en inglés).
- [roguish: GPU Test App AIR for Mobile](#)
- [Jonathan Campos: Optimization Techniques for AIR for Android apps](#) (Técnicas de optimización de para las aplicaciones de AIR para Android; en inglés)
- [Charles Schulze: AIR 2.6 Game Development: iOS included](#)

Ciclo de vida de la aplicación

Si la aplicación deja de seleccionar otra aplicación, AIR elimina la velocidad de fotogramas en 4 fotogramas por segundo y deja de representar los gráficos. En valores inferiores a esta velocidad de fotogramas, la red de transmisión y las conexiones de socket tienden a bloquearse. Si la aplicación no utiliza estas conexiones, la velocidad de fotogramas se puede disminuir aún más.

Si es adecuado, se debe detener la reproducción de audio y eliminar los detectores en los sensores del acelerómetro y localización geográfica. El objeto `NativeApplication` de AIR distribuye eventos de activación y desactivación. Utilice estos eventos para administrar la transición entre el estado activo y de fondo.

Las mayor parte de los sistemas operativos móviles finalizan las aplicaciones de fondo sin aviso. Al guardar el estado de la aplicación con frecuencia, la aplicación debe poder restaurarse a sí misma a un estado razonable si vuelve a su estado activo desde el segundo plano o se inicia de nuevo.

Densidad de la información

El tamaño físico de la pantalla de los dispositivos móviles es inferior al de los equipos de escritorio, aunque su densidad de píxeles (píxeles por pulgada=) es mayor. El mismo tamaño de fuente producirá letras que son físicamente más pequeñas en la pantalla del dispositivo móvil que en un equipo de escritorio. A menudo se debe emplear una fuente más grande para garantizar la legibilidad. En general, punto 14 es el tamaño de fuente más pequeño que se puede leer fácilmente.

La mayoría de los dispositivos móviles se utilizan en el momento y con malas condiciones de iluminación. Analice cuánta información puede mostrar en pantalla de forma realista y con legibilidad. Suele ser menor que la que visualizaría en una pantalla con las mismas dimensiones de píxeles en un equipo de escritorio.

También se debe tener en cuenta que cuando un usuario está tocando la pantalla, los dedos y la mano bloquean parte de la vista de la pantalla. Sitúe los elementos interactivos en los lados y la parte inferior de la pantalla cuando el usuario ya no tenga que interactuar con los mismos durante más tiempo que un toque momentáneo.

Escritura de texto

Muchos dispositivos utilizan un teclado virtual para la introducción de texto. Los teclados virtuales oscurecen parte de la pantalla y su uso es a veces complejo. Evite confiar en los eventos de teclado (excepto para las teclas programables).

Considere alternativas de implementación a los campos de introducción de texto. Por ejemplo, para que el usuario introduzca un valor numérico no es necesario disponer de un campo de texto. Puede incluir dos botones para aumentar o disminuir el valor.

Teclas programables

Los dispositivos móviles incluyen un número variado de teclas programables. Las teclas programables son botones que se programan para tener diferentes funciones. Siga las convenciones de la plataforma para estas teclas en la aplicación.

Cambios de orientación de la pantalla

El contenido móvil se puede visualizar en orientación vertical y horizontal. Por ello, debe tener en cuenta qué hará la aplicación cuando cambie la orientación de la pantalla. Para obtener más información, consulte [Orientación del escenario](#).

Atenuación de pantalla

AIR no evita automáticamente que la pantalla se atenúe mientras se reproduce el vídeo. Se puede utilizar la propiedad `systemIdleMode` del objeto `NativeApplication` de AIR para controlar si el dispositivo entrará en modo de ahorro de energía. (En algunas plataformas, se deben solicitar los permisos adecuados para que funcione esta característica.)

Llamadas de teléfono entrantes

El motor de ejecución de AIR silencia el audio automáticamente cuando el usuario realiza o recibe una llamada de teléfono. En Android, se debe establecer el permiso `READ_PHONE_STATE` de Android en el descriptor de la aplicación si esta reproduce audio mientras esté en segundo plano. De lo contrario, Android evita que el motor de ejecución detecte llamadas de teléfono y silencia el audio automáticamente. Consulte “[Permisos de Android](#)” en la página 82.

Destinos de selección

Tenga en cuenta el tamaño de los destinos de selección cuando diseñe botones y otros elementos de la interfaz de usuario que este tocará durante el uso. Estos elementos deben ser lo suficientemente grandes como para activarlos cómodamente con un dedo en una pantalla táctil. Asimismo, asegúrese de que dispone de espacio suficiente entre los destinos. El área de destino debe ser de unos 44 píxeles a 57 píxeles en cada lado para una pantalla de dispositivo móvil típica de ppp elevados.

Tamaño de la instalación del paquete de la aplicación

Los dispositivos móviles suelen tener menos espacio de almacenamiento para instalar aplicaciones y datos que los equipos de escritorio. Minimice el tamaño del paquete eliminando los recursos y las bibliotecas no utilizados.

En Android, el paquete de la aplicación no se extrae en archivos independientes cuando se instala la aplicación. Los recursos se descomprimen en un almacenamiento temporal cuando se accede a los mismos. Para reducir esta superficie de almacenamiento del recurso descomprimido, cierre las transmisiones de URL y archivos cuando los recursos se hayan cargado por completo.

Acceso al sistema de archivos

Los diferentes sistemas operativos móviles imponen distintas restricciones al acceso al sistema de archivos y estas limitaciones tienden a ser distintas de las aplicadas por los sistemas operativos de escritorio. Por lo tanto, el lugar adecuado para guardar los archivos y los datos puede variar según la plataforma.

Una consecuencia de la variación en los sistemas de archivos es que los métodos abreviados a los directorios comunes proporcionados por la clase `File` de AIR no siempre están disponibles. La siguiente tabla muestra qué métodos abreviados se pueden usar en Android y iOS:

	Android	iOS
File.applicationDirectory	Solo lectura mediante URL (sin ruta nativa)	Solo lectura
File.applicationStorageDirectory	Disponible	Disponible
File.cacheDirectory	Disponible	Disponible
File.desktopDirectory	Raíz de sdcard	No disponible
File.documentsDirectory	Raíz de sdcard	Disponible
File.userDirectory	Raíz de sdcard	No disponible
File.createTempDirectory()	Disponible	Disponible
File.createTempFile()	Disponible	Disponible

Las directrices de Apple para aplicaciones de iOS estipulan normas específicas para determinar las ubicaciones de almacenamiento de archivos según el caso. Por ejemplo: un directorio designado como ubicación remota para albergar copias de seguridad únicamente puede contener archivos con datos introducidos por usuarios o información que no sea posible volver a generar o cargar. Para obtener información sobre cómo cumplir con las directrices de Apple para copia de seguridad y almacenamiento de archivos en caché, consulte Control de copias de seguridad y almacenamiento en caché.

Componentes de interfaz

Adobe ha desarrollado una versión optimizada para móvil de la arquitectura de Flex. Para obtener más información, consulte [Developing Mobile Applications with Flex and Flash Builder](#).

Los proyectos del componente Community adecuados para las aplicaciones móviles también están disponibles. Se incluyen:

- [Feathers UI controls for Starling](#) de Josh Tynjala
- [Versión de aspecto configurable de componentes mínimos](#) de Derrick Grigg
- [Componentes as3flobile](#) de Todd Anderson

Procesamiento de gráficos acelerados Stage3D

Desde AIR 3.2, AIR para móviles admite procesamiento de gráficos acelerados Stage 3D. Las API [Stage3D](#) de ActionScript son un conjunto de API de bajo nivel con aceleración por GPU que permiten usar funciones 2D y 3D avanzadas. Estas API de bajo nivel ofrecen flexibilidad a los desarrolladores para poder aprovechar la aceleración por hardware de GPU y mejorar significativamente el rendimiento. También puede utilizar motores de juegos que admitan las API Stage3D de ActionScript.

Para obtener más información, consulte [Motores de juegos, 3D y Stage3D](#).

Suavizado de vídeo

Para mejorar el rendimiento, el suavizado de vídeo está desactivado en AIR.

Funciones nativas

AIR 3.0+

Muchas plataformas para móvil incluyen funciones a las que aún no se puede acceder desde la API estándar de AIR. En AIR 3, es posible ampliar AIR con sus propias bibliotecas nativas de código. Estas bibliotecas de extensiones nativas pueden acceder a funciones disponibles en el sistema operativo e incluso a funciones específicas de un dispositivo determinado. Las extensiones nativas se pueden escribir en C en iOS, y en Java o en C en Android. Para obtener información sobre el desarrollo de extensiones nativas, consulte Introducción a extensiones nativas para Adobe AIR.

Flujo de trabajo para crear aplicaciones de AIR para dispositivos móviles

El flujo de trabajo para crear una aplicación de AIR para dispositivos móviles (o de otro tipo) es, en general, muy similar al utilizado para la creación de una aplicación de escritorio. Las diferencias principales de flujo de trabajo se producen cuando llega el momento de empaquetar, depurar e instalar una aplicación. Por ejemplo, las aplicaciones de AIR para Android utilizan el formato del paquete nativo APK de Android en lugar del formato del paquete de AIR. Por lo tanto, también utilizan los mecanismos de actualización e instalación de Android estándar.

AIR para Android

Los siguientes pasos son típicos al desarrollar una aplicación de AIR para Android:

- Escriba código ActionScript o MXML.
- Cree un archivo descriptor de la aplicación de AIR (utilizando el espacio de nombres 2.5 o posterior).
- Compile la aplicación.
- Empaquete la aplicación como paquete de Android (.apk).
- Instale el motor de ejecución de AIR en el dispositivo o emulador de Android (si utiliza un motor externo; el motor captador es el predeterminado en AIR 3.7 y posterior).
- Instale la aplicación en el dispositivo (o emulador de Android).
- Inicie la aplicación en el dispositivo.

Puede utilizar Adobe Flash Builder, Adobe Flash Professional CS5 o las herramientas de línea de comandos para llevar a cabo estos pasos.

Una vez que la aplicación de AIR está finalizada y empaquetada como archivo APK, se puede enviar a Android Market o distribuirse con otros medios.

AIR para iOS

Los siguientes pasos son típicos al desarrollar una aplicación de AIR para iOS:

- Instale iTunes.
- Genere los ID y los archivos del desarrollador necesarios en Apple iOS Provisioning Portal. Estos elementos incluyen:
 - Certificado de Developer
 - ID de la aplicación

- Archivo de suministro

Se deben incluir los ID de cualquier dispositivo de prueba en los que se vayan a instalar la aplicación al crear el perfil de suministro.

- Convierta del certificado de desarrollo y la clave privada en un archivo de almacén de claves P12.
- Escriba código MXML o ActionScript de la aplicación.
- Compile la aplicación con un compilador de ActionScript o MXML.
- Cree gráficos de iconos y de la pantalla inicial de la aplicación.
- Cree el descriptor de la aplicación (utilizando el espacio de nombres 2.6 o posterior).
- Empaquete el archivo IPA utilizando ADT.
- Utilice iTunes para situar el perfil de suministro en el dispositivo de prueba.
- Instale y pruebe la aplicación en el dispositivo de iOS. Puede utilizar iTunes o ADT a través de USB (se admite USB en AIR 3.4 y posterior) para instalar el archivo IPA.

Una vez finalizada la aplicación de AIR, se puede volver a empaquetar utilizando un certificado de distribución y perfil de suministro. Después estará lista para enviarse a Apple App Store.

Configuración de las propiedades de una aplicación móvil

Tal y como sucede con otras aplicaciones de AIR, las propiedades básicas de la aplicación se establecen en el archivo descriptor de la aplicación. Las aplicaciones móviles omiten algunas de las propiedades específicas del escritorio como, por ejemplo, transparencia y tamaño de la ventana. Las aplicaciones móviles también pueden utilizar sus propias propiedades específicas de la plataforma. Por, ejemplo se puede incluir un elemento `android` para las aplicaciones de Android y un elemento `iPhone` para las aplicaciones de iOS.

Configuración común

Varias opciones de configuración del descriptor de la aplicación son importantes para todas las aplicaciones de dispositivo móvil.

Versión necesaria del motor de ejecución de AIR

Especifique la versión del motor de ejecución de AIR que necesita la aplicación utilizando el espacio de nombres del archivo descriptor de la aplicación.

El espacio de nombres, asignado en el elemento `application`, determina, en gran parte, qué funciones puede utilizar la aplicación. Por ejemplo, si la aplicación utiliza el espacio de nombres de AIR 2.7 y el usuario tiene alguna versión posterior instalada, la aplicación aún verá el comportamiento de AIR 2.7 (aunque el comportamiento haya cambiado en la versión posterior). Solo cuando se cambia el espacio de nombres y se publica una actualización, la aplicación tendrá acceso a las nuevas funciones y características. Las soluciones de seguridad son una excepción importante a esta regla.

En los dispositivos que utilizan un motor de ejecución independiente de la aplicación, como Android en AIR 3.6 y anterior, al usuario se le pedirá que instale o actualice AIR si no dispone de la versión necesaria. En los dispositivos que incorporan el motor de ejecución, como iPhone, no se produce esta situación (debido a que la versión necesaria se empaqueta con la aplicación en primer lugar).

Nota: (AIR 3.7 y posterior) de forma predeterminada, ADT empaqueta el tiempo de ejecución con las aplicaciones de Android.

Especifique el espacio de nombres utilizando el atributo `xmlns` del elemento raíz `application`: Los siguientes espacios de nombres se pueden utilizar para las aplicaciones móviles (dependiendo de la plataforma móvil):

```
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">
```

Nota: la compatibilidad con los dispositivos de iOS 3 se proporciona en el SDK de Packager for iPhone SDK, basada en el SDK de AIR 2.0. Para obtener información sobre la creación de aplicaciones de AIR para iOS 3, consulte [Creación de aplicaciones para iPhone](#). El SDK de AIR 2.6 (y versiones posteriores) admite iOS 4 y versiones más recientes en dispositivos iPhone 3Gs, iPhone 4 e iPad.

Más temas de ayuda

“[application](#)” en la página 219

Identidad de la aplicación

Las distintas configuraciones deben ser exclusivas para cada aplicación que se publique. Entre los valores de configuración se incluyen el ID, el nombre y el nombre del archivo.

ID de la aplicación Android

En Android, el ID se convierte en el nombre del paquete de Android, indicando como prefijo “air.” en el ID de AIR. De este modo, si el ID de la aplicación es `com.example.MyApp`, el nombre del paquete de Android será `air.com.example.MyApp`.

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

Asimismo, si el ID no es un nombre de paquete válido en el sistema operativo Android, se convierte a un nombre válido. Los guiones cambian a caracteres de subrayado y los primeros dígitos de cualquier componente de ID van precedidos de una letra “A” mayúscula. Por ejemplo, el ID: `3-goats.1-boat`, se transforma en el nombre de paquete: `air.A3_goats.A1_boat`.

Nota: el prefijo añadido al ID de la aplicación se puede utilizar para identificar las aplicaciones de AIR en Android Market. Si no desea que su aplicación se identifique como aplicación de AIR por el prefijo, debe desempaquetar el archivo APK, cambiar el ID de aplicación y volver a empaquetarlo tal y como se describe en [Opt-out of AIR application analytics for Android](#) (Cancelación voluntaria de la analítica de la aplicación de AIR para Android; en inglés).

ID de la aplicación iOS

Establezca el ID de la aplicación de AIR para que coincida con el ID de la aplicación creado en Apple iOS Provisioning Portal.

Los ID de las aplicaciones de iOS contienen un ID de raíz del paquete seguido de un identificador del paquete. El ID de raíz del paquete es una cadena de caracteres, por ejemplo `5RM86Z4DJM`, que Apple asigna al ID de aplicación. El identificador del paquete contiene un nombre de estilo de dominio inverso que puede escoger. El identificador del paquete puede terminar con un asterisco (*), lo que indica que se trata de un ID de aplicación comodín. Si el identificador del paquete termina con un carácter comodín, este se puede reemplazar con cualquier cadena válida.

Por ejemplo:

- Si el ID de la aplicación de Apple es 5RM86Z4DJM.com.example.helloWorld, se debe usar com.example.helloWorld en el descriptor de la aplicación.
- Si su ID de aplicación de Apple es 96LPVWEASL.com.example.* (ID de la aplicación comodín), se puede utilizar com.example.helloWorld, or com.example.anotherApp o algún otro ID que comience con com.example.
- Finalmente, si el ID de la aplicación de Apple es solo la raíz del paquete y un comodín como, por ejemplo: 38JE93KJL.*, se puede emplear cualquier ID de la aplicación en AIR.

Cuando especifique el ID de aplicación, omita la parte del ID de raíz del paquete en el ID de aplicación.

Más temas de ayuda

[“id”](#) en la página 236

[“filename”](#) en la página 231

[“name”](#) en la página 244

Versión de la aplicación

En AIR 2.5 y posterior, especifique la versión de la aplicación en el elemento `versionNumber`. El elemento `version` ya no puede volver a utilizarse. Cuando se especifica un valor para `versionNumber`, se debe utilizar una secuencia de hasta tres nombres separados por puntos; por ejemplo, “0.1.2”. Cada segmento del número de versión puede tener hasta tres dígitos. (Es decir, “999.999.999” es el mayor número de versión permitido.) No es necesario incluir los tres segmentos en el número; “1” y “1.0” son también números de la versión legal.

También se puede especificar una etiqueta para la versión utilizando el elemento `versionLabel`. Cuando se añade una etiqueta de versión, se muestra en lugar del número de versión en estos lugares como cuadros de diálogo del instalador de aplicaciones de Android. Se debe especificar una etiqueta de la versión para las aplicaciones que se distribuyen utilizando Android Market. Si no se especifica ningún valor `versionLabel` en el descriptor de la aplicación de AIR, el valor `versionNumber` se asigna al campo de etiqueta de la versión de Android.

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

En Android, `versionNumber` de AIR se traduce en el entero de Android `versionCode` con la fórmula: $a*1000000 + b*1000 + c$, siendo a, b y c componentes del número de versión de AIR: a.b.c.

Más temas de ayuda

[“version”](#) en la página 252

[“versionLabel”](#) en la página 253

[“versionNumber”](#) en la página 253

Archivo SWF de la aplicación principal

Especifique el archivo SWF de la aplicación principal en el elemento secundario `content` del elemento `initialWindow`. Cuando se centre en dispositivos en el perfil móvil, se debe utilizar un archivo SWF (las aplicaciones basadas en HTML no se admiten).

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

El archivo se debe incluir en el paquete de AIR (utilizando ADT o su IDE). Simplemente hacer referencia al nombre en el descriptor de la aplicación no hace que el archivo se incluya en el paquete automáticamente.

Propiedades de la pantalla principal

Diversos elementos secundarios del elemento `initialWindow` controlan el comportamiento y la apariencia inicial de la pantalla de la aplicación principal.

- **aspectRatio:** especifica si la aplicación debe visualizar inicialmente el contenido en formato vertical (*portrait*) (mayor altura que anchura), en formato horizontal (*landscape*) (menor altura que anchura) o en cualquier formato (*any*) (el escenario se orienta automáticamente en todas las orientaciones).

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

- **autoOrients:** especifica si el escenario debe cambiar automáticamente la orientación conforme el usuario gira el dispositivo o realiza otro gesto relacionado con la orientación como, por ejemplo, apertura o cierre de un teclado deslizante. Si el valor es *false* (predeterminado), el escenario no cambiará la orientación con el dispositivo.

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

- **depthAndStencil:** especifica el uso del búfer de estencil o de profundidad. Normalmente estos búferes se utilizan al trabajar con contenido en 3D.

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

- **fullScreen:** especifica si la aplicación debe abarcar toda la pantalla del dispositivo, o bien, compartir la pantalla con el fondo cromático del sistema operativo normal como, por ejemplo, una barra de estado del sistema.

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

- **renderMode:** especifica si el motor de ejecución debe procesar la aplicación con la unidad de procesamiento de gráficos (GPU) o la unidad de procesamiento central principal (CPU). En general, el procesamiento con GPU aumentará la velocidad del proceso, pero algunas funciones, como determinados modos de fusión y filtros de PixelBender, no están disponibles en este modo. Asimismo, diferentes dispositivos y controladores de dispositivo cuentan con limitaciones y capacidades de GPU que varían. Siempre se debe probar la aplicación en una amplia variedad de dispositivos, especialmente cuando se utilice el modo de GPU.

Puede establecer el modo de procesamiento como *gpu*, *cpu*, *direct* o *auto*. El valor predeterminado es *auto*, que actualmente vuelve al modo de *cpu*.

Nota: para poder aprovechar la aceleración de GPU del contenido de Flash con plataformas de AIR para móviles, Adobe recomienda utilizar `renderMode="direct"` (es decir, Stage3D) en vez de `renderMode="gpu"`. Adobe oficialmente admite y recomienda las siguientes arquitecturas basadas en Stage3D: Starling (2D) y Away3D (3D). Para obtener más información sobre Stage3D y Starling/Away3D, consulte <http://gaming.adobe.com/getstarted/>.

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

Nota: No se puede utilizar `renderMode="direct"` para aplicaciones ejecutadas en segundo plano.

Entre las limitaciones del modo GPU se encuentran:

- La arquitectura de Flex no admite el modo de procesamiento de GPU.
- No se admite el uso de filtros
- No se admiten los rellenos y fusiones de PixelBender.
- No se admiten los siguientes modos de mezcla: capa, alfa, borrado, luz fuerte, aclarar y oscurecer.
- No se recomienda el uso del modo de procesamiento con GPU en una aplicación que reproduce vídeo.

- En el modo de procesamiento con GPU, los campos de texto no se reubican adecuadamente en una posición visible cuando se abre el teclado virtual. Para garantizar que el campo de texto pueda verse mientras que el usuario introduce el texto, utilice la propiedad `softKeyboardRect` de los eventos de teclado de pantalla y escenario para mover el campo de texto al área visible.
- Si un objeto de visualización no se puede representar mediante la GPU, no se mostrará. Por ejemplo, si se aplica un filtro a un objeto de visualización, el objeto no se mostrará.

Nota: la implementación la GPU para iOS en AIR 2.6+ es muy distinta de la implementación que se usaba anteriormente en la versión AIR 2.0. Se aplican diferentes consideraciones de optimización.

Más temas de ayuda

[“aspectRatio”](#) en la página 223

[“autoOrients”](#) en la página 223

[“depthAndStencil”](#) en la página 227

[“fullScreen”](#) en la página 235

[“renderMode”](#) en la página 247

Perfiles admitidos

El elemento `supportedProfiles` se puede añadir para especificar qué perfiles del dispositivo admite la aplicación. Utilice el perfil `mobileDevice` para dispositivos móviles. Cuando se ejecuta una aplicación con Adobe Debug Launcher (ADL), ADL utiliza el primer perfil en la lista como perfil activo. También se puede emplear el indicador `-profile` al ejecutar ADL para seleccionar un perfil concreto en la lista admitida. Si la aplicación se ejecuta en todos los perfiles, el elemento `supportedProfiles` se puede excluir. ADL utiliza el perfil de escritorio como perfil activo predeterminado en este caso.

Para especificar que la aplicación admite los perfiles de escritorio y dispositivo móvil y que normalmente se desea probar la aplicación en el perfil móvil, añada el siguiente elemento:

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

Más temas de ayuda

[“supportedProfiles”](#) en la página 250

[“Perfiles de dispositivo”](#) en la página 256

[“AIR Debug Launcher \(ADL\)”](#) en la página 168

Extensiones nativas necesarias

Las aplicaciones que admiten el perfil `mobileDevice` pueden utilizar extensiones nativas.

Declare todas las extensiones nativas que la aplicación de AIR utiliza en el descriptor de la aplicación. El siguiente ejemplo ilustra la sintaxis para especificar dos extensiones nativas necesarias:

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

El elemento `extensionID` tiene el mismo valor que `id` en el archivo descriptor de la extensión. El archivo descriptor de la extensión es un archivo XML denominado `extension.xml`. Se empaqueta en el archivo ANE que se recibe del desarrollador de extensiones nativas.

Comportamiento del teclado virtual

Establezca el elemento `softKeyboardBehavior` en `none` para poder deshabilitar el comportamiento de cambio de tamaño y desplazamiento automáticos que utiliza el motor de ejecución para garantizar que el campo de introducción de texto seleccionado se puede ver cuando se activa el teclado virtual. Si se desactiva el comportamiento automático, la aplicación debe asegurar que el área de introducción de texto y otro contenido relevante esté visible una vez mostrado el teclado. Puede utilizar la propiedad `softKeyboardRect` del escenario junto con `SoftKeyboardEvent` para detectar el momento en que el teclado se abre y determinar el área que se oscurece.

Para activar el comportamiento automático, establezca el valor del elemento en `pan`:

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

Debido a que `pan` es el valor predeterminado, con la omisión del elemento `softKeyboardBehavior` también se activa el comportamiento del teclado automático.

Nota: cuando también se emplea la representación con GPU, el comportamiento de desplazamiento no se admite.

Más temas de ayuda

“[softKeyboardBehavior](#)” en la página 249

[Stage.softKeyboardRect](#)

[SoftKeyboardEvent](#)

Configuración de Android

En la plataforma Android, se puede usar el elemento `android` del descriptor de la aplicación para añadir información en el manifiesto de la aplicación de Android, que es un archivo de propiedades de la aplicación utilizado por el sistema operativo Android. ADT genera automáticamente el archivo `Manifest.xml` de Android cuando se crea el paquete APK. AIR establece unas cuantas propiedades en los valores necesarios para que funcionen determinadas funciones. Cualquier otra propiedad definida en la sección de Android del descriptor de la aplicación de AIR se añade a la sección correspondiente del archivo `Manifest.xml`.

Nota: en la mayoría de las aplicaciones de AIR, se deben establecer los permisos de Android necesarios por la aplicación en el elemento `android`, pero generalmente no es necesario establecer ninguna otra propiedad.

Solo se pueden establecer los atributos que adoptar valores booleanos, enteros o de cadena. La definición de referencias en el paquete de la aplicación no se admite.

Nota: El motor de ejecución requiere como mínimo la versión del SDK igual o mayor que 14. Si desea crear una aplicación solo para versiones más altas, asegúrese de que el elemento `manifest` incluya `<uses-sdk android:minSdkVersion=""></uses-sdk>` con la versión correcta.

Configuración reservada del manifiesto de Android

AIR establece varias entrada de manifiesto en el documento de manifiesto de Android generado para garantizar que las funciones del motor de ejecución y la aplicación funcionan correctamente. No se pueden definir las siguientes opciones:

Elemento `manifest`

Los siguientes atributos del elemento de manifiesto no se pueden establecer:

- `package`
- `android:versionCode`

- android:versionName
- xmlns:android

Elemento activity

Los siguientes atributos del elemento de actividad principal no se pueden establecer:

- android:label
- android:icon

Elemento application

Los siguientes atributos del elemento de la aplicación no se pueden establecer:

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

Permisos de Android

El modelo de seguridad de Android requiere que cada aplicación solicite permiso para poder utilizar las funciones que tienen implicaciones de seguridad o privacidad. Estos permisos se deben especificar cuando la aplicación se empaqueta y no se pueden modificar en tiempo de ejecución. El sistema operativo Android informa al usuario sobre qué permisos solicita la aplicación cuando el usuario la instala. Si un permiso necesario para una función no se solicita, puede que el sistema operativo Android genere una excepción cuando la aplicación acceda a la función, pero la excepción no está garantizada. Las excepciones se transmiten a la aplicación mediante el motor de ejecución. En el caso de un error silencioso, se añade un mensaje de error de permiso al registro del sistema de Android.

En AIR, los permisos de Android se especifican en el elemento `android` del descriptor de la aplicación. El siguiente formato se utiliza para añadir permisos (`PERMISSION_NAME` es el nombre de un permiso de Android):

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

Las sentencias `uses-permissions` del elemento `manifest` se añaden directamente al documento manifiesto de Android.

Los siguientes permisos son necesarios para utilizar diversas funciones de AIR:

ACCESS_COARSE_LOCATION Permite que la aplicación acceda a datos de ubicación de red WIFI y móvil a través de la clase `Geolocation`.

ACCESS_FINE_LOCATION Permite que la aplicación acceda a datos GPS a través de la clase `Geolocation`.

ACCESS_NETWORK_STATE y ACCESS_WIFI_STATE Permite que la aplicación acceda a la información de red a través de la clase NetworkInfo.

CAMERA Permite que la aplicación acceda a la cámara.

Nota: cuando se pide permiso para utilizar la función de cámara, Android entiende que la aplicación también requiere la cámara. Si la cámara es una función opcional de la aplicación, se debe añadir un elemento `uses-feature` al manifiesto para la cámara, establecido en atributo necesario en `false`. Consulte “Filtro de compatibilidad con Android” en la página 84.

INTERNET Permite que la aplicación realice solicitudes de red y permite la depuración remota.

READ_PHONE_STATE Permite que el motor de ejecución de AIR silencie el audio durante las llamadas de teléfono. Este permiso se debe establecer si la aplicación reproduce audio mientras está en segundo plano.

RECORD_AUDIO Permite que la aplicación acceda al micrófono.

WAKE_LOCK y DISABLE_KEYGUARD Permite que la aplicación impida que el dispositivo entre en reposo con la configuración de la clase SystemIdleMode.

WRITE_EXTERNAL_STORAGE Permite que la aplicación escriba en la tarjeta de memoria externa del dispositivo.

Por ejemplo, para establecer permisos para una aplicación que requiera todos los permisos, se puede añadir lo siguiente al descriptor de la aplicación:

```
<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>
```

Más temas de ayuda

[Seguridad y permisos de Android](#)

[Clase Manifest.permission de Android](#)

Esquemas de URI personalizados de Android

Puede utilizar un esquema de URI personalizado para iniciar una aplicación de AIR desde una página web o una aplicación nativa de Android. La compatibilidad con URI personalizado se basa en los filtros prácticos especificados en el manifiesto de Android, por lo que esta técnica no se puede utilizar en otras plataformas.

Para usar un URI personalizado, añada un filtro práctico al descriptor de la aplicación en el bloque `<android>`. Ambos elementos `intent-filter` del siguiente ejemplo se deben especificar. Edite la sentencia `<data android:scheme="my-customuri"/>` para reflejar la cadena URI para el esquema personalizado.

```
<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>
```

Un filtro práctico informa al sistema operativo Android de que la aplicación está disponible para realizar una operación concreta. En el caso de un URI personalizado, esto significa que el usuario ha hecho clic en un vínculo utilizando ese esquema de URI (y el navegador no sabe cómo gestionarlo).

Cuando la aplicación se invoca mediante un URI personalizado, el objeto `NativeApplication` distribuye un evento `invoke`. La URL del vínculo, incluyendo parámetros de consulta, se ubica en el conjunto `arguments` del objeto `InvokeEvent`. Se puede usar cualquier número de filtros prácticos.

Nota: los vínculos de una instancia de `StageWebView` no pueden abrir direcciones URL que utilicen un esquema URI personalizado.

Más temas de ayuda

[Filtros intent de Android](#)

[Categorías y acciones de Android](#)

Filtro de compatibilidad con Android

El sistema operativo Android utiliza una serie de elementos en el archivo de manifiesto de la aplicación para determinar si la aplicación es compatible con un dispositivo concreto. La incorporación de esta información al archivo de manifiesto es opcional. Si no incluye estos elementos, la aplicación se puede instalar en cualquier dispositivo de Android. Sin embargo, puede que no funcione correctamente en cualquier dispositivo de Android. Por ejemplo, una aplicación de cámara no será útil en un teléfono que no disponga de cámara.

Etiquetas del archivo de manifiesto de Android que se pueden utilizar para filtro:

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

Aplicaciones de cámara

Si se solicita el permiso de cámara para la aplicación, Android asume que la aplicación requiere todas las funciones de cámara disponibles, incluyendo el enfoque y flash automáticos. Si la aplicación no requiere todas las funciones de cámara, o si la cámara es una función opcional, se deben establecer los distintos elementos `uses-feature` para la cámara con el fin de indicar que son opcionales. De lo contrario, los usuarios con dispositivos que carecen de una función o que no tienen cámara, no podrán encontrar la aplicación en Android Market.

En el siguiente ejemplo se muestra cómo solicitar permiso para la cámara y hacer que todas las funciones de cámara sean opcionales:

```
<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>
```

Aplicaciones de grabación de audio

Si se solicita permiso para grabar audio, Android también entiende que la aplicación requiere un micrófono. Si la grabación de audio es una función opcional de la aplicación, se puede añadir una etiqueta `uses-feature` para especificar que el micrófono no es necesario. De lo contrario, los usuarios con dispositivos que no disponen de micrófono no podrá localizar la aplicación en Android Market.

En el siguiente ejemplo se muestra cómo solicitar permiso para utilizar el micrófono mientras que el hardware del micrófono se hace opcional:

```
<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>
```

Más temas de ayuda

[Desarrolladores de Android: Compatibilidad con Android](#)

[Desarrolladores de Android: constantes de nombre de la función de Android](#)

Ubicación de instalación

Se puede permitir que la aplicación se instale o se mueva a la tarjeta de memoria externa, estableciendo el atributo `installLocation` del elemento `manifest` de Android en `auto` o `preferExternal`:

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

El sistema operativo Android no garantiza que la aplicación se instalará en la memoria externa. Un usuario también puede mover la aplicación entre la memoria interna y externa utilizando la aplicación de configuración del sistema.

Aun instalada en la memoria externa, los datos de usuario y la caché de la aplicación como, por ejemplo, el contenido del directorio de almacenamiento de la aplicación, los objetos compartidos y los archivos temporales, aún se almacenan en la memoria interna. Para evitar el uso de demasiada memoria interna, sea selectivo respecto a los datos que se guardan en el directorio de almacenamiento de la aplicación. Las grandes cantidades de datos se deben guardar en SDCard utilizando las ubicaciones `File.userDirectory` o `File.documentsDirectory` (que ambas asignaciones en la raíz de la tarjeta SD en Android).

Activación de Flash Player y otros plug-ins en un objeto StageWebView

En Android 3.0+, una aplicación debe activar la aceleración por hardware en el elemento de la aplicación de Android para que el contenido del plug-in se visualice en un objeto `StageWebView`. Para activar el procesamiento mediante plugin, establezca el atributo `android:hardwareAccelerated` del elemento `application` en `true`:

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

Profundidad de color

AIR 3+

En AIR 3 y versiones posteriores, el motor de ejecución establece la visualización con representación de colores de 32 bits. En versiones anteriores de AIR, el motor de ejecución utiliza colores de 16 bits. Puede hacer que el motor de ejecución utilice colores de 16 bits con el elemento `<colorDepth>` del descriptor de la aplicación:

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

Con profundidad de color de 16 bits se puede aumentar el rendimiento de procesamiento, pero a costa de la fidelidad de los colores.

Configuración de iOS

La configuración que solo se aplica a los dispositivos de iOS se sitúa en el elemento `<iPhone>` en el descriptor de la aplicación. El elemento `iPhone` puede tener un elemento `InfoAdditions`, un elemento `requestedDisplayResolution`, un elemento `Entitlements`, un elemento `externalSwfs` y un elemento `forceCpuRenderModeForDevices` como elementos secundarios.

El elemento `InfoAdditions` permite especificar pares clave-valor que se añaden al archivo de configuración `Info.plist` para la aplicación. Por ejemplo, los siguientes valores definen la barra de estado de la aplicación y establecen que la aplicación no requiere acceso permanente a redes Wi-Fi.

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

Los parámetros de `InfoAdditions` se encierran entre etiquetas `CDATA`.

El elemento `Entitlements` permite especificar pares clave-valor al archivo de configuración `Entitlements.plist` para la aplicación. El archivo de configuración `Entitlements.plist` proporciona acceso a la aplicación a determinadas funciones de iOS, como las notificaciones push.

Para obtener información más detallada sobre otros parámetros de `Info.plist` y el archivo de configuración `Entitlements.plist`, consulte la documentación para desarrolladores de Apple.

Soporte para tareas en segundo plano en iOS

AIR 3.3

Adobe AIR 3.3 y versiones posteriores admiten la multitarea en iOS si se habilitan ciertos comportamientos en segundo plano:

- Audio
- Actualizaciones de ubicación
- Redes
- Anulación de la ejecución de la aplicación en segundo plano

Nota: Con SWF versión 21 y anteriores, AIR no admite la ejecución de iOS y Android en segundo plano cuando se ha establecido `renderMode direct`. Debido a esta restricción, las aplicaciones basadas en Stage3D no pueden ejecutar tareas en segundo plano como la reproducción de audio, actualizaciones de ubicación, carga o descarga de red, etc. iOS no permite llamadas OpenGL o de representación en segundo plano. El sistema iOS cierra las aplicaciones que intentan hacer llamadas OpenGL en segundo plano. Android no restringe ni las llamadas OpenGL ni otras tareas de aplicaciones en segundo plano, como la reproducción de audio. Con SWF versión 22 y posteriores, las aplicaciones de AIR

móviles pueden ejecutarse en un segundo plano cuando se ha establecido `renderMode direct`. El tiempo de ejecución de AIR iOS produce un error de `ActionScript` (3768: la API `Stage3D` no se puede usar durante la ejecución en segundo plano) si las llamadas `OpenGLES` se realizan en segundo plano. Sin embargo, no hay errores en Android porque sus aplicaciones nativas pueden realizar llamadas `OpenGLES` en segundo plano. Para el uso óptimo de recursos móviles, no realice llamadas de procesamiento cuando una aplicación se está ejecutando en segundo plano.

Audio en segundo plano

Para habilitar la reproducción de audio y la grabación en segundo plano, incluya el siguiente par clave-valor en el elemento `InfoAdditions`:

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

Actualizaciones de ubicación en segundo plano

Para habilitar las actualizaciones de ubicación en segundo plano, incluya el siguiente par clave-valor en el elemento `InfoAdditions`:

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

Nota: utilice esta función solo cuando sea necesario, ya que las API de ubicación consumen mucha batería.

Redes en segundo plano

Para ejecutar tareas cortas en segundo plano, la aplicación establece la propiedad `NativeApplication.nativeApplication.executeInBackground` en `true`.

Por ejemplo, la aplicación puede iniciar una operación de carga de archivo tras la cual el usuario acceder a otra aplicación en primer plano. Cuando la aplicación recibe un evento de finalización de carga, puede establecer `NativeApplication.nativeApplication.executeInBackground` en `false`.

Establecer la propiedad `NativeApplication.nativeApplication.executeInBackground` en `true` no garantiza que la aplicación se ejecute indefinidamente, ya que iOS impone un límite de tiempo a las tareas en segundo plano. Cuando iOS detiene el procesamiento en segundo plano, AIR distribuye el evento `NativeApplication.suspend`.

Anulación de la ejecución en segundo plano

La aplicación puede anular de forma explícita la ejecución en segundo plano si incluye el siguiente par clave-valor en el elemento `InfoAdditions`:

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

Configuración reservada de InfoAdditions de iOS

AIR establece varias entradas en el archivo Info.plist generado para garantizar que las funciones del motor de ejecución y la aplicación funcionan correctamente. No se pueden definir las siguientes opciones:

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

Nota: Puede definir *MinimumOSVersion*. La definición de *MinimumOSVersion* se proporciona en AIR 3.3 y posterior.

Compatibilidad con diferentes modelos de dispositivo de iOS

Para la compatibilidad con iPad, incluya la configuración adecuada de clave-valor para `UIDeviceFamily` en el elemento `InfoAdditions`. La configuración de `UIDeviceFamily` se expresa en un conjunto de cadenas. Cada cadena define los dispositivos admitidos. La configuración `<string>1</string>` define la compatibilidad con el iPhone y el iPod touch. La configuración `<string>2</string>` define la compatibilidad con iPad. La configuración `<string>3</string>` define la compatibilidad con tvOS. Si especifica solo una de estas cadenas, solo se admitirá dicha familia de dispositivos. Por ejemplo, la siguiente configuración limita la compatibilidad al iPad:

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

La siguiente configuración admite ambas familias de dispositivos (iPhone/iPod touch e iPad):

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

Además, en AIR 3.7 y posterior, puede utilizar la etiqueta `forceCPURenderModeForDevices` para forzar el modo de representación de CPU para un conjunto de dispositivos específicos y activar el modo de representación de GPU para los restantes dispositivos iOS.

Esta etiqueta se agrega como un elemento secundario de la etiqueta `iPhone` y se especifica una lista de nombres de modelos de dispositivos separados por espacios. Para obtener una lista de nombres de modelos de dispositivos válidos, consulte “[forceCPURenderModeForDevices](#)” en la página 234.

Por ejemplo, para utilizar el modo de CPU en iPod, iPhones y iPad antiguos, y activar el modo de GPU para todos los demás dispositivos, especifique lo siguiente en el descriptor de la aplicación:

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

Pantallas de alta resolución

El elemento `requestedDisplayResolution` especifica si la aplicación debe utilizar el modo de resolución *estándar* o *alta* en los dispositivos de iOS con pantallas de alta resolución.

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

En el modo de alta resolución, cada píxel se puede tratar en una pantalla de alta resolución de forma individual. En el modo estándar, la pantalla del dispositivo aparecerá en la aplicación como pantalla de resolución estándar. Al dibujar un solo píxel en este modo, se establecerá el color de cuatro píxeles en la pantalla de alta resolución.

El valor predeterminado es `standard`. Cuando se piensa en dispositivos iOS, `requestedDisplayResolution` se usa como elemento secundario del elemento `iPhone` (no del elemento `InfoAdditionsinitialWindow`).

Si quiere usar configuraciones distintas en dispositivos diferentes, especifique el valor predeterminado como valor del elemento `requestedDisplayResolution`. Use el atributo `excludeDevices` para especificar dispositivos que deberían usar el valor opuesto. Por ejemplo, con el código siguiente se usa el modo de alta resolución para todos los dispositivos que lo admiten, excepto para iPads de tercera generación, que usan el modo estándar:

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

El atributo `excludeDevices` está disponible a partir de AIR 3.6.

Más temas de ayuda

“[requestedDisplayResolution](#)” en la página 247

[Renaun Erickson: Developing for both retina and non-retina iOS screens using AIR 2.6](#)

Esquemas de URI personalizados de Android

Es posible registrar un esquema de URI personalizado para permitir que la aplicación se invoque mediante un vínculo en una página web u otra aplicación nativa en el dispositivo. Para registrar un esquema de URI, añada una clave `CFBundleURLTypes` al elemento `InfoAdditions`. En el siguiente ejemplo se registra un esquema de URI denominado `com.example.app` para permitir que una aplicación se pueda invocar mediante las direcciones URL con la forma: `example://foo`.

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

Cuando la aplicación se invoca mediante un URI personalizado, el objeto `NativeApplication` distribuye un evento `invoke`. La URL del vínculo, incluyendo parámetros de consulta, se ubica en el conjunto `arguments` del objeto `InvokeEvent`. Se puede utilizar cualquier número de esquemas de URI personalizadas.

Nota: los vínculos de una instancia de `StageWebView` no pueden abrir direcciones URL que utilicen un esquema URI personalizado.

Nota: si otra aplicación ya ha registrado un esquema, la aplicación no puede reemplazarlo, ya que la aplicación ya registró ese esquema de URI.

Filtro de compatibilidad con iOS

Añada entradas a un conjunto `UIRequiredDeviceCapabilities` en el elemento `InfoAdditions` si la aplicación solo se debe utilizar en dispositivos con capacidades de hardware o software específicas. Por ejemplo, la siguiente entrada indica que una aplicación requiere una cámara fija y un micrófono:

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

Si un dispositivo no tiene la capacidad correspondiente, la aplicación no podrá instalarse. Entre las opciones de configuración de capacidad relevantes para las aplicaciones de AIR se incluyen:

Telefonía	Flash de cámara
Wifi	Videocámara
sms	Acelerómetro
Cámara fija	Servicios de localización
Cámara de enfoque automático	gps
Cámara orientada al frente	Micrófono

AIR 2.6+ añade automáticamente `armv7` y `opengles-2` a la lista de capacidades requeridas.

Nota: no es necesario incluir estas capacidades en el descriptor de la aplicación para que la aplicación las utilice. Utilice la configuración `UIRequiredDeviceCapabilities` solo para evitar que los usuarios instalen la aplicación en dispositivos en los que no puedan funcionar adecuadamente.

Salida en lugar de pausa

Cuando un usuario cambia de una aplicación de AIR, se sitúa de fondo y se detiene. Si desea que la aplicación se cierre completamente en lugar de detenerse, establezca la propiedad `UIApplicationExitsOnSuspend` en `YES`:


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

Reducir tamaño de descarga mediante la carga externa, los archivos SWF solo de recurso AIR 3.7

Puede minimizar el tamaño de descarga inicial de la aplicación mediante el empaquetamiento de un subconjunto de los archivos SWF usados por su aplicación y la carga del resto (solo activos) de archivos SWF externos en tiempo de ejecución con el método `Loader.load()`. Para utilizar esta función, se debe empaquetar la aplicación de tal manera que ADT mueva todo el código de bytes ActionScript (ABC) de los archivos SWF cargados de forma externa al SWF de la aplicación principal, dejando un archivo SWF que contenga únicamente activos. Esto se hace para cumplir con la regla de la Apple Store que prohíbe descargar código después de que una aplicación esté instalada.

ADT hace lo siguiente para admitir archivos SWF cargados externamente (también denominados archivos SWF quitados):

- Lee el archivo de texto especificado en el subelemento `<externalSwfs>` del elemento `<iPhone>` para acceder a la lista delimitada por líneas de los archivos SWF que se deben cargar en tiempo de ejecución:

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

- Transfiere el código ABC desde cada archivo SWF cargado externamente al ejecutable principal.
- Omite los archivos SWF cargados de forma externa en el archivo .ipa.
- Copia los archivos SWF quitados al directorio `.remoteStrippedSWFs`. Estos archivos SWF se alojan en un servidor web y la aplicación los carga, según el caso, en tiempo de ejecución.

Los archivos SWF que se deben cargar en tiempo de ejecución se indican mediante la especificación de sus nombres, uno por línea en un archivo de texto, como se muestra en el siguiente ejemplo:

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

La ruta de archivo especificada es relativa al archivo descriptor de la aplicación. Además, debe especificar estos archivos SWF como activos en el comando `adt`.

Nota: Esta función solo afecta a los embalajes estándar. Para un empaquetado rápido (utilizando por ejemplo, intérprete, simulador o depurar) ADT no crea archivos SWF quitados.



Para obtener más información sobre esta función, incluido código de ejemplo, consulte [Alojamiento externo de archivos SWF secundarios para aplicaciones de AIR en iOS](#), publicado en un blog del ingeniero de Adobe Abhinav Dhandh.

Compatibilidad con geolocalización

Para la compatibilidad con geolocalización, añada uno de los pares clave-valor siguientes al elemento `InfoAdditions`:

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

Iconos de la aplicación

La siguiente tabla incluye los tamaños de icono utilizados en cada plataforma móvil:

Tamaño del icono	Plataforma
29x29	iOS
36x36	Android
40 x 40	iOS
48x48	Android, iOS
50x50	iOS
57x57	iOS
58x58	iOS
60 x 60	iOS
72x72	Android, iOS
75 x 75	iOS
76 x 76	iOS
80 x 80	iOS
87 x 87	iOS
96x96	Android
100x100	iOS
114x114	iOS
120 x 120	iOS
144x144	Android, iOS
152 x 152	iOS
167 x 167	iOS
180 x 180	iOS
192x192	Android
512x512	Android, iOS
1024x1024	iOS

Especifique la ruta a los archivos de icono en el elemento de icono del archivo descriptor de la aplicación:

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

Si no proporciona un icono de un tamaño determinado, el siguiente tamaño mayor se utiliza y se escala para ajustarse.

Iconos en Android

En Android, los iconos especificados en el descriptor de la aplicación se utilizan como icono de inicio de la aplicación. El icono de inicio de la aplicación debe proporcionarse como un conjunto de imágenes PNG de 36x36, 48x48, 72x72, 96x96, 144x144 y 192x192 píxeles. Estos tamaños de icono se utilizan para pantallas de baja densidad, media densidad y alta densidad, respectivamente.

Los desarrolladores deben enviar el icono de 512x512 píxeles cuando envíen la aplicación a Google Play Store.

Iconos en iOS

Los iconos definidos en el descriptor de la aplicación se utilizan en los siguientes lugares para una aplicación de iOS:

- Icono de 29x29 píxeles: icono de búsqueda de Spotlight para iPhone/iPod de baja resolución e icono de Ajustes para iPad de baja resolución.
- Icono de 40 x 40 píxeles: icono de búsqueda de Spotlight para iPad de baja resolución.
- Icono de 48x48 píxeles: AIR añade un borde a la imagen y la utiliza como icono de 50x50 para la búsqueda de Spotlight en iPad de baja resolución.
- Icono de 50x50 píxeles: búsqueda de Spotlight para iPad de baja resolución.
- Icono de 57x57 píxeles: icono de aplicaciones para iPhone/iPod de baja resolución.
- Icono de 58x58 píxeles: icono de Spotlight para iPhone/iPod con pantalla retina e icono de Ajustes para iPad con pantalla retina.
- Icono de 60x60 píxeles: icono de aplicaciones para iPhone/iPod de baja resolución.
- Icono de 72x72 píxeles (opcional): icono de aplicaciones para iPad de baja resolución.
- Icono de 76x76 píxeles (opcional): icono de aplicaciones para iPad de baja resolución.
- Icono de 80 x 80 píxeles: búsqueda de Spotlight para iPhone/iPod/iPad de baja resolución.
- Icono de 100x100 píxeles: búsqueda de Spotlight para iPad con pantalla retina.
- Icono de 114x114 píxeles: icono de aplicaciones para iPhone/iPod con pantalla retina.
- Icono de 120 x 120 píxeles: icono de aplicaciones para iPhone/iPod de baja resolución.
- Icono de 152 x 152 píxeles: icono de aplicaciones para iPad de baja resolución.
- Icono de 167 x 167 píxeles: icono de aplicaciones para iPad Pro de alta resolución.
- Icono de 512x512 píxeles: icono de aplicaciones para iPhone/iPod/iPad de baja resolución. iTunes muestra este icono. El archivo PNG de 512 píxeles se utiliza únicamente para probar versiones de desarrollo de la aplicación. Cuando se envía la aplicación final al App Store de Apple, debe enviarse la imagen de 512 píxeles por separado y en formato JPG. No está incluido en el IPA.
- Icono de 1024x1024 píxeles: icono de aplicaciones para iPhone/iPod/iPad con pantalla retina.

iOS añade un efecto brillante al icono. No se necesario aplicar el efecto a la imagen de origen. Si quiere eliminar este efecto brillante predeterminado, añade lo siguiente al elemento `InfoAdditions` en el archivo descriptor de la aplicación:

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

Nota: en iOS, los metadatos de aplicación se insertan como metadatos png en los iconos de la aplicación, de modo que Adobe pueda llevar un seguimiento del número de aplicaciones de AIR disponibles en el App Store de Apple iOS. Si no quiere que su aplicación se identifique como aplicación de AIR por sus metadatos de icono, debe desempaquetar el archivo IPA, eliminar los metadatos de icono y volver a empaquetarlo. Este procedimiento se describe en el artículo [Opt-out of AIR application analytics for iOS](#) (Cancelación voluntaria de la analítica de la aplicación de AIR para iOS; en inglés).

Más temas de ayuda

“icon” en la página 235

“imageNxN” en la página 237

[Desarrolladores de Android: Directrices para el diseño de iconos](#)

[Directrices de interfaz humana iOS: Directrices de creación de iconos e imágenes personalizados](#)

Imágenes de inicio de iOS

Además de los iconos de la aplicación, también debe proporcionar al menos una imagen de inicio con el nombre *Default.png*. Opcionalmente, se pueden incluir imágenes de inicio independientes para las diferentes orientaciones de inicio y distintas resoluciones (incluida pantalla retina de alta resolución y relación de aspecto 16:9) y otros dispositivos. También se pueden incluir diferentes imágenes de inicio para utilizarse cuando la aplicación se invoque mediante una URL.

No se hace referencia a este archivo en el descriptor de la aplicación y se debe reemplazar en el directorio de la aplicación raíz. (No coloque los archivos en un subdirectorio.)

Esquema de nomenclatura de archivos

Asigne un nombre a la imagen según el esquema siguiente:

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

Solo se requiere la parte *basename* del nombre del archivo. Es *Default* (con D mayúscula) o el nombre especificado mediante la clave `UILaunchImageFile` en el elemento `InfoAdditions` del descriptor de la aplicación.

La parte del *modificador de tamaño de pantalla* indica el tamaño de la pantalla cuando no se ajusta a uno de los tamaños de pantalla estándar. Este modificador solo se aplica a modelos de iPhone y de iPod touch con pantallas de relación de aspecto 16:9, por ejemplo, el iPhone 5 y el iPod touch (5ª generación). El único valor admitido por este modificador es `-568h`. Dado que estos dispositivos admiten pantallas de alta resolución (retina), el modificador del tamaño de pantalla siempre se utiliza con una imagen con modificador de escala `@2x`. El nombre predeterminado completo de la imagen de inicio para estos dispositivos es `Default-568h@2x.png`.

La parte *urischeme* es la cadena utilizada para identificar el esquema de URI. Esta parte solo se aplica si la aplicación admite uno o varios esquemas de URL personalizados. Por ejemplo, si la aplicación se puede invocar mediante un vínculo como, por ejemplo, `example://foo`, utilice `-example` como parte del esquema del nombre de archivo de la imagen de inicio.

La parte *orientation* permite especificar varias imágenes de inicio en función de la orientación que tenga el dispositivo cuando se inicia la aplicación. Esta parte solo se aplica a imágenes para aplicaciones de iPad. Puede tener uno de los valores siguientes, en referencia a la orientación que tiene el dispositivo cuando se inicia la aplicación:

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

La parte *scale* es @2x (para iPhone 4, iPhone 5 y iPhone 6) o @3x (para iPhone 6 plus) para las imágenes de inicio utilizadas para pantallas de alta resolución (retina). (Omita la parte de la escala completamente para las imágenes utilizadas para las pantallas de visualización estándar.) En el caso de imágenes de inicio para dispositivos más largos, como el iPhone 5 y el iPod touch (5ª generación), también se debe especificar el modificador de tamaño de pantalla -528h después de la parte básica del nombre y antes de cualquier otra parte.

La parte *device* se utiliza para designar imágenes de inicio para dispositivos portátiles pequeños y teléfonos. Esta parte se utiliza cuando la aplicación es una aplicación universal compatible tanto con dispositivos portátiles pequeños como con tabletas con aplicaciones sencillas binarias. Los valores admitidos son ~ipad o ~iphone (para iPhone e iPod touch).

Para iPhone, solo se pueden incluir imágenes de proporción de aspecto verticales. Sin embargo, en el caso de iPhone 6 plus, también se pueden agregar imágenes horizontales. Utilice imágenes de 320x480 píxeles para dispositivos con resolución estándar, imágenes de 640x960 píxeles para dispositivos de alta resolución e imágenes de 640x1136 píxeles para dispositivos con relación de aspecto 16:9 como el iPhone 5 y el iPod touch (5ª generación).

Para iPad, puede incluir imágenes del modo siguiente:

- AIR 3.3 y anterior - Imágenes que no estén a pantalla completa: puede incluir imágenes con relación de aspecto horizontal (1024x748 para resolución normal, 2048x1496 para alta resolución) y vertical (768x1004 para resolución normal, 1536x2008 para alta resolución).
- AIR 3.3 y posterior - Imágenes a pantalla completa: puede incluir imágenes con relación de aspecto horizontal (1024x768 para resolución normal, 2048x1536 para alta resolución) y vertical (768x1024 para resolución normal, 1536x2048 para alta resolución). Tenga en cuenta que al empaquetar una imagen a pantalla completa para una aplicación que no admite imágenes a pantalla completa, los 20 píxeles superiores (40 píxeles para alta resolución) quedarán tapados por la barra de estado. Evite incluir información importante en esta área.

Ejemplos

La siguiente tabla muestra un conjunto de ejemplo de imágenes de inicio que se podrían incluir para una aplicación hipotética que admita la gama más amplia de dispositivos y orientaciones y que se pueda iniciar con direcciones URL utilizando el esquema `example://`:

Nombre de archivo	Tamaño de la imagen	Uso
Default.png	320 x 480	iPhone, resolución estándar
Default@2x.png	640 x 960	iPhone, alta resolución
Default-568h@2x.png	640 x 1136	iPhone, alta resolución, relación de aspecto 16:9

Nombre de archivo	Tamaño de la imagen	Uso
Default-Portrait.png	768 x 1004 (AIR 3.3 y anterior) 768 x 1024 (AIR 3.4 y posterior)	iPad, orientación vertical
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 y anterior) 1536 x 2048 (AIR 3.4 y posterior)	iPad, alta resolución, orientación vertical
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 y anterior) 768 x 1024 (AIR 3.4 y posterior)	iPad, orientación vertical boca abajo
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 y anterior) 1536 x 2048 (AIR 3.4 y posterior)	iPad, alta resolución, orientación vertical boca abajo
Default-Landscape.png	1024 x 768	iPad, orientación horizontal izquierda
Default-LandscapeLeft@2x.png	1536 x 2048	iPad, alta resolución, orientación horizontal izquierda
Default-LandscapeRight.png	1024 x 768	iPad, orientación horizontal derecha
Default-LandscapeRight@2x.png	1536 x 2048	iPad, alta resolución, orientación horizontal derecha
Default-example.png	320 x 480	ejemplo:// URL en iPhone estándar
Default-example@2x.png	640 x 960	ejemplo:// URL en iPhone de alta resolución
Default-example~ipad.png	768 x 1004	ejemplo:// URL en iPad en orientaciones verticales
Default-example-Landscape.png	1024 x 768	ejemplo:// URL en iPad en orientaciones horizontales

Este ejemplo solo ilustra un enfoque. Podría, por ejemplo, utilizar la imagen `Default.png` para iPad y especificar imágenes de inicio concretas para el iPhone y el iPod con `Default~iphone.png` y `Default@2x~iphone.png`.

Véase también

[Guía de programación de aplicaciones iOS: imágenes de inicio de aplicación](#)

Imágenes de inicio para empaquetar para dispositivos iOS

Dispositivos	Resolución (píxeles)	Nombre de la imagen de inicio	Orientación
iPhone			
iPhone 4 (sin retina)	640 x 960	Default~iphone.png	Vertical
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Vertical
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	Vertical
iPhone6, iPhone7	750 x 1334	Default-375w-667h@2x~iphone.png	Vertical

iPhone6+, iPhone 7+	1242 x 2208	Default-414w-736h@3x~iphone.png	Vertical
iPhone6+, iPhone7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	Horizontal
iPad			
iPad 1, 2	768 x 1024	Default-Portrait~ipad.png	Vertical
iPad 1, 2	768 x 1024	Default-PortraitUpsideDown~ipad.png	Vertical boca abajo
iPad 1, 2	1024 x 768	Default-Landscape~ipad.png	Horizontal izquierda
iPad 1, 2	1024 x 768	Default-LandscapeRight~ipad.png	Derecha horizontal
iPad 3, Air	1536 x 2048	Default-Portrait@2x~ipad.png	Vertical
iPad 3, Air	1536 x 2048	Default-PortraitUpsideDown@2x~ipad.png	Vertical boca abajo
iPad 3, Air	1536 x 2048	Default-LandscapeLeft@2x~ipad.png	Horizontal izquierda
iPad 3, Air	1536 x 2048	Default-LandscapeRight@2x~ipad.png	Derecha horizontal
iPad Pro	2048 x 2732	Default-Portrait@2x.png	Vertical
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Horizontal

Directrices para gráficos

puede crear cualquier ilustración que desee para la imagen de inicio, siempre y cuando tenga las dimensiones correctas. No obstante, suele ser mejor que la imagen del archivo coincida con el estado inicial de la aplicación. Se puede crear una imagen de inicio realizando una instantánea de la pantalla de inicio de la aplicación:

- 1 Abra la aplicación en el dispositivo iOS. Cuando aparezca la primera pantalla de la interfaz de usuario, pulse el botón Inicio del dispositivo (debajo de la pantalla) y no lo suelte. Con el botón Inicio presionado, pulse el botón de encendido/reposo (en la parte superior del dispositivo). De este modo hará una captura de pantalla y enviará la imagen al Carrete.
- 2 Transfiera la imagen al equipo de desarrollo desde iPhoto u otra aplicación de transferencia de fotografías.

No incluya texto en la imagen de inicio si la aplicación se localiza a diversos idiomas. La imagen de inicio es una imagen estática y el texto no coincidiría con el de cada idioma correspondiente.

Véase también

[Guía de interfaz humana iOS: imágenes de inicio](#)

Configuración omitida

Las aplicaciones de los dispositivos móviles omiten la configuración de la aplicación que se aplica a las funciones del sistema operativo de escritorio, de la ventana nativa y móvil. Los valores de configuración omitidos son:

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable

- `maxSize`
- `minimizable`
- `minSize`
- `programMenuFolder`
- `resizable`
- `systemChrome`
- `title`
- `transparent`
- `visible`
- `width`
- `x`
- `y`

Empaquetado de una aplicación de AIR móvil

Utilice el comando `-package` de ADT para crear el paquete de la aplicación para una aplicación de AIR destinada a un dispositivo móvil. El parámetro `-target` especifica la plataforma móvil para la que se crea el paquete.

Paquete de Android

Las aplicaciones de AIR en Android utilizan el formato del paquete de la aplicación (APK), en lugar del formato del paquete de AIR.

Los paquetes producidos por utilizando el tipo de destino `APK` se encuentran en un formato que se puede enviar a Android Market. Android Market presenta los requisitos a los que las aplicaciones enviadas deben ajustarse para ser aceptadas. Se deben consultar los requisitos más recientes antes de crear el paquete final. Consulte [Android Developers: Publishing on the Market](#) (Desarrolladores de Android: Publicación en el mercado; en inglés).

Al contrario de lo que sucede con las aplicaciones de iOS, se puede emplear un certificado de firma de código normal de Android para firmar la aplicación; sin embargo, para enviar una aplicación a Android Market, el certificado se debe ajustar a las reglas correspondientes, que requieren que el certificado sea válido al menos hasta el 2033. Se puede crear un certificado con el uso del comando `-certificate` de ADT.

Para enviar una aplicación a un mercado alternativo que no permita que la aplicación requiera una descarga de AIR desde el mercado de Google, se puede especificar una URL de descarga alternativa utilizando el parámetro `-airDownloadURL` de ADT. Cuando un usuario no dispone de la versión necesaria del motor de ejecución de AIR que inicia la aplicación, se le dirige a la URL especificada. Consulte “[Comando package de ADT](#)” en la página 175 para obtener más información.

De forma predeterminada, la aplicación de Android de los paquetes ADT son con tiempo de ejecución compartido. Así pues, para ejecutar la aplicación, el usuario debería instalar un motor de ejecución AIR aparte en el dispositivo.

Nota: Para forzar a ADT para que cree un APK que utilice un motor de ejecución captador, utilice `target apk-captive-runtime`.

Paquetes de iOS

Las aplicaciones de AIR en iOS utilizan el formato de paquete de iOS (IPA), en lugar del formato nativo de AIR.

Los paquetes producidos mediante ADT utilizando el tipo de destino `ipa-app-store` y el certificado de firma de código correcto y el perfil de suministro se encuentran en un formato que se puede enviar a Apple App Store. Utilice el tipo de destino `ipa-ad-hoc` para empaquetar una aplicación para la distribución ad hoc.

Es necesario utilizar el certificado de desarrollador correcto emitido por Apple para poder firmar la aplicación. Los diferentes certificados se utilizan para crear versiones de prueba que se emplean para el empaquetado final antes del envío de la aplicación.

Para ver un ejemplo del modo en que se empaqueta una aplicación iOS con Ant, consulte [Piotr Walczyszyn: Packaging AIR application for iOS devices with ADT command and ANT script](#) (en inglés)

Empaquetado con ADT

El SDK de AIR 2.6 y versiones posteriores admite el empaquetado tanto para iOS como para Android. Antes del empaquetado, todo el código ActionScript, MXML y cualquier código de extensión se debe compilar. También se debe disponer de un certificado de firma de código.

Para obtener una referencia detallada sobre las opciones y los comandos de ADT, consulte “[AIR Developer Tool \(ADT\)](#)” en la página 174.

Paquetes de Android APK

Creación de un paquete de APK

Para crear un paquete de APK, utilice el comando `package` de ADT, estableciendo el tipo de destino en `apk` para las versiones de lanzamiento, `apk-debug` para las versiones de prueba o `apk-emulator` para las versiones en modo de lanzamiento para su ejecución en un emulador.

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

Escriba todo el comando en una sola línea; los saltos de línea del ejemplo anterior solo están presentes para facilitar la lectura. Asimismo, en el ejemplo se da por sentado que la ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319 para obtener ayuda.)

Se debe ejecutar el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son `myApp-app.xml` (archivo descriptor de la aplicación), `myApp.swf` y un directorio de iconos.

Cuando se ejecuta el comando tal y como se muestra, ADT solicitará la contraseña del almacén de claves. (Los caracteres de la contraseña que se escriben no se muestran; simplemente presione Intro cuando termine de introducirlos.)

Nota: De forma predeterminada, todas las aplicaciones de AIR para Android tienen el prefijo `air.` en el nombre del paquete. Para excluir este comportamiento predeterminado, defina la variable de entorno `AIR_NOANDROIDFLAIR` en `true` en el equipo.

Creación de un paquete de APK para una aplicación con extensiones nativas

Para crear un paquete APK para una aplicación que utilice extensiones nativas, añada la opción `-extdir` además de las opciones habituales de empaquetado. Si varios archivos ANE comparten recursos o bibliotecas, ADT selecciona un solo recurso o biblioteca e ignora otras entradas duplicadas antes de emitir una advertencia. Esta opción especifica el directorio que contiene los archivos ANE empleados por la aplicación. Por ejemplo:

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

Creación de un paquete APK que incluye su propia versión del motor de ejecución de AIR

Para crear un paquete APK que contenga tanto la aplicación como una versión de captación del motor de ejecución de AIR, utilice el destino `apk-captive-runtime`. Esta opción especifica el directorio que contiene los archivos ANE empleados por la aplicación. Por ejemplo:

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

Inconvenientes posibles de esta técnica son:

- Las soluciones críticas de seguridad no están disponibles para los usuarios de forma automática cuando Adobe publica parches de seguridad.
- Huella más grande de la aplicación en la RAM

Nota: cuando se incluye el motor de ejecución, ADT añade los permisos `INTERNET` y `BROADCAST_STICKY` a la aplicación. Estos permisos son necesarios para el motor de ejecución de AIR.

Creación de un paquete de depuración de APK

Para crear una versión de la aplicación que se pueda utilizar con un depurador, utilice `apk-debug` como destino y especifique las opciones de conexión:

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

La etiqueta `-connect` indica al motor de ejecución de AIR del dispositivo dónde conectarse a un depurador remoto en la red. Para realizar la depuración a través de USB, se debe especificar la etiqueta `-listen`, especificando el puerto TCP para utilizar en la conexión de depuración:

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

Para que la mayoría de las funciones de depuración funcionen, también se deben compilar los archivos SWF y SWC de la aplicación con la depuración activada. Consulte “[Opciones de conexión del depurador](#)” en la página 192 para ver una descripción completa de los indicadores `-connect` y `-listen`.

Nota: De forma predeterminada, los paquetes ADT son una copia cautiva del motor de ejecución de AIR con su aplicación de Android mientras se empaqueta la aplicación con el destino `apk-debug`. Para forzar a ADT para que cree un APK que utilice un motor de ejecución externo, defina la variable de entorno `AIR_ANDROID_SHARED_RUNTIME` como `true`.

En Android, la aplicación también debe contar con permiso para acceder a Internet con el fin de conectarse al equipo que ejecuta el depurador a través de la red. Consulte “[Permisos de Android](#)” en la página 82.

Creación de un paquete APK para su uso en un emulador de Android

Se puede utilizar un paquete APK de depuración en un emulador de Android, pero no un paquete de modo de lanzamiento. Para crear un paquete de APK de modo de lanzamiento para su uso en un emulador, utilice el comando `package` de ADT, estableciendo el tipo de destino en `apk-emulator`:

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

En el ejemplo se da por sentado que la ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319 para obtener ayuda.)

Creación de un paquete APK a partir de un archivo de AIR o AIRI

Es posible crear un paquete APK directamente desde un archivo existente de AIR o AIRI:

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

El archivo de AIR debe utilizar el espacio de nombres de AIR 2.5 (o posterior) en el archivo descriptor de la aplicación.

Creación de un paquete APK para la plataforma Android x86

A partir de AIR 14, el argumento `-arch` se puede utilizar para empaquetar un APK para la plataforma Android x86. Por ejemplo:

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

Paquetes de iOS

En iOS, ADT convierte el código de bytes del archivo SWF y otros archivos de origen a una aplicación nativa de iOS.

- 1 Cree el archivo SWF con Flash Builder, Flash Professional o con un compilador de línea de comandos.
- 2 Abra una ventana de comandos o el Terminal y acceda a la carpeta del proyecto de su aplicación para iPhone.
- 3 Posteriormente, utilice la herramienta ADT para crear el archivo IPA y siga esta sintaxis:

```
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
```

Cambie el `adt` de referencia para incluir la ruta completa de la aplicación `adt`. La aplicación `adt` se instala en el subdirectorio `bin` del SDK de AIR.

Seleccione la opción `-target` que corresponda al tipo de aplicación para iPhone que quiera crear:

- `-target ipa-test`: elija esta opción para compilar rápidamente una versión de la aplicación y probarla en el iPhone de desarrollador. También puede usar `ipa-test-interpreter` para una compilación aún más rápida o `ipa-test-interpreter-simulator` para ejecutarlo en el simulador de iOS.
- `-target ipa-debug`: elija esta opción para compilar una versión de depuración de la aplicación y probarla en el iPhone del desarrollador. Con esta opción, puede utilizar una sesión de depuración para recibir salida `trace()` desde la aplicación para iPhone.

Puede incluir una de las siguientes opciones `-connect` (`CONNECT_OPTIONS`) para especificar la dirección IP del equipo de desarrollo en el que se ejecuta el depurador:

- `-connect`: la aplicación intentará conectarse mediante wifi a una sesión de depuración en el equipo de desarrollo utilizado para compilar la aplicación.
- `-connect IP_ADDRESS`: la aplicación intentará conectarse mediante wifi a una sesión de depuración en el equipo con la dirección IP especificada. Por ejemplo:

```
-target ipa-debug -connect 192.0.32.10
```
- `-connect HOST_NAME`: la aplicación intentará conectarse mediante wifi a una sesión de depuración en el equipo con el nombre de host especificado. Por ejemplo:

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

La opción `-connect` es opcional. Si no se especifica, la aplicación depurada resultante no intentará conectarse con el depurador del host. Otra posibilidad es especificar `-listen` en lugar de `-connect` para habilitar la depuración USB descrita en [“Depuración remota con FDB a través de USB”](#) en la página 112.

Si el intento de conexión de depuración falla, la aplicación presenta un diálogo que solicita al usuario que introduzca la dirección IP del equipo host de depuración. Un intento de conexión puede fallar si el dispositivo no está conectado a ninguna red wifi. También puede suceder que el dispositivo esté conectado pero no detrás del firewall del equipo host de depuración.

También puede utilizar `ipa-debug-interpreter` para lograr una compilación más rápida o `ipa-debug-interpreter-simulator` para ejecutar en el simulador de iOS.

Para obtener más información, consulte [“Depuración de una aplicación de AIR móvil”](#) en la página 106.

- `-target ipa-ad-hoc`: elija esta opción para crear una aplicación de implementación ad hoc. Consulte el centro de desarrollo de iPhone de Apple

- `-target ipa-app-store`: elija esta opción para crear una versión final del archivo IPA para su implementación en el App Store de Apple.

Reemplace `PROFILE_PATH` por la ruta del archivo del perfil de suministro de la aplicación. Para obtener más información sobre el suministro de perfiles, consulte “[Configuración de iOS](#)” en la página 71.

Utilice la opción `-platformsdk` para señalar al SDK del simulador de iOS cuando quiera ejecutar su aplicación en el simulador de iOS.

Sustituya `SIGNING_OPTIONS` para que haga referencia a su certificado y contraseña de desarrollador de iPhone. Utilice la siguiente sintaxis:

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

Reemplace `P12_FILE_PATH` por la ruta del archivo de certificado P12. Reemplace `PASSWORD` por la contraseña del certificado. (Consulte el ejemplo siguiente.) Para obtener más información sobre el archivo de certificado P12, consulte “[Conversión de un certificado de desarrollador en un archivo de almacén de claves P12](#)” en la página 207.

Nota: puede utilizar un certificado autofirmado al empaquetar para el simulador de iOS.

Sustituya `APP_DESCRIPTOR` para que haga referencia al archivo descriptor de la aplicación.

Sustituya `SOURCE_FILES` para que haga referencia al archivo SWF principal del proyecto seguido de cualquier otro activo que quiera incluir. Incluya las rutas a todos los archivos de icono que defina en el cuadro de diálogo de configuración de la aplicación en Flash Professional o en un archivo descriptor de la aplicación personalizado. Añada también el archivo de gráfico de la pantalla inicial, `Default.png`.

Utilice la opción `-extdir extension-directory` para especificar el directorio que contiene los archivos ANE (extensiones nativas) que utiliza la aplicación. Si la aplicación no utiliza extensiones nativas, no incluya esta opción.

Importante: no cree un subdirectorio en el directorio de la aplicación llamado `Resources`. El motor de ejecución suele crear una carpeta con este nombre para ajustarse a la estructura del paquete IPA. Si crea su propia carpeta `Resources` generará conflictos muy graves.

Creación de un paquete de iOS para depuración

Para crear un paquete de iOS para la instalación en dispositivos de prueba, utilice el comando `package` de ADT, estableciendo el tipo de destino en `ios-debug`. Antes de ejecutar este comando, se debe haber obtenido un certificado de firma de código de desarrollo y un archivo de suministro de Apple.

```
adt -package  
  
-target ipa-debug  
-storetype pkcs12 -keystore ../AppleDevelopment.p12  
-provisioning-profile AppleDevelopment.mobileprofile  
-connect 192.168.0.12 | -listen  
myApp.ipa  
myApp-app.xml  
myApp.swf icons Default.png
```

Nota: también puede utilizar `ipa-debug-interpreter` para lograr una compilación más rápida o `ipa-debug-interpreter-simulator` para ejecutar en el simulador de iOS

Escriba todo el comando en una sola línea; los saltos de línea del ejemplo anterior solo están presentes para facilitar la lectura. Asimismo, en el ejemplo se da por sentado que la ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319 para obtener ayuda.)

Se debe ejecutar el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son `myApp-app.xml` (archivo descriptor de la aplicación), `myApp.swf`, un directorio de iconos y el archivo `Default.png`.

La aplicación se debe firmar utilizando el certificado de distribución correcto emitido por Apple; no se podrán utilizar otros certificados de firma de código.

Utilice la opción `-connect` para la depuración wifi. La aplicación intenta iniciar una sesión de depuración con Flash Debugger (FDB) ejecutándose en la IP o nombre de host definido. Utilice la opción `-listen` para depuración USB. Primero inicie la aplicación y después inicie FDB, que a su vez inicia una sesión de depuración para la aplicación en ejecución. Consulte [“Conexión a Flash Debugger”](#) en la página 110 para obtener más información.

Creación de un paquete de iOS para el envío de App Store de Apple.

Para crear un paquete de iOS para su envío al almacén de aplicaciones de Apple, utilice el comando `package` de ADT, estableciendo el tipo de destino en `ios-app-store`. Antes de ejecutar este comando, se debe haber obtenido un certificado de firma de código de distribución y un archivo de suministro de Apple.

```
adt -package  
  
-target ipa-app-store  
-storetype pkcs12 -keystore ../AppleDistribution.p12  
-provisioning-profile AppleDistribution.mobileprofile  
myApp.ipa  
myApp-app.xml  
myApp.swf icons Default.png
```

Escriba todo el comando en una sola línea; los saltos de línea del ejemplo anterior solo están presentes para facilitar la lectura. Asimismo, en el ejemplo se da por sentado que la ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte [“Variables del entorno de ruta”](#) en la página 319 para obtener ayuda.)

Se debe ejecutar el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son `myApp-app.xml` (archivo descriptor de la aplicación), `myApp.swf`, un directorio de iconos y el archivo `Default.png`.

La aplicación se debe firmar utilizando el certificado de distribución correcto emitido por Apple; no se podrán utilizar otros certificados de firma de código.

Importante: Apple requiere el uso del programa *Application Loader* de Apple para poder cargar las aplicaciones en App Store. Apple solo publica *Application Loader* para Mac OS X. De este modo, mientras se puede desarrollar una aplicación de AIR para iPhone utilizando un equipo de Windows, se debe tener acceso a un equipo que ejecute OS X (versión 10.5.3 o posterior) para enviar la aplicación a App Store. El programa *Application Loader* se puede obtener en el centro para desarrolladores iOS de Apple (*Apple iOS Developer Center*).

Creación de un paquete de iOS para la distribución ad hoc

Para crear un paquete de iOS, utilice el comando `package` de ADT, estableciendo el tipo de destino en `ios-ad-hoc`. Antes de ejecutar este comando, se debe haber obtenido un certificado de firma de código de desarrollo y un archivo de suministro de Apple.

```
adt -package  
  
-target ipa-ad-hoc  
-storetype pkcs12 -keystore ../AppleDistribution.p12  
-provisioning-profile AppleDistribution.mobileprofile  
myApp.ipa  
myApp-app.xml  
myApp.swf icons Default.png
```

Escriba todo el comando en una sola línea; los saltos de línea del ejemplo anterior solo están presentes para facilitar la lectura. Asimismo, en el ejemplo se da por sentado que la ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte [“Variables del entorno de ruta”](#) en la página 319 para obtener ayuda.)

Se debe ejecutar el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son `myApp-app.xml` (archivo descriptor de la aplicación), `myApp.swf`, un directorio de iconos y el archivo `Default.png`.

La aplicación se debe firmar utilizando el certificado de distribución correcto emitido por Apple; no se podrán utilizar otros certificados de firma de código.

Creación de un paquete iOS para una aplicación con extensiones nativas

Para crear un paquete iOS para una aplicación que utilice extensiones nativas, recurra al comando `package` de ADT con la opción `-extdir`. Utilice el comando ADT según proceda para el destino (`ipa-app-store`, `ipa-debug`, `ipa-ad-hoc`, `ipa-test`). Por ejemplo:

```
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
```

Escriba todo el comando en una sola línea; los saltos de línea del ejemplo anterior solo están presentes para facilitar la lectura.

En cuanto a las extensiones nativas, el ejemplo asume que el directorio llamado `extensionsDir` se encuentra en el directorio en el que se ejecuta el comando. El directorio `extensionsDir` contiene los archivos ANE utilizados por la aplicación.

Depuración de una aplicación de AIR móvil

La aplicación móvil de AIR se puede depurar de varios modos. La forma más sencilla de localizar los problemas de lógica de la aplicación es depurar el equipo de desarrollo utilizando ADL o el simulador de iOS. También es posible instalar la aplicación en un dispositivo y realizar la depuración de forma remota con el depurador de Flash ejecutándose en un equipo de escritorio.

Simulación del dispositivo utilizando ADL

La forma más rápida y sencilla para probar y depurar la mayoría de las funciones de la aplicación móvil consiste en ejecutar la aplicación en un equipo de desarrollo con la utilidad Adobe Debug Launcher (ADL). ADL utiliza el elemento `supportedProfiles` en el descriptor de la aplicación para seleccionar qué perfil utilizar. Si se incluyen varios perfiles, ADL utiliza el primero de la lista. El parámetro `-profile` de ADL también se puede utilizar para seleccionar uno de los demás perfiles de la lista `supportedProfiles`. (Si no se incluye ningún elemento `supportedProfiles` en el descriptor de la aplicación, se puede especificar cualquier perfil para el argumento `-profile`.) Por ejemplo, utilice el siguiente comando para iniciar una aplicación que simule el perfil de dispositivo móvil:

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

Al simular el perfil móvil en el escritorio, la aplicación se ejecuta en un entorno que es el más similar a un dispositivo móvil de destino. Las API de ActionScript que no forman parte del perfil móvil no están disponibles. Sin embargo, ADL no distingue entre las capacidades de los diferentes dispositivos móviles. Por ejemplo, puede enviar pulsaciones de teclas programables simuladas a la aplicación, aunque el dispositivo de destino real no utilice este tipo de teclas.

ADL admite las simulaciones de los cambios de orientación del dispositivo y la introducción con teclas programables mediante los comandos de menú. Cuando ADL se ejecuta en el perfil del dispositivo móvil, ADL muestra un menú (en la ventana de la aplicación o la barra de menú de escritorio) que permite introducir la rotación del dispositivo o la introducción mediante las teclas programables.

Introducción con teclas programables

ADL simula los botones de las teclas programables para los botones Back (Atrás), Menu (Menú) y Search (Buscar) en un dispositivo móvil. Estas teclas se pueden enviar al dispositivo simulado utilizando el menú que se muestra cuando ADL se inicia utilizando el perfil móvil.

Rotación del dispositivo

ADL permite simular la rotación del dispositivo mediante el menú que se muestra cuando ADL se inicia utilizando el perfil móvil. El dispositivo simulado se puede rotar de derecha a izquierda.

La simulación de rotación solo afecta a una aplicación que activa la orientación automática. Esta función se puede activar estableciendo el elemento `autoOrients` en `true` en el descriptor de la aplicación.

Tamaño de la pantalla

La aplicación se puede probar en diferentes tamaños de pantalla, estableciendo el parámetro `-screensize` de ADL. Se puede transmitir el código para uno de los tipos de pantalla predefinidos o una cadena que contenga los cuatro valores que representan las dimensiones de píxel de las pantallas maximizada y normal.

Especifique siempre las dimensiones de píxel de la visualización horizontal, esto es, especifique la anchura como un valor más pequeño que el valor de la altura. Por ejemplo, el siguiente comando podría abrir ADL para simular la pantalla utilizada en Motorola Droid:

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Para obtener una lista de tipos de pantallas predefinidas, consulte [“Uso de ADL”](#) en la página 168.

Limitaciones

Algunas API que no se admiten en el perfil de escritorio no se pueden simular mediante ADL. Entre las API que no se simulan se incluyen:

- Accelerometer
- `cacheAsBitmapMatrix`
- CameraRoll
- CameraUI
- Localización geográfica
- Características multitáctil y gestos en los sistemas operativos de escritorio que no admiten estas funciones
- `SystemIdleMode`

Si la aplicación utiliza estas clases, se deben probar las funciones en un emulador o dispositivo real.

Del mismo modo, existen APIs que funcionan cuando se ejecutan en ADL en el escritorio, pero que no funcionan en todos los tipos de dispositivos móviles. Se incluyen:

- Códec de audio Speex y AAC.
- Accesibilidad y compatibilidad con el lector de pantalla.
- RTMPE

- Carga de archivos SWF que contienen código de bytes de ActionScript.
- Sombreados de PixelBender.

Asegúrese de probar las aplicaciones que utilicen estas funciones en los dispositivos de destino, ya que ADL no replica completamente el entorno de ejecución.

Simulación de dispositivos con el simulador de iOS

El simulador de iOS (solo para Mac) permite ejecutar y depurar aplicaciones de iOS fácilmente. Cuando se realizan pruebas con el simulador de iOS, no es necesario disponer de certificado de desarrollador ni de perfil de aprovisionamiento. Tiene que seguir creando un certificado p12, pero puede ser autofirmado.

De forma predeterminada, ADT siempre inicia el simulador de iPhone. Para cambiar el dispositivo de simulación, haga lo siguiente:

- Utilice el comando que se indica a continuación para ver los simuladores disponibles.

```
xcrun simctl list devices
```

El resultado puede ser similar al siguiente.

```
== 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)
....
```

- Puede elegir un simulador específico definiendo la variable de entorno AIR_IOS_SIMULATOR_DEVICE como se indica a continuación:

```
export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
```

Reinicie el proceso tras definir la variable de entorno y ejecute la aplicación en el dispositivo de simulador que desee.

Nota: si utiliza ADT con el simulador de iOS, también debe incluir la opción `-platformsdk` y especificar la ruta de acceso al SDK del simulador de iOS.

Para ejecutar una aplicación en el simulador de iOS:

- 1 Utilice el comando `adt -package` con `-target ipa-test-interpreter-simulator 0 -target ipa-debug-interpreter-simulator`, tal como se indica en el siguiente ejemplo:

```
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
```

Nota: Ahora ya no se necesitan opciones de forma en caso de simuladores, por lo tanto no se puede proporcionar ningún valor en el indicador de `-keystore` puesto que ADT no lo proporcionará.

- 2 Utilice el comando `adt -installApp` para instalar la aplicación en el simulador de iOS, tal como se indica en el siguiente ejemplo:

```
adt -installApp
        -platform ios
        -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
        -device ios-simulator
        -package sample_ipa_name.ipa
```

- 3 Utilice el comando `adt -launchApp` para ejecutar la aplicación en el simulador de iOS, tal como se indica en el siguiente ejemplo:

Nota: De forma predeterminada, el comando `adt -launchApp` ejecuta la aplicación en el simulador de iPhone. Para ejecutar la aplicación en el simulador de iPad, exporte la variable de entorno, `AIR_IOS_SIMULATOR_DEVICE = "iPad"` y, a continuación, utilice el comando `adt -launchApp`.

```
adt -launchApp
        -platform ios
        -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
        -device ios-simulator
        -appid sample_ipa_name
```

Para probar una extensión nativa en el simulador de iOS, utilice el nombre de plataforma `iPhone-x86` en el archivo `extension.xml` y especifique `library.a` (biblioteca estática) en el elemento `nativeLibrary`, tal como se indica en el siguiente ejemplo de `extension.xml`:

```
<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>
```

Nota: cuando se prueba una extensión nativa en el simulador de iOS, no se utiliza la biblioteca estática (archivo `.a`) que se compila para el dispositivo. De hecho, se debe utilizar la biblioteca estática compilada para el simulador.

Sentencias Trace

Cuando la aplicación móvil se ejecuta en el escritorio, la salida de la sentencia trace se imprime en la ventana de la terminar o el depurador utilizados para iniciar ADL. Cuando la aplicación se ejecuta en un dispositivo o emulador, es posible configurar una sesión de depuración remota en la salida de la sentencia trace. Si se admite, la salida de la sentencia trace también se puede ver utilizando las herramientas de desarrollo proporcionadas por el dispositivo o marcador del sistema operativo.

En todos los casos, los archivos SWF de la aplicación se deben compilar con la depuración activada para que el motor de ejecución genere cualquier sentencia trace.

Sentencias trace remotas en Android

Cuando la ejecución se realiza en un emulador o dispositivo de Android, la salida de la sentencia trace se puede ver en el registro del sistema de Android usando la utilidad Android Debug Bridge (ADB) incluida en el SDK de Android. Para ver la salida en su aplicación, ejecute el siguiente comando desde un símbolo del sistema o ventana de terminal de su equipo de desarrollo:

```
tools/adb logcat air.MyApp:I *:S
```

siendo *MyApp* el ID de la aplicación de AIR de su aplicación. El argumento `*:S` suprime la salida de todos los demás procesos. Para ver la información del sistema sobre la aplicación además de la salida de la sentencia trace, `ActivityManager` puede incluirse en la especificación del filtro logcat:

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

En estos ejemplos de comandos se da por sentado que ADB se ejecutando desde la carpeta del SDK de Android o que se ha añadido la carpeta `SDK` a la variable de su entorno de ruta.

Nota: en AIR 2.6+, la utilidad ADB se incluye en el SDK de AIR y se puede encontrar en la carpeta `lib/android/bin`.

Sentencias trace remotas en iOS

Para ver la salida de las sentencias trace desde una aplicación que se ejecuta en un dispositivo de iOS, se debe establecer una sesión de depuración remota utilizando Flash Debugger (FDB).

Más temas de ayuda

[Android Debug Bridge: Enable logcat Logging](#)

“[Variables del entorno de ruta](#)” en la página 319

Conexión a Flash Debugger

Para depurar una aplicación que se ejecuta en un dispositivo móvil, el depurador de Flash se puede ejecutar en el equipo de desarrollo y conectarse a través de la red. Para activar la depuración remota, se debe realizar lo siguiente:

- En Android, especifique el permiso `android.permission.INTERNET` en el descriptor de la aplicación.
- Compile los archivos SWF de la aplicación con la depuración activada.
- Empaquete la aplicación con el depurador `-target apk-debug`, para Android o `-target ipa-debug` para iOS, y el indicador `-connect` (depuración wifi) o `-listen` (depuración USB).

Si se lleva a cabo la depuración wifi, por lo que el dispositivo debe poder acceder al puerto TCP 7935 del equipo que ejecuta el depurador de Flash mediante la dirección IP o un nombre de dominio totalmente cualificado. Si se lleva a cabo la depuración remota con USB, el dispositivo debe poder acceder al puerto TCP 7936 o al puerto especificado en el indicador `-listen`.

En iOS, también puede especificar `-target ipa-debug-interpreter` o `-target ipa-debug-interpreter-simulator`.

Depuración remota con Flash Professional

Una vez que la aplicación esté lista para la depuración y los permisos se definan en el descriptor de la aplicación, realice lo siguiente:

- 1 Abra el cuadro de diálogo Configuración de AIR para Android.
- 2 En la ficha Implementación:
 - Seleccione “Depuración del dispositivo” para el tipo de implementación.

- Seleccione “Instalar aplicación en el dispositivo Android conectado” para Tras la publicación.
- Anule la selección de “Iniciar aplicación en el dispositivo Android conectado” para Tras la publicación.
- Establezca la ruta al SDK de Android, si es necesario.

3 Haga clic en Publicar.

La aplicación se instalará y se iniciará en el dispositivo.

4 Cierre el cuadro de diálogo de configuración de Android de AIR.

5 Seleccione Depurar > Comenzar sesión de depuración remota > ActionScript 3 en el menú de Flash Professional.

Flash Professional muestra el mensaje “Esperando la conexión del reproductor” en el panel de salida.

6 Inicie la aplicación en el dispositivo.

7 Indique la dirección IP o el nombre de host del equipo que ejecuta el depurador de Flash en el cuadro de diálogo de conexión de Adobe AIR y, a continuación, haga clic en Aceptar.

Depuración remota con FDB a través de una conexión de red

Para depurar una aplicación que se ejecuta en un dispositivo con Flash Debugger (FDB) de la línea de comandos, en primer lugar ejecute el depurador en el equipo de desarrollo y después inicie la aplicación en el dispositivo. El siguiente procedimiento utiliza las herramientas AMXMLC, FDB y ADT para compilar, empaquetar y depurar una aplicación en el dispositivo. En los ejemplos se supone que se está utilizando un SDK de Flex y AIR combinado y que el directorio bin se incluye en la variable del entorno de la ruta. (Esta suposición se realiza simplemente para simplificar los ejemplos del comando.)

1 Abra una ventana del símbolo del sistema o terminal y examine el directorio que contiene el código de origen para la aplicación.

2 Compile la aplicación con amxmlc, activando la depuración:

```
amxmlc -debug DebugExample.as
```

3 Empaquete la aplicación utilizando los destinos apk-debug o ipa-debug:

```
Android
adobe.adt:adobe.adt -package -target apk-debug -connect -storetype
pkcs12 -keystore ../../AndroidCert.p12 DebugExample.apk DebugExample-app.xml
DebugExample.swf

iOS
adobe.adt:adobe.adt -package -target ipa-debug -connect -storetype
pkcs12 -keystore ../../AppleDeveloperCert.p12 -provisioning-profile test.mobileprovision
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

Si siempre se utiliza el mismo nombre de host o dirección IP para la depuración, ese valor se puede indicar tras el indicador `-connect`. La aplicación intentará conectarse a esa dirección IP o nombre de host automáticamente. De lo contrario, debe introducir la información en el dispositivo cada vez que inicie la depuración.

4 Instale la aplicación.

En Android, puede utilizar el comando `-installApp` de ADT:

```
adobe.adt:adobe.adt -installApp -platform android -package DebugExample.apk
```

En iOS, puede instalar la aplicación mediante el comando `-installApp` de ADT o a través de iTunes.

5 En un segundo terminal o ventana de comandos y ejecutando FDB:

```
fdb
```

6 En la ventana de FDB, escriba el comando `run`:

```
Adobe fdb (Flash Player Debugger) [build 14159]
Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
(fdb) run
Waiting for Player to connect
```

- 7 Inicie la aplicación en el dispositivo.
- 8 Una vez que la aplicación se inicie en el dispositivo o emulador, se abrirá el cuadro de diálogo de conexión de Adobe AIR. (Si se ha especificado un nombre de host o dirección IP con la opción `-connect` cuando se empaquetó la aplicación, se intentará realizar la conexión automáticamente utilizando dicha dirección.) Indique la dirección adecuada y puntée en OK (Aceptar).

Para poder conectarse al depurador en este modo, el dispositivo debe ser capaz de resolver la dirección o nombre de `gist` y conectarse al puerto TCP 7935. Es necesario disponer de una conexión de red.

- 9 Cuando el motor de ejecución remoto se conecta al depurador, los puntos de corte se pueden establecer con el comando `break` de FDB y después iniciar la ejecución con el comando `continue`:

```
(fdb) run
Waiting for Player to connect
Player connected; session starting.
Set breakpoints and then type 'continue' to resume the
session.
[SWF]
Users:juser:Documents:FlashProjects:DebugExample:DebugExample.swf - 32,235 bytes after
decompression
(fdb) break clickHandler
Breakpoint 1 at 0x5993: file DebugExample.as, line 14
(fdb) continue
```

Depuración remota con FDB a través de USB

AIR 2.6 (Android) AIR 3.3 (iOS)

Para depurar una aplicación a través de una conexión USB, debe empaquetar la aplicación con la opción `-listen` y no con `-connect`. Si se especifica la opción `-listen`, el motor de ejecución detecta una conexión de Flash Debugger (FDB) en el puerto TCP 793 cuando se inicia la aplicación. A continuación, ejecuta FDB con la opción `-p` y FDB inicia la conexión.

Procedimiento de depuración a través de USB para Android

Para que el depurador de Flash que se ejecuta en el equipo de escritorio se pueda conectar al motor de ejecución de AIR que se ejecuta en el dispositivo o emulador, se debe usar Android Debug Bridge (ADB - la utilidad del SDK de Android) o iOS Debug Bridge (IDB - la utilidad del SDK de AIR) para reenviar el puerto del dispositivo al puerto de escritorio.

- 1 Abra una ventana del símbolo del sistema o terminal y examine el directorio que contiene el código de origen para la aplicación.
- 2 Compile la aplicación con `amxmlc`, activando la depuración:

```
amxmlc -debug DebugExample.as
```

- 3 Empaquete la aplicación con el destino de depuración adecuado (por ejemplo, `apk-debug`) y especifique la opción `-listen`:

```
adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

4 Conecte el dispositivo al equipo de depuración con cable USB. (Este procedimiento también se puede utilizar para depurar una aplicación que se ejecute en un emulador, en cuyo caso, una conexión USB no será necesaria ni posible.)

5 Instale la aplicación.

Se puede usar el comando `-installApp` de ADT:

```
adt -installApp -platform android -package DebugExample.apk
```

6 Reenvíe el puerto TCP 7936 desde el dispositivo o emulador al equipo de escritorio con la utilidad Android ADB:

```
adb forward tcp:7936 tcp:7936
```

7 Inicie la aplicación en el dispositivo.

8 En una ventana de comandos o terminal, ejecute FDB utilizando la opción `-p`:

```
fdb -p 7936
```

9 En la ventana de FDB, escriba el comando `run`:

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                     Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                     (fdb) run
```

10 La utilidad FDB intentará conectarse con la aplicación.

11 Cuando se establece la conexión remota, los puntos de corte se pueden establecer con el comando `break` de FDB y después iniciar la ejecución con el comando `continue`:

```
(fdb) run
                                     Player connected; session starting.
                                     Set breakpoints and then type 'continue' to resume the
session.
                                     [SWF]
Users:juser:Documents:FlashProjects:DebugExample:DebugExample.swf - 32,235 bytes after
decompression
                                     (fdb) break clickHandler
Breakpoint 1 at 0x5993: file DebugExample.as, line 14
                                     (fdb) continue
```

Nota: el puerto número 7936 se utiliza como predeterminado para la depuración con USB por el motor de ejecución de AIR y FDB. Es posible especificar diferentes puertos para utilizar con el parámetro del puerto `-listen` de ADT y el parámetro del puerto `-p` de FDB. En este caso, se debe utilizar la utilidad *Android Debug Bridge* para reenviar el número de puerto especificado en ADT al puerto especificado en FDB: `adb forward tcp:adt_listen_port# tcp:fdb_port#`

Procedimiento de depuración a través de USB para iOS

Para que el depurador de Flash que se ejecuta en el equipo de escritorio se pueda conectar al motor de ejecución de AIR que se ejecuta en el dispositivo o emulador, se debe usar la utilidad *iOS Debug Bridge* (IDB, la utilidad del SDK de AIR) reenviar el puerto del dispositivo al puerto de escritorio.

1 Abra una ventana del símbolo del sistema o terminal y examine el directorio que contiene el código de origen para la aplicación.

2 Compile la aplicación con `amxmlc`, activando la depuración:

```
amxmlc -debug DebugExample.as
```

3 Empaque la aplicación con el destino de depuración adecuado (por ejemplo `ipa-debug` o `ipa-debug-interpret`) y especifique la opción `-listen`:

```
adt -package -target ipa-debug-interpreter -listen 16000
                                xyz.mobileprovision -storetype pkcs12 -keystore
Certificates.p12
                                -storepass pass123 OutputFile.ipa InputFile-app.xml
InputFile.swf
```

- 4 Conecte el dispositivo al equipo de depuración con cable USB. (Este procedimiento también se puede utilizar para depurar una aplicación que se ejecute en un emulador, en cuyo caso, una conexión USB no será necesaria ni posible.)

- 5 Instalación y arranque de la aplicación en el dispositivo iOS. En AIR 3.4 y posterior, puede utilizar `adt -installApp` para instalar la aplicación a través de USB.

- 6 Determine el dispositivo con el comando `idb -devices` (IDB se encuentra en `air_sdk_root/lib/aot/bin/iOSBin/idb`):

```
./idb -devices

                                List of attached devices
                                Handle    UUID
                                1         91770d8381d12644df91fbcee1c5bbdacb735500
```

Nota: (AIR 3.4 y posterior) Puede utilizar `adt -devices` en lugar de `idb -devices` para determinar el control del dispositivo.

- 7 Redirija un puerto del escritorio al puerto especificado en el parámetro `adt -listen` (en este caso, 16000; el valor predeterminado es 7936) con la utilidad IDB y el ID de dispositivo encontrado en el paso anterior:

```
idb -forward 7936 16000 1
```

En este ejemplo, 7936 es el puerto de escritorio, 16000 es el puerto que detecta el dispositivo conectado y 1 es el ID de dispositivo del dispositivo conectado.

- 8 En una ventana de comandos o terminal, ejecute FDB utilizando la opción `-p`:

```
fdb -p 7936
```

- 9 En la ventana de FDB, escriba el comando `run`:

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                (fdb) run
```

- 10 La utilidad FDB intentará conectarse con la aplicación.

- 11 Cuando se establece la conexión remota, los puntos de corte se pueden establecer con el comando `break` de FDB y después iniciar la ejecución con el comando `continúe`:

Nota: el puerto número 7936 se utiliza como predeterminado para la depuración con USB por el motor de ejecución de AIR y FDB. Es posible especificar diferentes puertos para utilizar con el parámetro `-listen` de IDB y el parámetro `-p` de FDB.

Instalación de AIR y aplicaciones de AIR en dispositivos móviles

Los usuarios finales de la aplicación pueden instalar el motor de ejecución de AIR y las aplicaciones de AIR utilizando el mecanismo de distribución y la aplicación normal para sus dispositivos.

Por ejemplo, en Android, los usuarios pueden instalar aplicaciones desde Android Market. O bien, si han permitido la aplicación de aplicaciones desde orígenes desconocidos en la configuración de la aplicación, los usuarios pueden instalar una aplicación haciendo clic en el vínculo de una página web, o bien, copiando el paquete de la aplicación en su dispositivo y abriéndolo. Si un usuario intenta realizar la instalación en una aplicación de Android, pero no dispone aún del motor de ejecución instalado, se les redirigirá automáticamente a Market donde podrán instalar el motor de ejecución.

En iOS, existen dos formas de distribuir las aplicaciones a usuarios finales. El principal canal de distribución es Apple App Store. También se puede utilizar la distribución ad hoc para permitir que un número limitado de usuarios instalen la aplicación sin visitar App Store.

Instalación de aplicaciones y el motor de ejecución de AIR para desarrollo

Debido a que las aplicaciones de AIR en los dispositivos móviles se instalan como paquetes nativos, se pueden utilizar las instalaciones de la plataforma normal para instalar las aplicaciones para prueba. Si se admiten, es posible emplear los comandos de ADT para instalar las aplicaciones y el motor de ejecución de AIR. Actualmente, este enfoque se admite en Android.

En iOS, las aplicaciones se pueden instalar para su prueba utilizando iTunes. Las aplicaciones de prueba se deben firmar con un certificado de firma de código de Apple emitido específicamente para el desarrollo de la aplicación y empaquetado con un perfil de suministro de desarrollo. Una aplicación de AIR es un paquete con contenido propio en iOS. No se utiliza ningún motor de ejecución independiente.

Instalación de aplicaciones de AIR utilizando ADT

Mientras que se desarrollan aplicaciones de AIR, ADT se puede usar para instalar y desinstalar el motor de ejecución y las aplicaciones. (Su IDE puede integrar estos comandos para que no tenga que ejecutar ADT por sí mismo.)

El motor de ejecución de AIR se puede instalar en un dispositivo o emulador con la utilidad ADT de AIR. El SDK proporcionado para el dispositivo se debe instalar. Utilice el comando `-installRuntime`:

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

Si el parámetro `-package` no se especifica, el paquete del motor de ejecución adecuado para el dispositivo o emulador se selecciona de los disponibles en el SDK de AIR instalado.

Para instalar una aplicación de AIR en Android o en iOS (AIR 3.4 y versión posterior), utilice el comando `-installApp` similar:

```
adt -installApp -platform android -device deviceID -package path-to-app
```

El valor establecido para el argumento `-platform` debe coincidir con el dispositivo en el que se está realizando la instalación.

Nota: las versiones existentes del motor de ejecución de AIR o de la aplicación de AIR se deben instalar antes de proceder a la reinstalación.

Instalación de aplicaciones de AIR en dispositivos de iOS mediante iTunes

Para instalar una aplicación de AIR en un dispositivo de iOS para prueba:

- 1 Abra la aplicación iTunes.
- 2 Si aún no lo ha hecho, añada el archivo de suministro de esta aplicación a iTunes. En iTunes, seleccione Archivo > Añadir a la biblioteca. Posteriormente, seleccione el archivo de suministro (que tendrá `mobileprovision` como tipo de archivo provisional).

- 3 Algunas versiones de iTunes no sustituyen la aplicación si ya está instalada la misma versión. En ese caso, elimine la aplicación del dispositivo y de la lista de aplicaciones de iTunes.
- 4 Haga doble clic en el archivo IPA de su aplicación. Debe aparecer en la lista de aplicaciones en iTunes.
- 5 Conecte el dispositivo al puerto USB del equipo.
- 6 En iTunes, compruebe la ficha Aplicaciones del dispositivo y verifique que la aplicación aparece seleccionada en la lista de aplicaciones para instalar.
- 7 Seleccione el dispositivo en la lista de la izquierda de la aplicación iTunes. Haga clic en el botón Sincronizar. Cuando finalice la sincronización, la aplicación Hello World aparecerá en el iPhone.

Si no tiene instalada la versión más reciente, elimínela del dispositivo y de la lista de aplicaciones de iTunes y vuelva a realizar este procedimiento. Puede darse el caso si la versión instalada utiliza el mismo ID y versión que la aplicación existente.

Más temas de ayuda

[“Comando installRuntime de ADT”](#) en la página 186

[“Comando installApp de ADT”](#) en la página 183

Ejecución de aplicaciones de AIR en un dispositivo

Las aplicaciones de AIR instaladas se pueden iniciar utilizando la interfaz de usuario del dispositivo. Cuando se admite, las aplicaciones también se pueden iniciar de forma remota mediante la utilidad ADT de AIR:

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

El valor del argumento `-appid` debe ser el ID de la aplicación de AIR de la aplicación de AIR que se va a iniciar. Utilice el valor especificado en el descriptor de la aplicación de AIR (sin el prefijo *air.* añadido durante el empaquetado).

Si solo se ejecuta o se incluye un solo dispositivo o emulador, se puede omitir el indicador `-device`. El valor establecido para el argumento `-platform` debe coincidir con el dispositivo en el que se está realizando la instalación. Actualmente, el único valor admitido es *android*.

Eliminación de aplicaciones y del motor de ejecución de AIR

Se pueden emplear los medios normales para eliminar las aplicaciones proporcionadas por el sistema operativo del dispositivo. Cuando se admita, también se puede emplear la utilidad ADT de AIR para eliminar las aplicaciones y el motor de ejecución de AIR. Para eliminar el motor de ejecución, utilice el comando `-uninstallRuntime`:

```
adt -uninstallRuntime -platform android -device deviceID
```

Para desinstalar una aplicación, utilice el comando `-uninstallApp`:

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Si solo se ejecuta o se incluye un solo dispositivo o emulador, se puede omitir el indicador `-device`. El valor establecido para el argumento `-platform` debe coincidir con el dispositivo en el que se está realizando la instalación. Actualmente, el único valor admitido es *android*.

Configuración de un emulador

Para ejecutar la aplicación de AIR en un emulador del dispositivo, se suele usar el SDK del dispositivo para crear y ejecutar una instancia del emulador en su equipo de desarrollo. Posteriormente se podrá instalar la versión del emulador del motor de ejecución de AIR y la aplicación de AIR en el emulador. Se debe tener en cuenta que las aplicaciones en un emulador se suelen ejecutar de forma mucho más lenta que lo hacen en un dispositivo real.

Creación de un emulador de Android

- 1 Inicie la aplicación AVD Manager y el SDK de Android:
 - En Windows, ejecute el archivo Setup.exe del SDK, en la raíz del directorio del SDK.
 - En Mac OS, ejecute la aplicación de Android, en el subdirectorio de herramientas del directorio SDK de Android.
- 2 Seleccione la opción Settings (Configuración) y la opción "Force https://".
- 3 Seleccione la opción Available Packages (Paquetes disponibles). Verá una lista de los SDK disponibles de Android.
- 4 Seleccione un SDK de Android compatible (Android 2.3 o superior) y haga clic en el botón Install Selected (Instalar seleccionado).
- 5 Elija la opción Virtual Devices (Dispositivos virtuales) y haga clic en el botón New (Nuevo).
- 6 Configure los siguientes valores:
 - Nombre para el dispositivo virtual.
 - API de destino como, por ejemplo, Android 2.3, API nivel 8.
 - Tamaño para la tarjeta SD (por ejemplo, 1024).
 - Aspecto (por ejemplo, Default HVGA).
- 7 Haga clic en el botón Create AVD (Crear AVD).

Se debe tener en cuenta que la creación del dispositivo virtual puede tardar un tiempo dependiendo de la configuración del sistema operativo.

Ahora puede iniciar el nuevo dispositivo virtual.

- 1 Seleccione el dispositivo virtual en la aplicación AVD Manager. Se debe incluir el dispositivo virtual creado anteriormente.
- 2 Seleccione el dispositivo virtual y después haga clic en el botón Start (Iniciar).
- 3 Haga clic en el botón Launch (Iniciar) en la siguiente pantalla.

Debe aparecer una ventana del emulador en el escritorio. Esto puede tardar unos segundos. Puede que el sistema operativo Android tarde algún tiempo en inicializarse. Se pueden instalar aplicaciones empaquetadas con *apk-debug* y *apk-emulator* en un emulador. Las aplicaciones empaquetadas con el destino *apk* no funcionan en un emulador.

Más temas de ayuda

<http://developer.android.com/guide/developing/tools/othertools.html#android>

<http://developer.android.com/guide/developing/tools/emulator.html>

Actualización de aplicaciones móviles de AIR

Las aplicaciones de AIR móviles se distribuyen como paquetes nativos y, por lo tanto, utilizan mecanismos de actualización estándar de otras aplicaciones en la plataforma. Generalmente, esto implica el envío al mismo lugar de mercado o el uso del almacenamiento de aplicaciones para distribuir la aplicación original.

Las aplicaciones de AIR móviles no pueden utilizar el marco ni la clase Updater de AIR.

Actualización de aplicaciones de AIR en Android

Para las aplicaciones distribuidas en Android Market, se puede actualizar una aplicación, situando una nueva versión en Market, siempre que se cumplan los siguientes parámetros (estas directivas se aplican mediante Market y no a través de AIR):

- El paquete de APK se firma con el mismo certificado.
- EL ID de AIR es el mismo.
- El valor `versionNumber` del descriptor de la aplicación es superior. (Se debe incrementar el valor `versionLabel`, si se utiliza.)

A los usuarios que hayan descargado la aplicación de Android Market, se les notificará mediante su software de dispositivo que hay disponible una actualización.

Más temas de ayuda

[Android Developers: Publishing Updates on Android Market \(Desarrolladores de Android: Publicación de actualizaciones en Android Market; en inglés\)](#)

Actualización de aplicaciones de AIR en iOS

Para las aplicaciones de AIR distribuidas mediante el almacén de aplicaciones iTunes, es posible actualizar una aplicación enviando la actualización al almacén si se cumplen todos los puntos siguientes (estas políticas se aplican mediante el almacén de aplicaciones de Apple, no mediante AIR):

- El certificado de firma de código y los perfiles de suministro se envían al mismo ID de Apple.
- El paquete IPA utiliza el mismo ID de paquete de Apple
- La actualización no disminuye el grupo de dispositivos compatibles (es decir, si la aplicación original admite dispositivos que ejecutan iOS 3, no se podrá crear una actualización que elimine la compatibilidad con iOS 3).

Importante: dado que el SDK de AIR 2.6 y versiones posteriores no admite iOS 3, y AIR 2 sí lo hace, no es posible actualizar aplicaciones iOS publicadas que hayan sido desarrolladas con AIR 2 y con una actualización en AIR 2.6+.

Uso de notificaciones push

Las notificaciones push permiten a proveedores de notificaciones remotas enviar notificaciones a las aplicaciones que se ejecutan en un dispositivo móvil. AIR 3.4 admite notificaciones push para dispositivos iOS a través del Servicio de notificaciones push de Apple (APNs).

Nota: para habilitar las notificaciones push para una aplicación de AIR for Android, utilice una extensión nativa, como [as3c2dm](#), desarrollada por el experto en Adobe Piotr Walczyszyn.

El resto de esta sección describe cómo habilitar las notificaciones push en aplicaciones de AIR for iOS.

Nota: en esta sección se da por hecho que dispone de un ID de desarrollador de Apple, que está familiarizado con el flujo de trabajo de iOS y que ha desarrollado al menos una aplicación en un dispositivo iOS.

Información general sobre notificaciones push

El Servicio de notificaciones push de Apple (APNs) permite a los proveedores de notificaciones remotas enviar notificaciones a aplicaciones que se ejecutan en dispositivos iOS. El APNs admite los siguientes tipos de notificaciones:

- Alertas
- Globos
- Sonidos

Para obtener información detallada sobre el APNs, consulte developer.apple.com.

El uso de notificaciones push en la aplicación implica múltiples aspectos:

- **Aplicación cliente:** registra las notificaciones push, se comunica con los proveedores de notificaciones remotas y recibe notificaciones push.
- **iOS:** administra la interacción entre la aplicación cliente y el APNs.
- **APNs:** proporciona un objeto tokenID durante el registro del cliente y transfiere notificaciones desde los proveedores de notificaciones remotas al iOS.
- **Proveedor de notificaciones remotas:** almacenan información de tokenID y aplicaciones cliente y envía instantáneamente las notificaciones al APNs.

Flujo de trabajo de registro

El flujo de trabajo para registrar las notificaciones push con un servicio en el lado del servidor es el siguiente:

- 1 La aplicación cliente solicita a iOS la habilitación de notificaciones push.
- 2 iOS envía la solicitud al APNs.
- 3 El servidor del APNs devuelve un objeto tokenID a iOS.
- 4 iOS devuelve el objeto tokenID a la aplicación cliente.
- 5 La aplicación cliente (a través de un mecanismo específico de la aplicación) proporciona el objeto tokenID al proveedor de notificaciones remotas y este lo almacena para usarlo en las notificaciones push.

Flujo de trabajo de notificaciones

El flujo de trabajo de notificaciones es el siguiente:

- 1 El proveedor de notificaciones remotas genera una notificación y transfiere su carga al APNs junto con el objeto tokenID.
- 2 El APNs reenvía la notificación a iOS en el dispositivo.
- 3 iOS traslada instantáneamente la carga de la notificación a la aplicación.

API de notificaciones push

AIR 3.4 ya incluía un conjunto de API que admitían las notificaciones push de iOS. Estas API se encuentran en el paquete `flash.notifications` e incluyen las clases siguientes:

- `NotificationStyle`: define constantes para tipos de notificaciones: `ALERT`, `BADGE` y `SOUND.C`
- `RemoteNotifier`: permite suscribirse y anular la suscripción de notificaciones push.

- `RemoteNotifierSubscribeOptions`: permite seleccionar qué tipos de notificaciones se reciben. Utilice la propiedad `notificationStyles` para definir un vector de cadenas que registre los distintos tipos de notificaciones.

AIR 3.4 también incluye `flash.events.RemoteNotificationEvent`, distribuido por `RemoteNotifier` en los casos siguientes:

- Cuando se crea correctamente la suscripción en una aplicación y se recibe el objeto `tokenId` del APNs.
- Después de recibir una nueva notificación remota.

Además, `RemoteNotifier` distribuye un evento `flash.events.StatusEvent` si se produce un error durante el proceso de suscripción.

Administración de notificaciones push en una aplicación

Para registrar su aplicación en el servicio de notificaciones push, debe llevar a cabo los pasos siguientes:

- Cree el código que le suscriba a las notificaciones push en la aplicación.
- Habilite las notificaciones push en el archivo XML de la aplicación.
- Cree un archivo de suministro y un certificado que habilite los servicios push en iOS.

El siguiente código de ejemplo con comentarios realiza la suscripción a notificaciones push y administra los eventos de notificaciones push:

```
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 notifications 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());
}
}
}
```

Habilite las notificaciones push en el archivo XML de la aplicación

Para usar notificaciones push en la aplicación, incluya lo siguiente en la etiqueta `Entitlements` (debajo de la etiqueta `iphone`):

```
<iphone>
...
  <Entitlements>
    <![CDATA[
      <key>aps-environment</key>
      <string>development</string>
    ]]>
  </Entitlements>
</iphone>
```

Cuando esté listo para publicar la aplicación en el App Store, un elemento `<string>` para desarrollar en producción:

```
<string>production</string>
```

Si la aplicación admite cadenas localizadas, especifique los idiomas en la etiqueta `supportedLanguages`, debajo de la etiqueta `initialWindow`, tal y como se indica en el siguiente ejemplo:

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

Cree un archivo de suministro y un certificado que habilite los servicios push en iOS

Para permitir la comunicación entre la aplicación y el APNs, debe empaquetar la aplicación con un archivo de suministro y un certificado para habilitar los servicios push en iOS del modo siguiente:

- 1 Inicie sesión en su cuenta de desarrollador de Apple.
- 2 Vaya a Provisioning Portal.
- 3 Haga clic en la ficha App IDs.
- 4 Haga clic en el botón New App ID.
- 5 Especifique una descripción y un identificador de paquete (no puede utilizar * en el identificador).

- 6 Haga clic en Submit. El portal de suministro generará su identificador y volverá a mostrar la página de identificadores de aplicaciones.
- 7 Haga clic en Configure (a la derecha del identificador de la aplicación). Aparecerá la página Configure App ID.
- 8 Marque la casilla de verificación Enable for Apple Push Notification. Tenga en cuenta que existen dos tipos de certificados SSL push: uno para desarrollo/pruebas y otro para producción.
- 9 Haga clic en el botón Configure situado a la derecha del certificado SSL push de desarrollo. Aparecerá la página Generate Certificate Signing Request (CSR).
- 10 Genere un CSR con la utilidad Acceso a Llaveros, tal como se indica en la página.
- 11 Genere el certificado SSL.
- 12 Descargue el certificado SSL e instálelo.
- 13 (Opcional) Repita los pasos del 9 al 12 para generar el certificado SSL push de producción.
- 14 Haga clic en Done. Aparecerá la página Configure App ID.
- 15 Haga clic en Done. Aparecerá la página App IDs. Observe el círculo de color verde junto a la notificación push del identificador de su aplicación.
- 16 Es muy importante guardar los certificados SSL, ya que se utilizarán más adelante para la comunicación entre la aplicación y el proveedor.
- 17 Haga clic en la ficha Provisioning para visualizar la página Provisioning Profiles.
- 18 Cree un archivo de suministro para el nuevo identificador de la aplicación y descárguelo.
- 19 Haga clic en la ficha Certificates y descargue un nuevo certificado para el nuevo archivo de suministro.

Uso de sonidos para las notificaciones push

Para habilitar notificaciones con sonido para la aplicación, empaquete los archivos de sonido del mismo modo que haría con cualquier otro activo, pero en el mismo directorio que los archivos SWF y app-xml. Por ejemplo:

```
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 admite los siguientes formatos de datos de sonido (en archivos aiff, wav o caf):

- PCM lineal
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Uso de notificaciones de alerta localizadas

Para usar notificaciones de alerta localizadas en la aplicación, empaquete las cadenas localizadas en forma de carpetas lproj. Por ejemplo, si quiere admitir alertas en español, haga lo siguiente:

- 1 Cree una carpeta es.lproj en el proyecto al mismo nivel que el archivo app-xml.
- 2 Dentro de la carpeta es.lproj, cree un archivo de texto llamado Localizable.Strings.
- 3 Abra el archivo Localizable.Strings en un editor de texto y añada claves de mensaje y las cadenas localizadas correspondientes. Por ejemplo:

```
"PokeMessageFormat" = "La notificación de alertas en español."
```

- 4 Guarde el archivo.

- 5 Cuando la aplicación reciba una notificación de alerta con este valor clave y si el idioma del dispositivo es el español, se visualizará el texto de la alerta traducido.

Configuración de un proveedor de notificaciones remotas

Necesitará un proveedor de notificaciones remotas para enviar notificaciones push a la aplicación. Esta aplicación de servidor actúa como proveedor, acepta entradas push y transfiere la notificación y los datos al APNs que, por su lado, envía la notificación push a una aplicación cliente.

Para obtener información detallada sobre las notificaciones push desde un proveedor de notificaciones remotas, consulte [Comunicación del proveedor con el Servicio de notificaciones push de Apple](#) en la biblioteca de desarrolladores de Apple.

Opciones del proveedor de notificaciones remotas

Las opciones del proveedor de notificaciones remotas le permiten hacer lo siguiente:

- Cree su propio proveedor a partir del servidor open-source APNS-php. Puede configurar un servidor PHP con <http://code.google.com/p/apns-php/>. Este proyecto de Google Code permite diseñar una interfaz que se ajuste a sus requisitos concretos.
- Utilice un proveedor de servicios. Por ejemplo, <http://urbanairship.com/> ofrece proveedores APNs listos para usar. Tras registrarse en este servicio, deberá empezar por proporcionar su símbolo de dispositivo con un código similar al siguiente:

```
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);
}
```

Seguidamente, puede enviar las notificaciones de prueba con las herramientas de Urban Airship.

Certificados para el proveedor de notificaciones remotas

Debe copiar el certificado SSL y la clave privada (generada previamente) en la ubicación adecuada del servidor del proveedor de notificaciones remotas. Lo habitual es combinar estos dos archivos en un único archivo `.pem`. Para ello, haga lo siguiente:

- 1 Abra una ventana de Terminal.
- 2 Cree un archivo `.pem` desde el certificado SSL escribiendo el siguiente comando:

```
openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
```
- 3 Cree un archivo `.pem` desde el archivo de clave privada (`.p12`) escribiendo el siguiente comando:

```
openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
```
- 4 Combine los dos archivos `.pem` en uno solo escribiendo el siguiente comando:

```
cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
```
- 5 Proporcione el archivo `.pem` combinado al proveedor del servidor cuando vaya a crear la aplicación push en el lado del servidor.

Para obtener más información, consulte [Instalación del certificado SSL y la clave en el servidor](#) en el manual *Apple Local and Push Notification Programming Guide*.

Manejo de notificaciones push en una aplicación

Manejar las notificaciones push en una aplicación implica lo siguiente:

- Configuración y aceptación global de notificaciones push por parte del usuario
- Aceptación de notificaciones push individuales por parte del usuario
- Control de notificaciones push y datos de carga de notificaciones

Configuración y aceptación de notificaciones push

La primera vez que un usuario inicia una aplicación habilitada con notificaciones push, iOS muestra un diálogo de ***appname* Would Like to Send You Push Notifications** con los botones No permitir y Aceptar. Si el usuario selecciona Aceptar, la aplicación puede recibir todo tipo de notificaciones a las que se haya suscrito. Si el usuario selecciona No permitir, no recibirá ninguna notificación.

Nota: los usuarios pueden ir a *Ajustes > Notificaciones para controlar los tipos específicos de notificaciones que pueden recibir en cada aplicación habilitada con notificaciones push.*

Apple recomienda que cada vez que se active una aplicación, debe suscribirse a notificaciones push. Cuando la aplicación llama a `RemoteNotifier.subscribe()`, recibe un evento `RemoteNotificationEvent` de tipo `token` que contiene un objeto `tokenId` numérico único de 32 bytes para identificar de forma exclusiva la aplicación en dicho dispositivo.

Cuando el dispositivo recibe una notificación push, muestra un mensaje emergente con los botones Cerrar y Abrir. Si el usuario toque Cerrar, no sucede nada; si el usuario toque Abrir, iOS invoca la aplicación y esta recibe un evento `flash.events.RemoteNotificationEvent` de tipo `notification`, tal como se describe más abajo.

Control de notificaciones push y datos de carga

Cuando el proveedor de notificaciones remotas envía una notificación a un dispositivo (con el objeto tokenID), la aplicación recibe un evento `flash.events.RemoteNotificationEvent` de tipo `notification`, se esté ejecutando o no la aplicación. En este punto, la aplicación lleva a cabo el procesamiento de notificaciones específicas de la aplicación. Si la aplicación gestiona datos de notificaciones, podrá acceder a ellos a través de la propiedad `RemoteNotificationEvent.data` con formato JSON.

Capítulo 8: Desarrollo de aplicaciones de AIR para dispositivos de televisión

Funciones de AIR para TV

Se pueden crear aplicaciones de Adobe® AIR® para dispositivos de TV como, por ejemplo, televisiones, grabadores de vídeo digital y reproductores Blu-ray, si el dispositivo contiene Adobe AIR para TV. AIR para TV se optimiza para los dispositivos de TV, usando, por ejemplo, los aceleradores por hardware de un dispositivo para obtener gráficos y vídeo de alto rendimiento.

Las aplicaciones de AIR para los dispositivos de TV son aplicaciones basadas en SWF, no basadas en HTML. Su aplicación de AIR para TV puede aprovechar la aceleración por hardware, así como otras funciones de AIR que están diseñadas para el “entorno del salón”.

Perfiles de dispositivo

AIR utiliza perfiles para definir un conjunto de dispositivos de destino con capacidades similares. Utilice los siguientes perfiles para las aplicaciones de AIR para TV:

- Perfil `tv`. Utilice este perfil en las aplicaciones de AIR destinadas a un dispositivo de AIR para TV.
- Perfil `extendedTV`. Utilice este perfil si la aplicación de AIR para TV usa extensiones nativas.

Las capacidades de ActionScript definidas para estos perfiles se analizan en “[Perfiles de dispositivo](#)” en la página 256. Las diferencias específicas de ActionScript para las aplicaciones de AIR para TV se indican en [Referencia de ActionScript 3.0 para la plataforma de Adobe Flash](#).

Para obtener información sobre los perfiles de AIR para TV, consulte “[Perfiles admitidos](#)” en la página 149.

Aceleración de hardware

Los dispositivos de televisión proporcionan aceleradores de hardware que aumentan en gran medida el rendimiento de los gráficos y el vídeo en las aplicaciones de AIR. Para aprovechar estos aceleradores de hardware, consulte “[Consideraciones de diseño de las aplicaciones de AIR para TV](#)” en la página 130.

Protección del contenido

AIR para TV permite la creación de amplias experiencias de consumidor en el contenido de vídeo de alta calidad, desde éxitos de Hollywood hasta episodios de TV y películas independientes. Los proveedores de contenido pueden crear aplicaciones interactivas utilizando herramientas de Adobe. Pueden integrar productos de servidor de Adobe en su infraestructura de distribución de contenido o trabajar con uno de los socios del ecosistema de Adobe.

La protección del contenido es un requisito clave para la distribución de vídeo de alta calidad. AIR para TV admite Adobe® Flash® Access™, una solución de monetización y protección de contenido que se ajusta a los rigurosos requisitos de seguridad de los propietarios de contenido, incluyendo importantes estudios de cine.

Flash Access admite los siguientes elementos:

- Descarga y transmisión de vídeo.

- Diversos modelos de negocio, entre los que se incluyen compatibilidad con anuncios, suscripción, alquiler y venta electrónica.
- Diferentes tecnologías de transmisión de contenido, incluyendo la transmisión dinámica HTTP, la transmisión a través de RTMP (Real Time Media Protocol) utilizando Flash® Media Server y la descarga progresiva con HTTP.

AIR para TV también presenta compatibilidad incorporada con RTMPE, la versión cifrada de RTMP, para las soluciones de transmisión existentes con requisitos de seguridad más bajos. RTMPE y las tecnologías de verificación SWF relacionadas se admiten en Flash Media Server.

Para obtener más información, consulte [Adobe Flash Access](#).

Audio multicanal

Desde AIR 3, AIR para TV admite audio multicanal para vídeos de descarga progresiva desde un servidor HTTP. Este soporte incluye los códecs siguientes:

- AC-3 (Dolby Digital)
- E-AC-3 (Enhanced Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

Nota: aún no se admite audio multicanal en vídeos transmitidos desde Adobe Flash Media Server.

Entrada de juegos

Desde AIR 3, AIR para TV admite API de ActionScript que permiten que las aplicaciones puedan comunicarse con dispositivos de entrada de juegos, como joysticks, mandos o wands. Aunque estos dispositivos reciben el nombre de dispositivos de entrada de juegos, no solo los juegos pueden aprovechar los dispositivos: también cualquier aplicación de AIR para TV puede hacerlo.

Hay disponible una amplia gama de dispositivos de entrada con distintas funcionalidades. Por ello, los dispositivos se generalizan en la API de modo que una aplicación pueda funcionar correctamente con distintos tipos de dispositivos de entrada de juegos (muchos de ellos posiblemente desconocidos).

La clase `GameInput` es el punto de entrada de las API de ActionScript de entrada de juegos. Para obtener más información, consulte [GameInput](#).

Procesamiento de gráficos acelerados Stage3D

Desde AIR 3, AIR para TV admite procesamiento de gráficos acelerados Stage3D. Las API [Stage3D](#) de ActionScript son un conjunto de API de bajo nivel con aceleración por GPU que permiten usar funciones 2D y 3D avanzadas. Estas API de bajo nivel ofrecen flexibilidad a los desarrolladores para poder aprovechar la aceleración por hardware de GPU y mejorar significativamente el rendimiento. También puede utilizar motores de juegos que admitan las API Stage3D de ActionScript.

Para obtener más información, consulte [Motores de juegos, 3D y Stage3D](#).

Extensiones nativas

Cuando la aplicación se destina al perfil `extendedTV`, puede utilizar paquetes ANE (extensión nativa de AIR).

Generalmente, un fabricante del dispositivo proporciona paquetes ANE para ofrecer acceso a las funciones del dispositivo que no se admiten de otro modo en AIR. Por ejemplo, una extensión nativa podría permitir el cambio de canales en una televisión o detener la reproducción en un reproductor de vídeo.

Cuando se empaqueta una aplicación de AIR para TV que emplea paquetes ANE, el empaquetado se lleva a cabo en un archivo AIRN en lugar de en un archivo de AIR.

Las extensiones nativas para dispositivos de AIR para TV siempre son extensiones nativas *incluidas en el dispositivo*. Incluidas en el dispositivo quiere decir que las bibliotecas de extensiones están instaladas en el dispositivo de AIR para TV. El paquete ANE incluido en el paquete de la aplicación *nunca* contiene las bibliotecas nativas de la extensión. A veces contiene una versión exclusiva para ActionScript de la extensión nativa. Esta versión exclusiva de ActionScript es un simulador de la extensión. El fabricante del dispositivo instala la extensión real, incluidas las bibliotecas nativas, en el dispositivo.

Si está desarrollando extensiones nativas, tenga en cuenta lo siguiente:

- Consulte siempre con el fabricante del dispositivo si va a crear una extensión nativa de AIR para TV para sus dispositivos.
- En algunos dispositivos de AIR para TV, solamente el fabricante del dispositivo puede crear extensiones nativas.
- En todos los dispositivos de AIR para TV, el fabricante del dispositivo es quien decide qué extensiones nativas se pueden instalar.
- Las herramientas de desarrollo para crear extensiones nativas de AIR para TV varían según cada fabricante.

Para obtener más información sobre el uso de extensiones nativas en la aplicación de AIR, consulte [“Uso de extensiones nativas para Adobe AIR”](#) en la página 157.

Para obtener información sobre la creación de extensiones nativas, consulte [Desarrollo de extensiones nativas para Adobe AIR](#).

Consideraciones de diseño de las aplicaciones de AIR para TV

Consideraciones sobre vídeo

Directrices de la codificación de vídeo

Cuando se transmite vídeo a un dispositivo de TV, Adobe recomienda las siguientes directrices de codificación:

Códec de vídeo:	H.264, perfil principal o alto, codificación progresiva
Resolución:	720i, 720p, 1080i o 1080p
Velocidad de fotogramas:	24 fotogramas por segundo o 30 fotogramas por segundo
Códec de audio:	AAC-LC o AC-3, 44,1 kHz, estéreo, o los siguientes códecs de audio multicanal: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio o DTS-HD Master Audio

Velocidad de bits combinada: Hasta 8 Mbps dependiendo del ancho de banda disponible

Velocidad de bits de audio: Hasta 192 Kbps

Proporción de aspecto de píxeles: 1 × 1

Adobe recomienda el uso del códec H.264 para el vídeo que se transmite a dispositivos de AIR para TV.

***Nota:** AIR para TV también admite vídeo que se codifica con los códecs Sorenson Spark o On2 VP6. Sin embargo, el hardware no descodifica ni presenta estos códecs. Lo normal es que el motor de ejecución descodifique y presente estos códecs utilizando software y, por lo tanto, el vídeo se reproduzca con una velocidad de fotogramas mucho más baja. Por lo tanto, utilice H.264 si es posible.*

Clase StageVideo

AIR para TV admite la presentación y la descodificación de hardware del vídeo codificado con H.264. Utilice la clase StageVideo para activar esta función.

Consulte [Uso de la clase StageVideo para la presentación acelerada por hardware](#) en la *Guía del desarrollador de ActionScript 3.0* para obtener información sobre:

- La API de la clase StageVideo y clases relacionadas.
- Limitaciones del uso de la clase StageVideo.

Para una mejor compatibilidad con las aplicaciones de AIR existentes que utilicen el objeto Video para el vídeo codificado con H.264, AIR para TV utiliza *de forma interna* un objeto StageVideo. Esta operación implica que la reproducción de vídeo se beneficia de la presentación y la descodificación de hardware. Sin embargo, el objeto Video está sujeto a las mismas restricciones que un objeto StageVideo. Por ejemplo, si la aplicación intenta girar el vídeo, no se produce ninguna rotación, debido a que el hardware, no el motor de ejecución, está presentando el vídeo.

Sin embargo, cuando escriba nuevas aplicaciones, utilice el objeto StageVideo para el vídeo codificado con H.264.

Para ver un ejemplo del uso de la clase StageVideo, consulte [Delivering video and content for the Flash Platform on TV](#) (Transmisión de vídeo y contenido para la plataforma de Flash en TV; en inglés).

Directrices de transmisión de vídeo

En un dispositivo de AIR para TV, el ancho de banda disponible de la red puede variar durante la reproducción de vídeo. Por ejemplo, estas variaciones pueden suceder cuando otro usuario comienza a utilizar la misma conexión a Internet.

Por lo tanto, Adobe recomienda que el sistema de transmisión de vídeo utilice capacidades de velocidad de bits adaptativas. Por ejemplo, en el servidor, Flash Media Server admite capacidades de velocidad de bits adaptativas. En el cliente, puede utilizar Open Source Media Framework (OSMF).

Los siguientes protocolos están disponibles para transmitir contenido de vídeo en la red a una aplicación de AIR para TV:

- Transmisión dinámica HTTP y HTTPS (formato F4F)
- Transmisión RTMP, RTMPE, RTMFP, RTMPT y RTMPTE
- Descarga progresiva HTTP y HTTPS

Para más información, consulte las referencias siguientes:

- [Guía del desarrollador de Adobe Flash Media Server](#)
- [Open Source Media Framework](#)

Consideraciones sobre audio

ActionScript para la reproducción de sonido no es diferente en las aplicaciones de AIR para TV respecto a otras aplicaciones de AIR. Para obtener más información, consulte [Trabajo con sonido](#) en la *Guía del desarrollador de ActionScript 3.0*.

En cuanto al soporte de audio multicanal en AIR para TV, tenga en cuenta lo siguiente:

- AIR para TV admite audio multicanal para vídeos de descarga progresiva desde un servidor HTTP. aún no se admite audio multicanal en vídeos transmitidos desde Adobe Flash Media Server.
- Aunque AIR para TV admite muchos códecs de audio, no todos los *dispositivos* AIR para TV admiten todo el conjunto. Utilice el método `flash.system.Capabilities.hasMultiChannelAudio()` para comprobar si un dispositivo AIR para TV admite un códec de audio multicanal concreto, por ejemplo, AC-3.

Por ejemplo, considere una aplicación que descarga progresivamente un archivo de vídeo desde un servidor. El servidor tiene distintos archivos de vídeo H.264 que admiten distintos códecs de audio multicanal. La aplicación puede usar `hasMultiChannelAudio()` para determinar qué archivos de vídeo solicitar al servidor. Como alternativa, la aplicación puede enviar al servidor la cadena contenida en `Capabilities.serverString`. Esta cadena indica qué códecs de audio multicanal están disponibles y permiten al servidor seleccionar el archivo de vídeo adecuado.

- Al usar uno de los códecs de audio DTS, pueden darse casos en los que `hasMultiChannelAudio()` devuelva `true`, pero no se reproduzca el audio DTS.

Por ejemplo, un reproductor de Blu-ray con una salida S/PDIF, conectado a un amplificador antiguo. El amplificador antiguo no admite DTS, pero S/PDIF no tiene ningún protocolo para notificar al reproductor de Blu-ray. Si el reproductor de Blu-ray envía el flujo DTS al amplificador antiguo, el usuario no oye nada. Por lo tanto, cuando se utiliza DTS lo mejor es proporcionar una interfaz de usuario para que el usuario pueda indicar si no se está reproduciendo ningún sonido. A continuación, la aplicación puede volver a un códec diferente.

La siguiente tabla resume cuándo utiliza los diferentes códecs de audio en las aplicaciones de AIR para TV. La tabla también indica cuándo utilizan los dispositivos AIR para TV aceleradores por hardware para descodificar un códec de audio. La descodificación por hardware mejora el rendimiento y descarga la CPU.

Códec de audio	Disponibilidad en dispositivo AIR para TV	Descodificación por hardware	Cuándo utilizar este códec de audio	Más información
AAC	Siempre	Siempre	En vídeos codificados con H.264. Para la transmisión de audio como, por ejemplo, servicio de transmisión de música por Internet.	Cuando se utilice una transmisión AAC de solo audio, encapsule la transmisión de audio en un contenedor MP4.
mp3	Siempre	No	Para sonidos en los archivos SWF de la aplicación. En vídeos codificados con Sorenson Spark o On2 VP6.	Un vídeo H.264 que utiliza mp3 para el audio no se puede reproducir en dispositivos de AIR para TV.
AC-3 (Dolby Digital) E-AC-3 (Enhanced Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Comprobar	Sí	En vídeos codificados con H.264.	Normalmente, AIR para TV transfiere un flujo de audio multicanal a un receptor de audio/vídeo externo que descodifica y reproduce el audio.
Speex	Siempre	No	Recepción de una transmisión de voz en vivo.	Un vídeo H.264 que utiliza Speex para el audio no se puede reproducir en dispositivos de AIR para TV. Utilice Speex solamente con vídeos codificados con Sorenson Spark o On2 VP6.
NellyMoser	Siempre	No	Recepción de una transmisión de voz en vivo.	Un vídeo H.264 que utiliza NellyMoser para el audio no se puede reproducir en dispositivos de AIR para TV. Utilice NellyMoser solamente con vídeos codificados con Sorenson Spark o con On2 VP6.

Nota: algunos archivos de vídeo contienen dos flujos de audio. Por ejemplo, un archivo de vídeo puede contener un flujo AAC y uno AC3. AIR para TV no admite estos archivos de vídeo y si se utilizan, el resultado será ausencia de vídeo y sonido.

Aceleración de hardware de gráficos

Uso de la aceleración de gráficos de hardware

Los dispositivos de AIR para TV proporcionan aceleración de hardware para las operaciones de gráficos 2D. Los aceleradores de gráficos de hardware del dispositivo descargan la CPU para realizar las siguientes operaciones:

- Procesamiento de mapa de bits
- Escala de mapa de bits
- Fusión de mapa de bits
- Relleno de rectángulo sólido

Esta aceleración de gráficos de hardware implica que diversas operaciones gráficas en una aplicación de AIR para TV pueden tener un rendimiento alto. Entre algunas de estas operaciones se incluyen:

- Transiciones de deslizamiento
- Transiciones de escala
- Aparición y desaparición de imágenes
- Composición de varias imágenes con alfa

Para obtener las ventajas de rendimiento de la aceleración de gráficos de hardware para estos tipos de operaciones, utilice una de las siguientes técnicas:

- Establezca la propiedad `cacheAsBitmap` en `true` en los objetos `MovieClip` y otros objetos de visualización que presenten contenido inalterable en su mayoría. A continuación, realice transiciones de deslizamiento, transiciones de aparición o desaparición progresiva y mezcla alfa en estos objetos.
- Utilice la propiedad `cacheAsBitmapMatrix` en los objetos de visualización si desea escalar o trasladar (aplicar cambio de posición `x` e `y`).

Con el uso de las operaciones de la clase `Matrix` para la escala y la traslación, los aceleradores de hardware del dispositivo realizan las operaciones. Como alternativa, se debe tener en cuenta el escenario donde se cambian las dimensiones de un objeto de visualización que tiene su propiedad `cacheAsBitmap` establecida en `true`. Cuando cambian las dimensiones, el software del motor de ejecución vuelve a dibujar el mapa de bits. Al volver a dibujar con software se obtiene un rendimiento más bajo que al escalar con la aceleración de hardware utilizando una operación `Matrix`.

Por ejemplo, considere una aplicación que muestra una imagen que se expande cuando el usuario la selecciona. Utilice la operación de escala `Matrix` varias veces para dar la sensación de la expansión de la imagen. Sin embargo, dependiendo del tamaño de la imagen final y la imagen original, la calidad de la imagen final puede ser inaceptable. Por lo tanto, restablezca las dimensiones del objeto de visualización una vez completadas las operaciones de expansión. Debido a que `cacheAsBitmap` es `true`, el software del motor de ejecución vuelve a dibujar el objeto de visualización, pero solo una vez, y representa una imagen de alta calidad.

***Nota:** generalmente, los dispositivos de AIR para TV no admiten el sesgo y la rotación acelerados con hardware. Por lo tanto, si se especifica el sesgo y la rotación en la clase `Matrix`, AIR para TV realiza todas las operaciones `Matrix` en el software. Estas operaciones de software pueden tener un impacto perjudicial en el rendimiento.*

- Use la clase `BitmapData` para crear un comportamiento personalizado de almacenamiento en caché de mapas de bits.

Para obtener más información sobre el almacenamiento en caché de mapa de bits, consulte lo siguiente:

- [Almacenamiento en caché de objetos de visualización](#)
- [Almacenamiento en caché de mapa de bits](#)
- [Almacenamiento manual en caché de mapa de bits](#)

Administración de memoria gráfica

Para llevar a cabo las operaciones de gráficos acelerados, los aceleradores de hardware utilizan memoria gráfica especial. Si la aplicación utiliza toda la memoria gráfica, la aplicación se ejecuta con más lentitud, ya que AIR para TV vuelve a utilizar el software para las operaciones de gráficos.

Para administrar el uso de memoria gráfica de la aplicación:

- Cuando se termine de utilizar una imagen u otros datos de mapa de bits, libere su memoria gráfica asociada. Para ello, llame al método `dispose()` de la propiedad `bitmapData` del objeto `Bitmap`. Por ejemplo:

```
myBitmap.bitmapData.dispose();
```

***Nota:** al liberar la referencia al objeto `BitmapData` no se libera inmediatamente la memoria gráfica. El recopilador de elementos no utilizados del motor de ejecución libera finalmente la memoria gráfica, pero la llamada a `dispose()` da a la aplicación más control.*

- Utilice PerfMaster Deluxe, una aplicación de AIR que proporciona Adobe para comprender mejor la aceleración de gráficos de hardware en el dispositivo de destino. Esta aplicación muestra los fotogramas por segundo para ejecutar distintas operaciones. Utilice PerfMaster Deluxe para comparar diferentes implementaciones de la misma operación. Por ejemplo, compare el movimiento de una imagen de mapa de bits frente al movimiento de una imagen vectorial. PerfMaster Deluxe se encuentra disponible en [Flash Platform for TV](#).

Administración de la lista de visualización

Para que un objeto de visualización permanezca invisible, establezca la propiedad `visible` del objeto en `false`. El objeto seguirá en la lista de visualización pero AIR para TV no lo procesará ni lo visualizará. Esta técnica resulta útil para objetos que suelen aparecer y desaparecer de la vista, ya que solamente añade un poco de carga de procesamiento. Sin embargo, establecer la propiedad `visible` como `false` no libera recursos de objeto en absoluto. Por lo tanto, cuando se ha terminado de visualizar un objeto, o al menos no se va a visualizar durante un periodo prolongado, se recomienda quitar el objeto de la lista de visualización. Asimismo, se recomienda definir todas las referencias al objeto como `null`. Estas acciones permiten al recopilador de elementos no utilizados liberar recursos del objeto.

Uso de imágenes PNG y JPEG

Dos formatos de imagen comunes en las aplicaciones son PNG y JPEG. En relación a estos formatos de imagen en las aplicaciones de AIR para TV, se debe tener en cuenta lo siguiente:

- AIR para TV suele utilizar la aceleración de hardware para decodificar archivos JPEG.
- AIR para TV suele emplear software para decodificar archivos PNG. La decodificación de archivos PNG en software es rápida.
- PNG es el único formato de mapa de bits que se utiliza en plataformas distintas y que admite transparencias (un canal alfa).

Por lo tanto, utilice estos formatos de imagen tal y como se indica en las aplicaciones:

- Utilice archivos JPEG para las fotografías para aprovechar la decodificación acelerada de hardware.
- Utilice archivos de imagen PNG para los elementos de la interfaz de usuario. Los elementos de la interfaz de usuario pueden tener una configuración alfa y la decodificación de software proporciona un rendimiento suficientemente rápido para los elementos de la interfaz de usuario.

Escenario en las aplicaciones de AIR para TV

Cuando se crea una aplicación de AIR para TV, se debe tener en cuenta lo siguiente al trabajar con la clase `Stage`:

- Resolución de pantalla
- Área de visualización segura
- Modo de escala del escenario
- Alineación del escenario
- Estad de visualización del escenario
- Diseño para varios tamaños de pantalla
- Configuración de calidad del escenario

Resolución de pantalla

Actualmente, los dispositivos de TV suelen tener una de las siguientes resoluciones de pantalla: 540p, 720p y 1080p. Estas resoluciones presentan los siguientes valores en la clase Capabilities de ActionScript:

Resolución de pantalla	Capabilities.screenResolutionX	Capabilities.screenResolutionY
540p	960	540
720p	1280	720
1080p	1920	1080

Para escribir una aplicación de AIR para TV de pantalla completa para un dispositivo específico, codifique `Stage.stageWidth` y `Stage.stageHeight` en la resolución de pantalla del dispositivo. Sin embargo, para escribir una aplicación de pantalla completa que se ejecute en varios dispositivos, utilice las propiedades `Capabilities.screenResolutionX` y `Capabilities.screenResolutionY` para definir las dimensiones del escenario.

Por ejemplo:

```
stage.stageWidth = Capabilities.screenResolutionX;  
stage.stageHeight = Capabilities.screenResolutionY;
```

Área de visualización segura

El *área de visualización segura* de una televisión es un área de la pantalla que está insertada desde los bordes de la pantalla. Esta área está insertada en una posición lo suficientemente lejos para que el usuario final pueda verla en su totalidad, sin que el bisel de la TV oscurezca ninguna parte del área. Debido a que el bisel, que es el marco físico alrededor de la pantalla, varía según el fabricante, la inserción necesaria también varía. El área de visualización segura intenta garantizar el área de la pantalla que se encuentra visible. El área de visualización segura también se denomina *área segura de título*.

La *sobreexploración* es el área de la pantalla que no está visible porque se encuentra detrás del bisel.

Adobe recomienda una inserción del 7,5% en cada borde de la pantalla. Por ejemplo:



Área de visualización segura para una resolución de pantalla de 1920 x 1080

El área de visualización segura siempre se debe tener en cuenta al diseñar una aplicación de AIR para TV de pantalla completa:

- Utilice toda la pantalla para los fondos como, por ejemplo, colores o imágenes de fondo.
- Use únicamente el área de visualización segura para elementos fundamentales de la aplicación como, por ejemplo, texto, gráficos, vídeo y elementos de la interfaz de usuario como, por ejemplo, botones.

En la siguiente tabla se muestran las dimensiones del área de visualización segura para cada una de las resoluciones típicas de pantalla, utilizando una inserción del 7,5%.

Resolución de pantalla	Anchura y altura del área de visualización segura	Anchura de la inserción izquierda y derecha	Altura de la inserción superior e inferior
960 x 540	816 x 460	72	40
1280 x 720	1088 x 612	96	54
1920 x 1080	1632 x 918	144	81

Sin embargo, una práctica mejor consiste en calcular siempre dinámicamente el área de visualización segura. Por ejemplo:

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;  
  
horizontalInset = .075 * Capabilities.screenResolutionX;  
verticalInset = .075 * Capabilities.screenResolutionY;  
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);  
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

Modo de escala del escenario

Establezca `Stage.scaleMode` en `StageScaleMode.NO_SCALE`, y detecte los eventos de cambio de tamaño del escenario.

```
stage.scaleMode = StageScaleMode.NO_SCALE;  
stage.addEventListener(Event.RESIZE, layoutHandler);
```

Esta configuración hace que las coordenadas del escenario sean las mismas que las coordenadas de píxel. Junto con el estado de visualización `FULL_SCREEN_INTERACTIVE` y la alineación del escenario `TOP_LEFT`, esta configuración permite utilizar con eficacia el área de visualización segura.

Concretamente, en las aplicaciones de pantalla completa, este modo de escala significa que las propiedades `stageWidth` y `stageHeight` de la clase `Stage` se corresponden con las propiedades `screenResolutionX` y `screenResolutionY` de la clase `Capabilities`.

Asimismo, cuando la ventana de la aplicación cambia de tamaño, el contenido del escenario conserva su tamaño definido. El motor de ejecución no realiza escala ni diseño automáticos. Asimismo, el motor de ejecución distribuye el evento `resize` de la clase `Stage` cuando la ventana cambia el tamaño. Por lo tanto, se tiene el control total sobre cómo ajustar el contenido de la aplicación cuando esta comienza y cuando la ventana de la aplicación cambia de tamaño.

Nota: el comportamiento de `NO_SCALE` es el mismo que con cualquier aplicación de AIR. Sin embargo, en las aplicaciones de AIR para TV, el uso de esta configuración resulta fundamental en el uso del área de visualización segura.

Alineación del escenario

Establezca `Stage.align` en `StageAlign.TOP_LEFT`:

```
stage.align = StageAlign.TOP_LEFT;
```

Esta alineación sitúa la coordenada 0, 0 en la esquina superior izquierda de la pantalla, lo que resulta adecuado para la colocación de contenido con el uso de ActionScript.

Junto con el modo de escala `NO_SCALE` y el estado de visualización `FULL_SCREEN_INTERACTIVE`, esta configuración permite utilizar de forma eficaz el área de visualización segura.

Estado de visualización del escenario

Establezca `Stage.displayState` en una aplicación de AIR for TV de pantalla completa en `StageDisplayState.FULL_SCREEN_INTERACTIVE`:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Este valor configura la aplicación de AIR para que expanda el escenario a toda la pantalla del usuario con la entrada de usuario activada.

Adobe recomienda el uso del valor `FULL_SCREEN_INTERACTIVE`. Junto con el modo de escala `NO_SCALE` y la alineación del escenario `TOP_LEFT`, esta configuración permite utilizar de forma eficaz el área de visualización segura.

Por lo tanto, para las aplicaciones de pantalla completa, en un controlador para el evento `ADDED_TO_STAGE` en la clase de documento principal, realice lo siguiente:

```
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;
}
```

Posteriormente, en el controlador para el evento `RESIZE`:

- Compare los tamaños de resolución de pantalla con la anchura y la altura del escenario. Si son los mismos, el evento `RESIZE` se ha generado, ya que el estado de visualización del escenario cambió a `FULL_SCREEN_INTERACTIVE`.
- Calcule y guarde las dimensiones del área de visualización segura y las inserciones correspondientes.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {
        // Calculate and save safe viewing area dimensions.
    }
}
```

Cuando las dimensiones del escenario coinciden con `Capabilities.screenResolutionX` y `screenResolutionY`, AIR para TV hace que el hardware entregue la mejor fidelidad posible para el vídeo y los gráficos.

Nota: la fidelidad con la que se visualizan los gráficos y el vídeo en una pantalla de TV puede no coincidir con los valores de `Capabilities.screenResolutionX` y `screenResolutionY`, que a su vez dependen del dispositivo en el que se ejecute AIR para TV. Por ejemplo, una grabadora de vídeo digital que ejecute AIR para TV puede tener una resolución de pantalla de 1280 x 720 y el TV conectado puede tener una resolución de pantalla de 1920 x 1080. No obstante, AIR para TV hace que el hardware entregue la mejor fidelidad posible. Por lo tanto, en este ejemplo, el hardware mostrará un vídeo de 1080p con resolución de pantalla de 1920 x 1080.

Diseño para varios tamaños de pantalla

Es posible desarrollar la misma aplicación de AIR para TV de pantalla completa para que funcione y presente un buen aspecto en varios dispositivos de AIR para TV. Realice lo siguiente:

- 1 Establezca la propiedades del escenario `scaleMode`, `align` y `displayState` en los valores recomendados: `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` y `StageDisplayState.FULL_SCREEN_INTERACTIVE`, respectivamente.
- 2 Configure el área de visualización segura en función de `Capabilities.screenResolutionX` y `Capabilities.screenResolutionY`.
- 3 Ajuste el tamaño y el diseño del contenido en función de la anchura y la altura del área de visualización segura.
Aunque los objetos del contenido son grandes, especialmente en comparación con las aplicaciones de dispositivo móvil, los conceptos como diseño dinámico, posicionamiento relativo y contenido adaptativo son los mismos. Para obtener más información sobre ActionScript para admitir estos conceptos, consulte [Authoring mobile Flash content for multiple screen sizes](#) (Creación de contenido móvil de Flash para varios tamaños de pantalla; en inglés).

Calidad del escenario

La propiedad `Stage.quality` de una aplicación de AIR para TV siempre es `StageQuality.High`. No se puede cambiar.

Esta propiedad especifica la calidad de representación para todos los objetos `Stage`.

Administración de la entrada de control remoto

Los usuarios suelen interactuar con la aplicación de AIR para TV utilizando un mando a distancia. Sin embargo, gestione la introducción mediante teclas del mismo modo que en el teclado de una aplicación de escritorio. Concretamente, administre el evento `KeyboardEvent.KEY_DOWN`. Para obtener más información, consulte [Captura de entradas de teclado](#) en la *Guía del desarrollador de ActionScript 3.0*.

Las teclas del mando a distancia se asignan a constantes de ActionScript. Por ejemplo, las teclas del teclado direccional de un mando a distancia se asignan de la siguiente manera:

Tecla del teclado direccional del mando a distancia	Constante de ActionScript 3.0
Arriba	<code>Keyboard.UP</code>
Abajo	<code>Keyboard.DOWN</code>
Izquierda	<code>Keyboard.LEFT</code>
Derecha	<code>Keyboard.RIGHT</code>
Aceptar o Seleccionar	<code>Keyboard.ENTER</code>

AIR 2.5 ha añadido muchas otras constantes de teclado para admitir la introducción mediante mando a distancia. Para obtener una lista completa, consulte la [clase Keyboard](#) en *Referencia de ActionScript 3.0 para la plataforma de Adobe Flash*.

Para garantizar que la aplicación funcione en tantos dispositivos como sea posible, Adobe recomienda lo siguiente:

- Utilice solo teclas de teclado direccional, si es posible.
Los diferentes dispositivos de mando a distancia presentan conjuntos distintos de teclas. Sin embargo, siempre suelen presentar teclas de teclado direccional.
Por ejemplo, un mando a distancia para un reproductor Blu-ray no suele tener una tecla de “canal posterior” y “canal anterior”. Incluso las teclas para reproducción, pausa y detenimiento no se encuentran en todos los controles remotos.
- Utilice las teclas de información y de menú si la aplicación necesita varias teclas de teclado direccional.
Las teclas de información y menú son las siguientes teclas más comunes de los controles remotos.
- Tenga en cuenta el uso frecuente de los controles remotos universales.
Aunque se esté creando una aplicación para un dispositivo concreto, se debe tener en cuenta que muchos usuarios no utilizan el mando a distancia que acompaña al dispositivo. En su lugar, utilizan un mando a distancia universal. Asimismo, los usuarios no siempre programan el mando a distancia universal para que coincida con todas las teclas del control remoto del dispositivo. Por lo tanto, se recomienda el uso únicamente de las teclas más comunes.
- Asegúrese de que el usuario siempre solucione una situación utilizando una de las teclas del teclado direccional.
En ocasiones la aplicación tiene un buen motivo para utilizar una tecla que no es de las más comunes en los mandos a distancia. Si se proporciona una ruta de escape con una de las teclas del teclado direccional, la aplicación se comportará de forma más eficaz en todos los dispositivos.
- No requiere entrada de puntero a no ser que sepa que el dispositivo de AIR para TV de destino tiene funcionalidad de entrada de puntero.
Aunque muchas aplicaciones de escritorio esperan la entrada del ratón, la mayoría de televisores no admiten entrada de puntero. Por lo tanto, si va a convertir aplicaciones de escritorio para ejecutarse en televisiones, asegúrese de modificar que la aplicación no espere la introducción con ratón. Estas modificaciones incluyen cambios en el control de eventos y cambios en las instrucciones para el usuario. Por ejemplo, cuando aparezca la pantalla de inicio de una aplicación, no muestre ningún texto que indique “Haga clic para comenzar”.

Administración de la selección

Si un elemento de la interfaz de usuario se centra en una aplicación de escritorio, se trata del destino de los eventos de introducción por parte del usuario como, por ejemplo, eventos de teclado y ratón. Asimismo, una aplicación resalta el elemento de la interfaz de usuario con la selección. La administración de la selección en una aplicación de AIR para TV es distinta a la administración en una aplicación de escritorio debido a lo siguiente:

- Las aplicaciones de escritorio suelen utilizar la tecla de tabulación para cambiar la selección al siguiente elemento de la interfaz de usuario. El uso de la tecla de tabulación no se aplica a las aplicaciones de AIR para TV. Los dispositivos de control remoto no suelen disponer de tecla de tabulación. Por lo tanto, la administración de la selección con la propiedad `tabEnabled` de un elemento `DisplayObject` como en el escritorio no es adecuada.
- Las aplicaciones de escritorio esperan que el usuario utilice el ratón para seleccionar un elemento de la interfaz de usuario.

Por lo tanto, en la aplicación, realice lo siguiente:

- Añada un detector de eventos al objeto Stage que detecte eventos de teclado como, por ejemplo, `KeyboardEvent.KEY_DOWN`.
- Proporcione una lógica de la aplicación para determinar qué elemento de la interfaz de usuario destacar para el usuario final. Asegúrese de destacar un elemento de la interfaz cuando se inicie la aplicación.

- En función de la lógica de la aplicación, distribuya un evento Keyboard que Stage reciba en el objeto del elemento de la interfaz de usuario adecuado.

`Stage.focus` o `Stage.assignFocus()` también se pueden utilizar para asignar el enfoque a un elemento de la interfaz de usuario. Posteriormente se puede añadir un detector de eventos a `DisplayObject` para que reciba los eventos de teclado.

Diseño de la interfaz de usuario

Asegúrese de que la interfaz de usuario de la aplicación de AIR para TV funcione bien en las televisiones con la incorporación de estas recomendaciones respecto a los siguientes aspectos:

- Nivel de respuesta de la aplicación.
- Posibilidades de uso de la aplicación.
- Expectativas y personalidad del usuario.

Nivel de respuesta

Utilice las siguientes sugerencias para que las aplicaciones de AIR para TV presenten el máximo nivel de respuesta.

- Cree el archivo SWF inicial de la aplicación lo más pequeño posible.

En el archivo SWF inicial, cargue solamente los recursos necesarios para iniciar la aplicación. Por ejemplo, cargue solo la imagen de la pantalla de inicio de la aplicación.

Aunque esta recomendación es válida para las aplicaciones de AIR de escritorio, es más importante en los dispositivos de AIR para TV. Por ejemplo, los dispositivos de AIR para TV no presentan la potencia de procesamiento equivalente de los equipos de escritorio. Asimismo, almacenan la aplicación en la memoria flash, a la que no es tan fácil de acceder como en los discos duros de los equipos de escritorio.

- Haga que la aplicación se ejecute con una velocidad de fotogramas de al menos 20 fotogramas por segundo.

Diseñe los gráficos para que logren este objetivo. La complejidad de las operaciones de los gráficos puede afectar a los fotogramas por segundo. Para obtener sugerencias sobre la mejora del rendimiento de la representación, consulte [Optimización del rendimiento para la plataforma de Flash](#).

Nota: *el hardware de gráficos en los dispositivos de AIR para TV suele actualizar la pantalla a una velocidad de 60 Hz o 120 Hz (60 o 120 veces por segundo). El hardware examina el escenario para buscar actualizaciones a, por ejemplo, 30 fotogramas por segundo o a 60 fotogramas por segundo para la visualización en la pantalla a 60-Hz o 120-Hz. Sin embargo, el que el usuario cuente con estas velocidades de fotogramas depende de la complejidad de las operaciones gráficas de la aplicación.*

- Actualice la pantalla en 100 - 200 milisegundos de introducción por parte del usuario.

Los usuarios se impacientan si las actualizaciones tardan mucho, lo que implica varias pulsaciones de teclas.

Posibilidades de uso

Los usuarios de las aplicaciones de AIR para TV se encuentran en un entorno de “sala de estar”. Se encuentran sentados frente al televisor a una distancia de unos 3 metros. La habitación se encuentra a veces a oscuras. Se suele utilizar un mando a distancia. Varias personas pueden utilizar la aplicación, unas veces de forma conjunta y otras veces una persona cada vez.

Por lo tanto, para diseñar la interfaz de usuario para distintas posibilidades de uso en una TV, se debe tener en cuenta lo siguiente:

- Los elementos de la interfaz de usuario deben ser amplios.
Al diseñar el texto, los botones y cualquier otro elemento de la interfaz de usuario, se debe tener en cuenta que el usuario está sentado en la sala. Haga que todo sea fácil de ver y de leer a una distancia de, por ejemplo, unos 3 metros. No llene en exceso la pantalla porque esta sea amplia.
- Utilice un buen contraste para que el contenido se pueda ver y leer con facilidad en la sala.
- Facilite la localización de los elementos necesarios de la interfaz de usuario destacándolos.
- Utilice el movimiento solo si es necesario. Por ejemplo, el desplazamiento de una pantalla a la siguiente para una continuidad puede resultar adecuado. Sin embargo, el movimiento puede desconcertar si no ayuda al usuario a desplazarse o no es natural para la aplicación.
- Ofrezca siempre una forma evidente y sencilla para el usuario para que se desplace por la interfaz.

Para obtener más información sobre el uso del mando a distancia, consulte “[Administración de la entrada de control remoto](#)” en la página 139.

Expectativas y personalidad del usuario

Se debe tener en cuenta que los usuarios de aplicaciones de AIR para TV suelen buscar entretenimiento de calidad en TV en un entorno divertido y relajado. No necesariamente deben tener conocimientos sobre informática o tecnología.

Por lo tanto, diseñe las aplicaciones de AIR para TV con las siguientes características:

- No emplee términos técnicos.
- Evite los diálogos modales.
- Utilice instrucciones informales y amables que sean adecuadas para su aplicación en casa y no para un entorno técnico o de trabajo.
- Use gráficos que tengan la calidad de producción elevada que esperan los espectadores de TV.
- Cree una interfaz de usuario que funcione fácilmente con un dispositivo de control remoto. No utilice interfaces de usuario o elementos de diseño que se ajusten mejor a una aplicación de escritorio o para móvil. Por ejemplo, las interfaces de usuario de dispositivos de escritorio y móviles suelen incluir botones de puntero y clic que necesitan un ratón o un dedo.

Fuentes y texto

Puede utilizar fuentes de dispositivo o fuentes incorporadas en la aplicación de AIR para TV.

Las fuentes de dispositivo se encuentran instaladas en un dispositivo. Todos los dispositivos de AIR para TV cuentan con las siguientes fuentes:

Nombre de la fuente	Descripción
_sans	La fuente de dispositivo <code>_sans</code> es un tipo de letra sans-serif. Fuente de dispositivo <code>_sans</code> instalada en todos los dispositivos de AIR para TV es Myriad Pro. Normalmente, el tipo sans-serif tiene mejor aspecto en un televisor que los tipos serif, debido a la distancia de visualización.
_serif	La fuente de dispositivo <code>_serif</code> es un tipo de letra serif. La fuente de dispositivo <code>_serif</code> instalada en todos los dispositivos de AIR para TV es Minion Pro.
_typewriter	La fuente de dispositivo <code>_typewriter</code> es una fuente monospace. La fuente de dispositivo <code>_typewriter</code> instalada en todos los dispositivos de AIR para TV es Courier Std.

Todos los dispositivos de AIR para TV también cuenta con las siguientes fuentes de dispositivo asiáticas:

Nombre de la fuente	Idioma	Categoría de tipo de letra	Código de configuración regional
RyoGothicPlusN-Regular	Japonés	sans	ja
RyoTextPlusN-Regular	Japonés	serif	ja
AdobeGothicStd-Light	Coreano	sans	ko
AdobeHeitiStd-Regular	Chino simplificado	sans	zh_CN
AdobeSongStd-Light	Chino simplificado	serif	zh_CN
AdobeMingStd-Light	Chino tradicional	serif	zh_TW and zh_HK

Las fuentes de dispositivo de AIR para TV:

- Proceden de la biblioteca de tipos Adobe® Type Library.
- Presentan un buen aspecto en televisión.
- Están diseñadas para los títulos de vídeo.
- Son contornos de fuente, no fuentes de mapa de bits.

Nota: los fabricantes suelen incluir otras fuentes en el dispositivo. Estas fuentes del dispositivo proporcionadas por los fabricantes se encuentran instaladas además de las fuentes del dispositivo de AIR para TV.

Adobe proporciona una aplicación denominada FontMaster Deluxe que muestra todas las fuentes de dispositivo en el propio dispositivo. La aplicación se encuentra disponible en [Flash Platform for TV](#).

También se pueden incorporar fuentes a la aplicación de AIR for TV. Para obtener información sobre las fuentes incorporadas, consulte [Representación avanzada de texto](#) en la *Guía del desarrollador de ActionScript 3.0*.

Adobe recomienda lo siguiente en relación al uso de campos de texto TLF:

- Utilice los campos de texto TLF para el texto de idiomas asiáticos para aprovechar la configuración local en la que se ejecuta la aplicación. Establezca la propiedad `locale` del objeto `TextLayoutFormat` asociado con el objeto `TLFTextField`. Para determinar la configuración local actual, consulte [Elección de una configuración regional](#) en la *Guía del desarrollador de ActionScript 3.0*.

- Especifique el nombre de la fuente en la propiedad `fontFamily` en el objeto `TextLayoutFormat` si la fuente no es una de las fuentes del dispositivo de AIR para TV. AIR para TV utiliza la fuente si esta se encuentra disponible en el dispositivo. Si la fuente solicitada no está en el dispositivo, en función de la configuración `regional`, AIR para TV sustituye la fuente del dispositivo adecuada de AIR para TV.
- Especifique `_sans`, `_serif` o `_typewriter` para la propiedad `fontFamily`, junto con la propiedad `locale` para que AIR para TV seleccione la fuente de dispositivo correcta. Dependiendo de la configuración regional, AIR para TV realiza la selección de su conjunto de fuentes de dispositivo asiáticas o su conjunto de fuentes de dispositivo no asiáticas. Estos valores proporciona una forma sencilla de utilizar automáticamente la fuente correcta para las cuatro configuraciones regionales asiáticas principales e inglés.

***Nota:** si se utilizan los campos de texto clásicos para texto de idioma asiático, especifique un nombre de fuente de un dispositivo de AIR para TV para garantizar la representación adecuada. Si sabe que otra fuente se encuentra instalada en el dispositivo de destino, también puede especificarla.*

Respecto al rendimiento de la aplicación, se debe tener en cuenta lo siguiente:

- Los campos de texto clásicos ofrecen un rendimiento más rápido que los campos de texto TLF.
- Un campo de texto clásico que utiliza fuentes de mapas de bits proporciona el rendimiento más rápido.
Las fuentes de mapas de bits proporciona un mapa de bits para cada carácter, a diferencia de las fuentes de contorno, que proporcionan los datos de contorno sobre cada carácter. Tanto las fuentes de dispositivo como las incorporadas pueden ser fuentes de mapa de bits.
- Si se especifica una fuente de dispositivo, asegúrese de que la fuente se encuentra instalada en el dispositivo de destino. Si no está instalada en el dispositivo, AIR para TV busca y utiliza otra fuente que se encuentre instalada en el dispositivo. Sin embargo, este comportamiento ralentiza el rendimiento de la aplicación.
- Tal y como sucede con cualquier objeto de visualización, si un objeto `TextField` no suele modificarse, establezca la propiedad `cacheAsBitmap` del objeto en `true`. Esta configuración mejora el rendimiento de las transiciones como, por ejemplo, aparición y desaparición gradual, deslizamiento y mezcla alfa. Utilice `cacheAsBitmapMatrix` para la escala y traslación. Para obtener más información, consulte “[Aceleración de hardware de gráficos](#)” en la página 133.

Seguridad del sistema de archivos

Las aplicaciones de AIR para TV son aplicaciones de AIR y, por lo tanto, pueden acceder al sistema de archivos del dispositivo. Sin embargo, en un dispositivo de “sala de estar” resulta sumamente importante que una aplicación no pueda acceder a los archivos del sistema del dispositivo o a los archivos de otras aplicaciones. Los usuarios de TV y de los dispositivos asociados no esperan ni toleran ningún fallo de dispositivo: después de todo, están viendo la televisión.

Por lo tanto, una aplicación de AIR para TV cuenta con una vista limitada del sistema de archivos del dispositivo. Al utilizar `ActionScript 3.0`, la aplicación solo puede acceder a directorios específicos (y a sus subdirectorios). Asimismo, los nombres del directorio que se utilizan en `ActionScript` no son los reales del dispositivo. Esta capa adicional protege las aplicaciones de AIR para TV frente al acceso inadvertido o malintencionado a los archivos locales que no les pertenecen.

Para obtener formación, consulte [Directory view for AIR for TV applications](#) (Vista del directorio para las aplicaciones de AIR for TV; en inglés).

Entorno limitado de la aplicación de AIR

Las aplicaciones de AIR for TV se ejecutan en el entorno limitado de la aplicación de AIR, descrito en [The AIR application sandbox](#) (Entorno limitado de la aplicación de AIR; en inglés).

La única diferencia para las aplicaciones de AIR para TV es que tienen acceso limitado al sistema de archivos, tal y como se describe en “[Seguridad del sistema de archivos](#)” en la página 144.

Ciclo de vida de la aplicación

A diferencia del entorno de escritorio, el usuario final no puede cerrar la ventana en la que se ejecuta la aplicación de AIR para TV. Por lo tanto, proporcione un mecanismo de interfaz de usuario para salir de la aplicación.

Generalmente, un dispositivo permite que el usuario final salga en cualquier momento de una aplicación con la tecla de salir del mando a distancia. Sin embargo, AIR para TV no distribuye el evento `flash.events.Event.EXITING` en la aplicación. Por lo tanto, guarde el estado de la aplicación con frecuencia de modo que la aplicación se pueda restaurar a un estado razonable cuando se vuelva a iniciar.

Cookies HTTP

AIR para TV admite cookies HTTP persistentes y cookies de sesión. AIR para TV guarda las cookies de cada aplicación de AIR en un directorio específico de la aplicación:

```
/app-storage/<app id>/Local Store
```

El archivo de cookie se llama `cookies`.

Nota: AIR en otros dispositivos, por ejemplo dispositivos de escritorio, no almacena las cookies de forma separada para cada aplicación. El almacenamiento de cookies específicas de cada aplicación admite el modelo de seguridad de aplicación y sistema de AIR para TV.

Utilice la propiedad `URLRequest.manageCookies` de ActionScript del siguiente modo:

- Establezca `manageCookies` en `true`. Este es el valor predeterminado. Significa que AIR para TV añade automáticamente las cookies a las solicitudes HTTP y recuerda las cookies en la respuesta de HTTP.

Nota: aunque `manageCookies` sea `true`, la aplicación puede añadir manualmente una cookie a una solicitud HTTP utilizando `URLRequest.requestHeaders`. Si esta cookie tiene el mismo nombre que una cookie que está administrando AIR para TV, la solicitud contiene dos cookies con el mismo nombre. Los valores de las dos cookies pueden ser diferentes.

- Establezca `manageCookies` en `false`. Este valor significa que la aplicación es responsable de enviar cookies en las solicitudes HTTP y de recordar las cookies en la respuesta HTTP.

Para obtener más información, consulte [URLRequest](#).

Flujo de trabajo para desarrollar una aplicación de AIR para TV

Es posible desarrollar aplicaciones de AIR para TV con las siguientes herramientas de desarrollo de la plataforma de Adobe Flash:

- Adobe Flash Professional
Adobe Flash Professional CS5.5 admite AIR 2.5 para TV, la primera versión de AIR que admite aplicaciones de AIR para TV.
- Adobe Flash® Builder®
Flash Builder 4.5 admite AIR 2.5 para TV.

- SDK de AIR

A partir de AIR 2.5, las aplicaciones se pueden desarrollar utilizando las herramientas de la línea de comandos que se proporcionan con el SDK de AIR. Para descargar el SDK de AIR, consulte <http://www.adobe.com/es/go/products/air/sdk/>.

Uso de Flash Professional

El uso de Flash Professional para desarrollar, probar y publicar aplicaciones de AIR para TV es similar al uso de la herramienta para las aplicaciones de escritorio de AIR.

Sin embargo, al escribir el código de ActionScript 3.0, utilice únicamente clases y métodos que admitan los perfiles de AIR `tv` y `extendedTV`. Para obtener más información, consulte “[Perfiles de dispositivo](#)” en la página 256.

Configuración del proyecto

Realice lo siguiente para configurar el proyecto para una aplicación de AIR para TV:

- En la ficha Flash del cuadro de diálogo Configuración de publicación, defina el valor del reproductor al menos en AIR 2.5.
- En la ficha General del cuadro de diálogo Configuración de Adobe AIR (Configuración de aplicación e instalador), establezca el perfil como `TV` o `TV ampliada`.

Depuración

La aplicación se puede ejecutar utilizando AIR Debug Launcher en Flash Professional. Realice lo siguiente:

- Para ejecutar la aplicación en modo de depuración, seleccione:
Elija Depurar > Depurar película > En AIR Debug Launcher (escritorio).
Una vez realizada esta selección, para las siguientes ejecuciones de depuración, puede seleccionar:
Seleccione Depurar > Depurar película > Depurar.
- Para ejecutar la aplicación sin capacidades del modo de depuración, seleccione:
Seleccione Control > Probar película > En AIR Debug Launcher (escritorio).
Una vez realizada esta selección, puede seleccionar Control > Probar película > Probar para ejecuciones posteriores.

Debido a el perfil de AIR se establece en TV o TV ampliada, AIR Debug Launcher proporciona un menú denominado Remote Control Buttons (Botones de control remoto). Este menú se puede utilizar para simular la pulsación de teclas en un dispositivo de mando a distancia.

Para obtener más información, consulte “[Depuración remota con Flash Professional](#)” en la página 155.

Uso de extensiones nativas

Si su aplicación utiliza una extensión nativa, siga las instrucciones de la sección “[Lista de tareas para utilizar una extensión nativa](#)” en la página 159.

Sin embargo, cuando una aplicación utiliza extensiones nativas:

- No se puede publicar la aplicación utilizando Flash Professional. Para publicar la aplicación, utilice ADT. Consulte “[Empaquetado con ADT](#)” en la página 151.
- No es posible ejecutar o depurar la aplicación con Flash Professional. Para depurar la aplicación en el equipo de desarrollo, utilice ADL. Consulte “[Simulación del dispositivo utilizando ADL](#)” en la página 153.

Uso de Flash Builder

Inicio con Flash Builder 4.5, Flash Builder admite el desarrollo de AIR para TV. El uso de Flash Builder para desarrollar, probar y publicar aplicaciones de AIR para TV es similar al uso de la herramienta para las aplicaciones de escritorio de AIR.

Configuración de la aplicación

Asegúrese de que la aplicación:

- Utiliza el elemento `Application` como clase contenedora en el archivo MXML, si está utilizando un archivo MXML:

```
<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>
```

Importante: las aplicaciones de AIR para TV no admiten el elemento `WindowedApplication`.

Nota: no es necesario utilizar un archivo MXML. En su lugar se puede crear un proyecto de ActionScript 3.0.

- Utilice únicamente clases y métodos de ActionScript 3.0 que admitan los perfiles de AIR `tv` y `extendedTV`. Para obtener más información, consulte “[Perfiles de dispositivo](#)” en la página 256.

Asimismo, en el archivo XML de la aplicación, compruebe lo siguiente:

- El atributo `xmlns` del elemento `application` se establece en AIR 2.5:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```
- El elemento `supportedProfiles` incluye `tv` o `extendedTV`:

```
<supportedProfiles>tv</supportedProfiles>
```

Depuración de la aplicación para

La aplicación se puede ejecutar utilizando AIR Debug Launcher en Flash Builder. Realice lo siguiente:

- 1 Seleccione Ejecutar > Configuraciones de depuración.
- 2 Asegúrese de que el campo Perfil se establece en Escritorio.
- 3 Seleccione Ejecutar > Depurar para realizar la ejecución en modo de depuración, o bien, Ejecutar > Ejecutar para realizar la ejecución sin capacidades del modo de depuración.

Debido a que el elemento `supportedProfiles` se establece en TV o TV ampliada, AIR Debug Launcher incluye un menú denominado Remote Control Buttons (Botones del control remoto). Este menú se puede utilizar para simular la pulsación de teclas en un dispositivo de mando a distancia.

Para obtener más información, consulte “[Depuración remota con Flash Builder](#)” en la página 155.

Uso de extensiones nativas

Si su aplicación utiliza una extensión nativa, siga las instrucciones de la sección “[Lista de tareas para utilizar una extensión nativa](#)” en la página 159.

Sin embargo, cuando una aplicación utiliza extensiones nativas:

- No se puede publicar una aplicación utilizando Flash Builder. Para publicar la aplicación, utilice ADT. Consulte “[Empaquetado con ADT](#)” en la página 151.

- No es posible ejecutar o depurar la aplicación con Flash Builder. Para depurar la aplicación en el equipo de desarrollo, utilice ADL. Consulte “[Simulación del dispositivo utilizando ADL](#)” en la página 153.

Propiedades del descriptor de la aplicación de AIR para TV

Tal y como sucede con otras aplicaciones de AIR, las propiedades básicas de la aplicación se establecen en el archivo descriptor de la aplicación. Las aplicaciones de perfil de TV omiten algunas de las propiedades específicas del escritorio como, por ejemplo, tamaño y transparencia. Las aplicaciones cuyos dispositivos se encuentran en el perfil `extendedTV` pueden utilizar extensiones nativas. Estas aplicaciones identifican la extensiones nativas utilizadas en un elemento `extensions`.

Configuración común

Varias opciones de configuración del descriptor de la aplicación son importantes para todas las aplicaciones de perfil de TV.

Versión necesaria del motor de ejecución de AIR

Especifique la versión del motor de ejecución de AIR que necesita la aplicación utilizando el espacio de nombres del archivo descriptor de la aplicación.

El espacio de nombres, asignado en el elemento `application`, determina, en gran parte, qué funciones puede utilizar la aplicación. Por ejemplo, considere una aplicación que utilice el espacio nombres de AIR 2.5, pero el usuario tiene alguna versión futura instalada. En este caso, la aplicación aún ve el comportamiento de AIR 2.5, aunque el comportamiento sea diferente en la versión futura de AIR. Solo cuando se cambia el espacio de nombres y se publica una actualización, la aplicación tendrá acceso a las nuevas funciones y características. Las soluciones de seguridad son una excepción importante a esta regla.

Especifique el espacio de nombres utilizando el atributo `xmlns` del elemento raíz `application`:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5 es la primera versión de las aplicaciones de AIR que admiten TV.

Identidad de la aplicación

Las distintas configuraciones deben ser exclusivas para cada aplicación que se publique. Estos ajustes incluyen los elementos `id`, `name` y `filename`.

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

Versión de la aplicación

Especifique la versión de la aplicación en el elemento `versionNumber`. Cuando se especifica un valor para `versionNumber`, se debe utilizar una secuencia de hasta tres nombres separados por puntos; por ejemplo, “0.1.2”. Cada segmento del número de versión puede tener hasta tres dígitos. (Es decir, “999.999.999” es el mayor número de versión permitido.) No es necesario incluir los tres segmentos en el número; “1” y “1.0” son también números de la versión legal.

También se puede especificar una etiqueta para la versión utilizando el elemento `versionLabel`. Cuando se añade una etiqueta de la versión, esta se muestra en lugar del número de versión.

```
<versionNumber>1.23.7</versionNumber>  
<versionLabel>1.23 Beta 7</versionLabel>
```

Archivo SWF de la aplicación principal

Especifique el archivo SWF principal de la aplicación en el elemento secundario `versionLabel` del elemento `initialWindow`. Cuando se abordan dispositivos de destino en el perfil tv, se debe utilizar un archivo SWF (las aplicaciones basadas en HTML no se admiten).

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

El archivo se debe incluir en el paquete de AIR (utilizando ADT o su IDE). Simplemente hacer referencia al nombre en el descriptor de la aplicación no hace que el archivo se incluya en el paquete automáticamente.

Propiedades de la pantalla principal

Diversos elementos secundarios del elemento `initialWindow` controlan el comportamiento y la apariencia inicial de la pantalla de la aplicación principal. Aunque la mayoría de estas propiedades se omiten en los dispositivos en los perfiles de TV, se puede utilizar el elemento `fullScreen`:

- **fullScreen**: especifica si la aplicación debe abarcar toda la pantalla del dispositivo o compartir la pantalla con el dispositivo cromático del sistema operativo normal.

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

El elemento visible

El elemento `visible` es un elemento secundario del elemento `initialWindow`. AIR para TV omite el elemento `visible` porque el contenido de la aplicación siempre es visible en los dispositivos de AIR para TV.

Sin embargo, debe definir el elemento `visible` como `true` si la aplicación también está diseñada para dispositivos de escritorio.

En dispositivos de escritorio, el valor predeterminado de este elemento es `false`. Por lo tanto, si no incluye el elemento `visible`, el contenido de la aplicación no será visible en dispositivos de escritorio. Aunque puede utilizar la clase `NativeWindow` de ActionScript para que el contenido sea visible en dispositivos para escritorio, los perfiles del dispositivo para TV no admiten la clase `NativeWindow`. Si intenta utilizar la clase `NativeWindow` en una aplicación que se ejecuta en un dispositivo de AIR para TV, la aplicación no se carga. Da igual que llame al método de la clase `NativeWindow`: una aplicación que utilice la clase no se cargará en ningún dispositivo de AIR para TV.

Perfiles admitidos

Si la aplicación solo se utiliza en un dispositivo de televisión, puede evitar su instalación en otros tipos de dispositivos informáticos. No incluya el resto de perfiles de la lista admitida en el elemento `supportedProfiles`:

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Si una aplicación utiliza una extensión nativa, incluya solamente el perfil `extendedTV` en la lista de perfiles admitidos:

```
<supportedProfiles>extendedTV</supportedProfiles>
```

Si se omite el elemento `supportedProfiles`, se da por sentado que la aplicación admite todos los perfiles.

No incluya *solamente* el perfil `tv` en la lista `supportedProfiles`. Algunos dispositivos de TV siempre ejecutan AIR para TV en un modo correspondiente al perfil `extendedTV`. Este comportamiento es así para que AIR para TV pueda ejecutar aplicaciones con extensiones nativas. Si el elemento `supportedProfiles` especifica solamente `tv`, está declarando que el contenido es incompatible con el modo de AIR para TV de `extendedTV`. Por lo tanto, algunos dispositivos de TV no cargarán una aplicación que especifique solamente el perfil `tv`.

Para obtener una lista de las clases de ActionScript admitidas en los perfiles `tv` y `extendedTV`, consulte [“Capacidades en diferentes perfiles”](#) en la página 257.

Extensiones nativas necesarias

Las aplicaciones que admiten el perfil `extendedTV` pueden utilizar extensiones nativas.

Declare todas las extensiones nativas que utilice la aplicación de AIR en el descriptor de la aplicación con los elementos `extensions` y `extensionID`. El siguiente ejemplo ilustra la sintaxis para especificar dos extensiones nativas necesarias:

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

Si no aparece una extensión, significa que la aplicación no puede usarla.

El elemento `extensionID` tiene el mismo valor que `id` en el archivo descriptor de la extensión. El archivo descriptor de la extensión es un archivo XML denominado `extension.xml`. Se empaqueta en el archivo ANE que se recibe del fabricante del dispositivo.

Si puede ver una extensión en el elemento `extensions`, pero el dispositivo de AIR para TV no tiene dicha extensión instalada, la aplicación no se ejecutará. La excepción de esta regla es si el archivo ANE empaquetado con la aplicación de AIR para TV tiene una versión de la extensión. Si es así, la aplicación se puede ejecutar y utilizará la versión de la extensión. La versión contiene código ActionScript pero no código nativo.

Iconos de la aplicación

Los requisitos para los iconos de la aplicación en los dispositivos de televisión dependen del propio dispositivo. Por ejemplo, el fabricante del dispositivo especifica lo siguiente

- Los iconos necesarios y sus tamaños.
- Los tipos de archivo necesarios y las convenciones de nombres.
- Cómo proporcionar los iconos para la aplicación como, por ejemplo, si empaquetar los iconos con la aplicación.
- Especificar o no los iconos en un elemento `icon` del archivo descriptor de la aplicación.
- Comportamiento si la aplicación no proporciona iconos.

Consulte con el fabricante del dispositivo para obtener más información.

Configuración omitida

Las aplicaciones de los dispositivos de televisión omiten la configuración de la aplicación que se aplica a las funciones del sistema operativo de escritorio, de la ventana nativa y móvil. Los valores de configuración omitidos son:

- `allowBrowserInvocation`
- `aspectRatio`

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

Empaquetado de una aplicación de AIR para TV

Empaquetado con ADT

La herramienta de la línea de comandos ADT de AIR se pueden emplear para empaquetar una aplicación de AIR para TV. A partir del SDK de AIR 2.5, ADT admite el empaquetado para dispositivos TV. Antes de empaquetar, compile todo el código ActionScript y MXML. También se debe disponer de un certificado de firma de código. Se puede crear un certificado con el uso del comando `-certificate` de ADT.

Para obtener una referencia detallada sobre las opciones y los comandos de ADT, consulte “[AIR Developer Tool \(ADT\)](#)” en la página 174.

Creación de un paquete de AIR

Para crear un paquete de AIR, utilice el comando `package` de ADT:

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

En el ejemplo se da por sentado que:

- La ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319.)
- El elemento `codesign.p12` del certificado está en el directorio principal donde se está ejecutando el comando ADT.

Ejecute el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son myApp-app.xml (archivo descriptor de la aplicación), myApp.swf y un directorio de iconos.

Cuando el comando se ejecuta tal y como se muestra, ADT solicitará la contraseña de almacén de claves. No todos los programas de shell muestran los caracteres de la contraseña que se escriben; simplemente presione Intro cuando termine la introducción. Como alternativa, se puede utilizar el parámetro `storepass` para incluir la contraseña en el comando de ADT.

Creación de un paquete de AIRN

Si la aplicación de AIR para TV utiliza una extensión nativa, cree un paquete de AIRN en lugar de un paquete de AIR. Para crear un paquete de AIRN, utilice el comando `package` de ADT, estableciendo el tipo de destino en `airn`.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml  
myApp.swf icons -extdir C:\extensions
```

En el ejemplo se da por sentado que:

- La ruta a la herramienta ADT está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319.)
- El elemento `codesign.p12` del certificado está en el directorio principal donde se está ejecutando el comando ADT.
- El parámetro `-extdir` asigna un nombre a un directorio que incluye los archivos ANE que utiliza la aplicación.

Estos archivos ANE contienen una versión del simulador o código auxiliar de solo de la extensión de ActionScript. La versión de la extensión de que contiene el código nativo se instala en el dispositivo de AIR para TV.

Ejecute el comando desde el directorio que contiene los archivos de la aplicación. Los archivos de la aplicación del ejemplo son myApp-app.xml (archivo descriptor de la aplicación), myApp.swf y un directorio de iconos.

Cuando el comando se ejecuta tal y como se muestra, ADT solicitará la contraseña de almacén de claves. No todos los programas de shell muestran los caracteres de la contraseña que se escriben; simplemente presione Intro cuando termine la introducción. Como alternativa, puede utilizar el parámetro `storepass` para incluir la contraseña en el comando.

También puede crear un archivo AIRI para una aplicación de AIR para TV que utilice extensiones nativas. El archivo AIR es como el archivo AIRN, excepto en que no está firmado. Por ejemplo:

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Posteriormente se puede crear un archivo AIRN a partir del archivo AIRI cuando esté listo para firmar la aplicación:

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Para obtener más información, consulte [Desarrollo de extensiones nativas para Adobe AIR](#).

Empaquetado con Flash Builder o Flash Professional

Flash Professional y Flash Builder permiten publicar o exportar los paquetes de AIR sin que tenga que ejecutar ADT por sí mismo. El procedimiento para crear un paquete de AIR para una aplicación de AIR se analiza en la documentación de estos programas.

Actualmente, sin embargo, solo ADT puede crear paquetes AIRN, los paquetes de aplicación para aplicaciones de AIR para TV que utilizan extensiones nativas.

Para más información, consulte las referencias siguientes:

- [Empaquetado de aplicaciones de AIR con Flash Builder](#)
- [Publicación para Adobe AIR con Flash Professional](#)

Depuración de aplicaciones de AIR para TV

Simulación del dispositivo utilizando ADL

La forma más rápida y sencilla de probar y depurar la mayoría de las funciones de la aplicación consiste en ejecutar la aplicación en el equipo de desarrollo mediante la utilidad Adobe Debug Launcher (ADL).

ADL utiliza el elemento `supportedProfiles` del descriptor de la aplicación para seleccionar qué perfil utilizar. Concretamente:

- Si se incluyen varios perfiles, ADL utiliza el primero de la lista.
- El parámetro `-profile` de ADL también se puede utilizar para seleccionar uno de los demás perfiles de la lista `supportedProfiles`.
- (Si no se incluye ningún elemento `supportedProfiles` en el descriptor de la aplicación, se puede especificar cualquier perfil para el argumento `-profile`.)

Por ejemplo, utilice el siguiente comando para iniciar una aplicación que simule el perfil de `tv`:

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

Al simular el perfil `tv` o `extendedTV` en el escritorio con ADL, la aplicación se ejecuta en un entorno que coincide más con un dispositivo de destino. Por ejemplo:

- Las APIs de ActionScript que no forman parte del perfil en el argumento `-profile` no están disponibles.
- ADL permite la introducción de controles de entrada de dispositivo como, por ejemplo, controles remotos, mediante los comandos de menú.
- La especificación de `tv` o `extendedTV` en el argumento `-profile` permite que ADL simule la clase `StageVideo` en el escritorio.
- La especificación de `extendedTV` en el argumento `-profile` permite que la aplicación utilice simuladores o códigos auxiliares de la extensión nativa con el archivo `AIRN` de la aplicación.

Sin embargo, debido a que ADL ejecuta la aplicación en el escritorio, la prueba de las aplicaciones de AIR para TV utilizando ADL tiene limitaciones:

- No refleja el rendimiento de la aplicación en el dispositivo. Realice pruebas de rendimiento en el dispositivo de destino.
- No simula las limitaciones del objeto `StageVideo`. Generalmente, se utiliza la clase `StageVideo`, no `Video`, para reproducir un vídeo al abordar dispositivos de AIR para TV. La clase `StageVideo` aprovecha las ventajas de rendimiento del hardware del dispositivo, pero presenta limitaciones de visualización. ADL reproduce el vídeo en el escritorio sin estas limitaciones. Por lo tanto, compruebe la reproducción de vídeo en el dispositivo de destino.
- No puede simular el código nativo de una extensión nativa. Sin embargo, puede especificar el perfil `extendedTV`, que admite las extensiones nativas, en el argumento `-profile` de ADL. ADL permite realizar las comprobaciones con la versión del simulador o de código auxiliar de solo de la extensión de ActionScript incluida en el paquete ANE. Sin embargo, generalmente la extensión de correspondiente que está instalada en el dispositivo también incluye código nativo. Para realizar la comprobación utilizando la extensión de con su código nativo, ejecute la aplicación en el dispositivo de destino.

Para obtener más información, consulte “[AIR Debug Launcher \(ADL\)](#)” en la página 168.

Uso de extensiones nativas

Si la aplicación utiliza extensiones nativas, el comando ADL tiene el aspecto del siguiente ejemplo:

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

En el ejemplo se da por sentado que:

- La ruta a la herramienta ADL está en la definición de la ruta del shell de la línea de comandos. (Consulte “[Variables del entorno de ruta](#)” en la página 319.)
- El directorio actual contiene los archivos de la aplicación. Estos archivos incluyen los archivos SWF y el archivo descriptor de la aplicación, que es myApp-app.xml en este ejemplo.
- El parámetro `-extdir` nombra a un directorio que contiene otro directorio para cada extensión nativa que utiliza la aplicación. Cada uno de estos directorios contienen el archivo ANE *unpacked* de una extensión nativa. Por ejemplo:

```
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
```

Estos archivos ANE sin empaquetar contienen versiones de simulador de la extensión (solo de ActionScript). La versión de la extensión de que contiene el código nativo se instala en el dispositivo de AIR para TV.

Para obtener más información, consulte [Desarrollo de extensiones nativas para Adobe AIR](#).

Entrada de control

ADL simula los botones de control remoto en un dispositivo de TV. Puede enviar estas introducciones con botón al dispositivo simulado utilizando el menú que se muestra cuando ADL se inicia utilizando uno de los perfiles de TV.

Tamaño de la pantalla

La aplicación se puede probar en diferentes tamaños de pantalla, estableciendo el parámetro `-screensize` de ADL. Se puede especificar una cadena que contenga cuatro valores que representen las anchuras y alturas de las pantallas normales y ampliadas.

Especifique siempre las dimensiones de píxel de la visualización horizontal, esto es, especifique la anchura como un valor más pequeño que el valor de la altura. Por ejemplo:

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

Sentencias Trace

Cuando se ejecuta la aplicación de TV en el escritorio, la salida de la sentencia trace se imprime en el depurador o la ventana de terminal para iniciar ADL.

Depuración remota con Flash Professional

Flash Professional se puede emplear para depurar de forma remota la aplicación de AIR para TV mientras que se ejecuta en el dispositivo de destino. Sin embargo, los pasos para configurar la depuración remota dependen del dispositivo. Por ejemplo, el kit de desarrollo de hardware de the Adobe® AIR® para TV MAX 2010 contiene documentación de los pasos detallados para ese dispositivo.

Independientemente del dispositivo de destino, realice los siguientes pasos para preparar la depuración remota:

- 1 En el cuadro de diálogo Configuración de publicación, en la ficha Flash, seleccione Permitir depuración.
Esta opción hace que Flash Professional incluya información de depuración en todos los archivos SWF que crea a partir del archivo FLA.
- 2 En la ficha Firma del cuadro de diálogo Configuración de Adobe AIR (Configuración de aplicación e instalador), seleccione la opción para preparar un archivo AIR Intermediate (AIRI).
Mientras se sigue desarrollando la aplicación, utilizar un archivo AIRI (que no necesita firma digital) es suficiente.
- 3 Publique su aplicación, creando el archivo de AIRI.

Los últimos pasos son la instalación y la ejecución de la aplicación en el dispositivo de destino. Sin embargo, estos pasos dependen del dispositivo.

Depuración remota con Flash Builder

Flash Builder se puede emplear para depurar de forma remota la aplicación de AIR para TV mientras que se ejecuta en el dispositivo de destino. Sin embargo, los pasos para configurar la depuración remota dependen del dispositivo.

Independientemente del dispositivo de destino, realice los siguientes pasos para preparar la depuración remota:

- 1 Seleccione Project (Proyecto) > Export Release Version (Exportar versión oficial). Seleccione la opción para preparar un archivo de AIR Intermediate (AIRI).
Mientras se sigue desarrollando la aplicación, utilizar un archivo AIRI (que no necesita firma digital) es suficiente.
- 2 Publique su aplicación, creando el archivo de AIRI.
- 3 Cambie el paquete de AIRI de la aplicación para que contenga archivos SWF que incluyan información de depuración.

Los archivos SWF que contienen información de depuración se sitúan en el directorio de proyecto de Flash Builder para la aplicación en un directorio denominado bin-debug. Reemplace estos archivos SWF en el paquete de AIRI con los archivos SWF en el directorio bin-debug.

En un equipo de desarrollo de Windows, este reemplazo se puede llevar a cabo realizando lo siguiente:

- 1 Cambie el nombre del archivo del paquete de AIRI para disponer de la extensión de nombre de archivo .zip en lugar de .airi.
- 2 Extraiga el contenido del archivo ZIP.
- 3 Reemplace los archivos SWF de la estructura del directorio extraído por los de bin-debug.
- 4 Vuelva a comprimir los archivos en el directorio extraído.
- 5 Cambie el archivo comprimido para disponer una vez más de la extensión de nombre de archivo .airi.

Si está utilizando un equipo de desarrollo Mac, los pasos para este reemplazo dependen del dispositivo. Sin embargo, suelen implicar lo siguiente:

- 1 Instale el paquete de AIRI en el dispositivo de destino.
- 2 Reemplace los archivos SWF en el directorio de instalación de la aplicación del dispositivo de destino con los archivos SWF del directorio bin-debug.

Por ejemplo, considere el dispositivo incluido con el kit de desarrollo de hardware de Adobe AIR para TV MAX 2010. Instale el paquete de AIRI tal y como se describe la documentación del kit. A continuación, utilice telnet en la línea de comandos del equipo de desarrollo de Mac para acceder al dispositivo de destino. Sustituya los archivos SWF en el directorio de instalación de la aplicación en `/opt/adobe/aircraft/apps/<nombre de la aplicación>/` con los archivos SWF del directorio bin-debug.

Los siguientes pasos se destinan a la depuración remota con Flash Builder y el dispositivo incluido en el kit de desarrollo de hardware de Adobe AIR para TV MAX 2010.

- 1 En el ordenador que ejecuta Flash Builder, el equipo de desarrollo, ejecute el conector de dispositivo (Device Connector) de AIR para TV que acompaña al kit de desarrollo de hardware de MAX 2010. Muestra la dirección IP del equipo de desarrollo.
- 2 En el dispositivo del kit de hardware, inicie la aplicación DevMaster, que también se incluye con el kit de desarrollo.
- 3 En la aplicación DevMaster, indique la dirección IP del equipo de desarrollo tal y como se muestra en el conector del dispositivo de AIR for TV.
- 4 En la aplicación DevMaster, compruebe que la opción Habilitar depuración remota esté seleccionada.
- 5 Salga de la aplicación DevMaster.
- 6 En el equipo de desarrollo, seleccione Start (Inicio) en el conector de AIR para TV.
- 7 En el dispositivo del kit de hardware, inicie otra aplicación. Compruebe que la información de seguimiento se muestra en el conector del dispositivo de AIR para TV.

Si no se muestra la información de seguimiento, el equipo de desarrollo y el dispositivo del kit de hardware no están conectados. Asegúrese de que el puerto del equipo de desarrollo que se utiliza para la información de seguimiento esté disponible. Se pueden seleccionar distintos puertos en el conector de dispositivo de AIR para TV. Asimismo, compruebe que el firewall acceda al puerto seleccionado.

A continuación, inicie el depurador en Flash Builder. Realice lo siguiente:

- 1 En Flash Builder, seleccione Run (Ejecutar) > Debug Configurations (Configuraciones de depuración).
- 2 En la configuración de depuración existente, que es para la depuración local, copie el nombre del proyecto.
- 3 En el cuadro de diálogo Debug Configurations (Configuraciones de depuración), seleccione Web Application (Aplicación web). A continuación, seleccione el icono New Launch Configuration (Nuevo configuración de inicio).
- 4 Pegue el nombre del proyecto en el campo Project (Proyecto).
- 5 En la sección URL Or Path To Launch (URL o ruta para iniciar), anule la selección de Use Default (Usar predeterminado). Asimismo, indique `about:blank` en el campo de texto.
- 6 Seleccione Aplicar para guardar los cambios.
- 7 Seleccione Depurar para iniciar el depurador de Flash Builder.
- 8 Inicie la aplicación el dispositivo del kit de hardware.

Ahora podrá utilizar el depurador de Flash Builder para, por ejemplo, definir los puntos de corte y examinar las variables.

Capítulo 9: Uso de extensiones nativas para Adobe AIR

Las extensiones nativas para Adobe AIR proporcionan API de ActionScript que permiten acceder a funciones específicas del dispositivo programadas con código nativo. Unas veces, los desarrolladores de extensiones nativas a veces trabajan con los fabricantes de los dispositivos; otras veces son desarrolladores de terceros.

Si está desarrollando una extensión nativa, consulte [Desarrollo de extensiones nativas para Adobe AIR](#).

Una extensión nativa es una combinación de:

- Clases de ActionScript.
- Código nativo.

No obstante, como desarrollador de una aplicación de AIR con extensión nativa, su trabajo se limitará a las clases de ActionScript.

Las extensiones nativas resultan útiles en las situaciones siguientes:

- La implementación del código nativo permite acceder a funciones específicas de la plataforma. Estas funciones específicas de la plataforma no están disponibles en las clases incluidas en ActionScript y no es posible implementarlas en clases de ActionScript específicas de la aplicación. La implementación del código nativo sí ofrece esta posibilidad, ya que tiene acceso al software y al hardware específico del dispositivo.
- Una implementación del código nativo a veces puede ser más rápida que una implementación que solamente utilice ActionScript.
- La implementación del código nativo puede proporcionar acceso a ActionScript al código nativo heredado.

Algunos ejemplos de extensiones nativas están disponibles en el Centro de desarrollo de Adobe. Por ejemplo, una extensión nativa permite que las aplicaciones de AIR accedan a la función de vibración de Android. Consulte [Extensiones nativas para Adobe AIR](#).

Archivos de extensión nativa de AIR (ANE)

Los desarrolladores de extensiones nativas empaquetan una extensión nativa en un archivo ANE. Un archivo ANE es un archivador que contiene las bibliotecas y los recursos necesarios para la extensión nativa.

En algunos dispositivos, el archivo ANE contiene la biblioteca del código nativo que utiliza la extensión nativa. En otros, la biblioteca del código nativo se instala en el dispositivo. En algunos casos, la extensión nativa no tiene ningún código nativo para un dispositivo concreto; se implementa solamente con ActionScript.

Como desarrollador de una aplicación de AIR, utilizará el archivo ANE del modo siguiente:

- Incluya el archivo ANE en la ruta de la biblioteca de la aplicación del mismo modo que incluiría un archivo SWC en la ruta de acceso de la biblioteca. Esta acción permite a la aplicación hacer referencia a las clases de ActionScript de la extensión.

Nota: cuando compile la aplicación, asegúrese de utilizar la vinculación dinámica para el archivo ANE. Si utiliza *Flash Builder*, especifique *External* en el panel de propiedades de rutas de acceso de *ActionScript Builder*; si utiliza la línea de comandos, especifique *-external-library-path*.

- Empaquete el archivo ANE con la aplicación de AIR.

Extensiones nativas y la clase NativeProcess de ActionScript

ActionScript 3.0 proporciona una clase NativeProcess. Esta clase permite que una aplicación de AIR ejecute procesos nativos en el sistema operativo del host. Esta función es parecida a las extensiones nativas, que proporcionan acceso a funciones y bibliotecas específicas de la plataforma. Si decide utilizar la clase NativeProcess y no una extensión nativa, tenga en cuenta lo siguiente:

- El perfil de AIR `extendedDesktop` es el único que admite la clase NativeProcess. Por lo tanto, para las aplicaciones con perfiles de AIR `mobileDevice` y `extendedMobileDevice`, las extensiones nativas son la única opción.
- Los desarrolladores de extensiones nativas a menudo proporcionan implementaciones nativas para varias plataformas, pero la API de ActionScript que se incluye suele ser la misma para todas. Si utiliza la clase NativeProcess, el código ActionScript para iniciar el proceso nativo puede variar en función de la plataforma.
- La clase NativeProcess inicia un proceso independiente, mientras que una extensión nativa se ejecuta en el mismo proceso que la aplicación de AIR. Por ello, si le preocupa que el código pueda bloquearse, utilizar la clase NativeProcess es lo más fiable. Sin embargo, un proceso independiente implica que posiblemente deba implementar gestión de comunicaciones entre procesos.

Extensiones nativas y clases de biblioteca de ActionScript (archivos SWC)

Un archivo SWC es una biblioteca de clases de ActionScript en formato archivador. El archivo SWC contiene un archivo SWF así como otros archivos de recursos. El archivo SWC es una forma cómoda de compartir clases de ActionScript sin tener que compartir el código ActionScript individual y los archivos de recursos.

Un paquete de extensiones nativas es un archivo ANE. Al igual que un archivo SWC, un archivo ANE también es una biblioteca de clases de ActionScript que contiene un archivo SWF y otros archivos de recursos en formato archivador. No obstante, la diferencia más importante entre un archivo ANE y uno SWC reside en que solamente el archivo ANE puede contener una biblioteca del código nativo.

***Nota:** cuando compile la aplicación, asegúrese de utilizar la vinculación dinámica para el archivo ANE. Si utiliza Flash Builder, especifique External en el panel de propiedades de rutas de acceso de ActionScript Builder; si utiliza la línea de comandos, especifique `-external-library-path`.*

Más temas de ayuda

[archivos SWC, información](#)

Dispositivos admitidos

Desde AIR 3, se pueden utilizar extensiones nativas en aplicaciones para los siguientes dispositivos:

- Dispositivos Android, a partir de Android 2.2

- Dispositivos iOS, a partir de iOS 4.0
- Simulador de iOS, a partir de AIR 3.3
- Blackberry PlayBook
- Dispositivos Windows de escritorio que admitan AIR 3.0
- Dispositivos Mac OS X de escritorio que admitan AIR 3.0

A menudo, la misma extensión nativa está destinada a varias plataformas. El archivo ANE de la extensión contiene bibliotecas de ActionScript y nativas para cada plataforma admitida. Normalmente, las bibliotecas de ActionScript tienen las mismas interfaces públicas para todas las plataformas. Las bibliotecas nativas son necesariamente distintas.

A veces, una extensión nativa admite una plataforma en concreto. La implementación predeterminada de la plataforma solo tiene código ActionScript, no código nativo. Si empaqueta una aplicación para una plataforma no admitida específicamente por la extensión, la aplicación utilizará la implementación predeterminada al ejecutarla. Por ejemplo, consideremos una extensión que proporciona una función que se aplica solamente a dispositivos móviles. La extensión también puede proporcionar una implementación predeterminada que la aplicación de escritorio puede usar para simular la función.

Perfiles de dispositivos admitidos

Los siguientes perfiles de AIR admiten extensiones nativas:

- `extendedDesktop`, a partir de AIR 3.0
- `mobileDevice`, a partir de AIR 3.0
- `extendedMobileDevice`, a partir de AIR 3.0

Más temas de ayuda

[Compatibilidad con perfil de AIR](#)

Lista de tareas para utilizar una extensión nativa

Para utilizar una extensión nativa en la aplicación, lleve a cabo las siguientes tareas:

- 1 Declare la extensión en el archivo descriptor de la aplicación.
- 2 Incluya el archivo ANE en la ruta de la biblioteca de la aplicación.
- 3 Empaquete la aplicación.

Declaración de la extensión en el archivo descriptor de la aplicación

Todas las aplicaciones de AIR tienen un archivo descriptor de la aplicación. Cuando una aplicación utiliza una extensión nativa, el archivo descriptor de la aplicación incluye un elemento `<extensions>`. Por ejemplo:

```
<extensions>
  <extensionID>com.example.Extension1</extensionID>
  <extensionID>com.example.Extension2</extensionID>
</extensions>
```

El elemento `extensionID` tiene el mismo valor que `id` en el archivo descriptor de la extensión. El archivo descriptor de la extensión es un archivo XML denominado `extension.xml`. Se empaqueta en el archivo ANE. Puede utilizar una herramienta extractora de archivos para buscar el archivo `extension.xml`.

Inclusión del archivo ANE en la ruta de la biblioteca de la aplicación

Para compilar una aplicación que utilice una extensión nativa, incluya el archivo ANE en la ruta de la biblioteca.

Uso del archivo ANE con Flash Builder

Si la aplicación utiliza una extensión nativa, incluya el archivo ANE para la extensión nativa en la ruta de biblioteca. Flash Builder se puede utilizar para compilar el código de ActionScript.

Siga los pasos descritos a continuación que utilizan Flash Builder 4.5.1:

- 1 Cambie la extensión del nombre de archivo del archivo ANE de `.ane` a `.swc`. Este paso es necesario para que Flash Builder pueda encontrar el archivo.
- 2 Seleccione Proyecto > Propiedades en el proyecto de Flash Builder.
- 3 Seleccione la ruta de compilación de Flex en el cuadro de diálogo Propiedades.
- 4 En la ficha Ruta de biblioteca, seleccione Añadir SWC...
- 5 Busque el archivo SWC y seleccione Abrir.
- 6 Seleccione Aceptar en el cuadro de diálogo Añadir SWC...

El archivo ANE aparece en la ficha Ruta de biblioteca en el cuadro de diálogo Propiedades.

- 7 Amplíe la entrada del archivo SWC. Haga doble clic en el Tipo de vínculo para abrir el cuadro de diálogo Opciones de elemento de ruta de biblioteca.
- 8 En el cuadro de diálogo Opciones de elemento de ruta de biblioteca, cambie el Tipo de vínculo a Externo.

Ahora podrá compilar la aplicación utilizando, por ejemplo, Proyecto > Crear proyecto.

Uso del archivo ANE con Flash Professional

Si la aplicación utiliza una extensión nativa, incluya el archivo ANE para la extensión nativa en la ruta de biblioteca. Posteriormente, Flash Professional CS5.5 se puede utilizar para compilar el código de ActionScript. Realice lo siguiente:

- 1 Cambie la extensión del nombre de archivo del archivo ANE de `.ane` a `.swc`. Este paso es necesario para que Flash Professional pueda encontrar el archivo.
- 2 Seleccione Archivo > Configuración de ActionScript en su archivo FLA.
- 3 Seleccione la ficha Ruta de biblioteca en el cuadro de diálogo Configuración avanzada de ActionScript 3.0.
- 4 Seleccione Navegar hasta el archivo SWC.
- 5 Busque el archivo SWC y seleccione Abrir.

El archivo SWC aparece ahora en la ficha Ruta de biblioteca en el cuadro de diálogo Configuración avanzada de ActionScript 3.0.

6 Con el archivo SWC seleccionado, seleccione el botón Seleccionar opciones de vinculación para una biblioteca.

7 En el cuadro de diálogo Opciones de elemento de ruta de biblioteca, cambie el Tipo de vínculo a Externo.

Empaquetado de una aplicación con extensiones nativas

Utilice ADT para empaquetar una aplicación con extensiones nativas. No es posible empaquetar la aplicación con Flash Professional CS5.5 o Flash Builder 4.5.1.

Para obtener más información sobre ADT, consulte [AIR Developer Tool \(ADT\)](#).

Por ejemplo, el siguiente comando ADT crea un archivo DMG (un archivo de instalación nativo para Mac OS X) para una aplicación que utiliza extensiones nativas:

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

El siguiente comando crea un paquete APK para un dispositivo Android:

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

El siguiente comando crea un paquete iOS para una aplicación de iPhone:

```
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
```

Debe tener en cuenta lo siguiente:

- Utilice un tipo de paquete de archivo de instalación nativo.
- Especifique el directorio de la extensión.
- Asegúrese de que el dispositivo de destino de la aplicación admite el archivo ANE.

Uso de un tipo de paquete de archivo de instalación nativo

El paquete de la aplicación debe ser un archivo de instalación nativo. No es posible crear un paquete AIR multiplataforma (un paquete .air) para una aplicación que utilice una extensión nativa, ya que las extensiones nativas suelen contener código nativo. Sin embargo, lo habitual es que una extensión nativa admita varias plataformas nativas con las mismas API de ActionScript. En estos casos, puede utilizarse el mismo archivo ANE en distintos paquetes de archivo de instalación nativos.

La siguiente tabla resume el valor que debe usarse en la opción `-target` del comando ADT:

Plataforma de destino de la aplicación	-target
Dispositivos de escritorio Mac OS X o Windows	-target native -target bundle
Android	-target apk u otros destinos de paquetes de Android
iOS	-target ipa-ad-hoc u otros destinos de paquetes de iOS
Simulador de iOS	-target ipa-test-interpretor-simulator -target ipa-debug-interpretor-simulator

Definición del directorio de la extensión

Utilice la opción ADT `-extdir` para informar a ADT del directorio que contiene las extensiones nativas (archivos ANE).

Para obtener más información sobre esta opción, consulte “[Opciones de ruta y archivo](#)” en la página 191.

Asegúrese de que el dispositivo de destino de la aplicación admite el archivo ANE

Cuando se proporciona un archivo ANE, el desarrollador de extensiones nativas informa sobre las plataformas admitidas por la extensión. También puede utilizar una herramienta de extracción de archivos para consultar el contenido del archivo ANE. Los archivos extraídos incluyen un directorio para cada plataforma admitida.

Saber qué plataforma admite la extensión es importante a la hora de empaquetar la aplicación que utilice el archivo ANE. Tenga en cuenta las siguientes reglas:

- Para crear un paquete de aplicación para Android, el archivo ANE debe incluir la plataforma `Android-ARM`. Como alternativa, el archivo ANE debe incluir la plataforma predeterminada y, al menos, otra plataforma más.
- Para crear un paquete de aplicación para iOS, el archivo ANE debe incluir la plataforma `iPhone-ARM`. Como alternativa, el archivo ANE debe incluir la plataforma predeterminada y, al menos, otra plataforma más.
- Para crear un paquete de aplicación para el simulador de iOS, el archivo ANE debe incluir la plataforma `iPhone-x86`.
- Para crear un paquete de aplicación para Mac OS X, el archivo ANE debe incluir la plataforma `MacOS-x86`. Como alternativa, el archivo ANE debe incluir la plataforma predeterminada y, al menos, otra plataforma más.
- Para crear un paquete de aplicación para Windows, el archivo ANE debe incluir la plataforma `Windows-x86`. Como alternativa, el archivo ANE debe incluir la plataforma predeterminada y, al menos, otra plataforma más.

Capítulo 10: Compiladores de ActionScript

Antes de que el código ActionScript y MXML se pueda incluir en una aplicación de AIR, se debe compilar. Si se utiliza un entorno de desarrollo integrado (Integrated Development Environment, IDE) como, por ejemplo, Adobe Flash Builder o Adobe Flash Professional, el IDE administra la compilación en un segundo plano. Sin embargo, también se pueden invocar los compiladores de ActionScript desde la línea de comandos para crear archivos SWF cuando no se utiliza un IDE o cuando se emplea un script de creación.

Información sobre las herramientas de la línea de comandos de AIR en el SDK de Flex

Cada una de las herramientas de la línea de comandos que se utiliza para crear una aplicación de Adobe AIR llama a la herramienta correspondiente utilizada para crear aplicaciones de

- amxmlc llama a mxmxmlc para compilar clase de la aplicación.
- acompc llama a compc para compilar clases de componente y biblioteca.
- aasdoc llama a asdoc para generar archivos de documentación a partir de comentarios de código de origen.

La única diferencia entre las versiones de las utilidades de Flex y AIR radica en que las versiones de AIR cargan opciones de configuración desde el archivo air-config.xml en lugar del archivo flex-config.xml.

Las herramientas del SDK de Flex y las opciones de la línea de comandos se describen detalladamente en la [documentación de Flex](#) (en inglés). Las herramientas del SDK de Flex se describen aquí en un nivel básico como ayuda en su introducción y para destacar las diferencias existentes entre la creación de aplicaciones de Flex y la creación de aplicaciones de AIR.

Más temas de ayuda

“[Creación de la primera aplicación de AIR de escritorio con el SDK de Flex](#)” en la página 40

Configuración del compilador

Generalmente se especifican opciones de compilación tanto en la línea de comandos como con uno o varios archivos de configuración. El archivo de configuración del SDK de Flex global contiene valores predeterminados que se utilizan siempre que se ejecutan los compiladores. Este archivo se puede editar para adaptarse a su entorno de desarrollo. Existen dos archivos de configuración de Flex globales ubicados en el directorio frameworks de la instalación del SDK de Flex. El archivo air-config.xml se usa cuando se ejecuta el compilador amxmlc. Este archivo configura el compilador para AIR incluyendo las bibliotecas de AIR. El archivo flex-config.xml se utiliza cuando se ejecuta mxmxmlc.

Los valores de configuración predeterminados resultan adecuados para detectar el modo de funcionamiento de Flex y AIR, pero cuando lleve a cabo proyectos de envergadura, analice más detalladamente las opciones disponibles. Se pueden proporcionar valores específicos del proyecto para las opciones del compilador en un archivo de configuración local que tenga prioridad sobre los valores globales para un proyecto determinado.

Nota: no se utilizan opciones de compilación específicamente para aplicaciones de AIR, pero se debe hacer referencia a las bibliotecas de AIR al compilar una aplicación de AIR. A estas bibliotecas se les suele hacer referencia en un archivo de configuración de nivel de proyecto, en un archivo para una herramienta de creación como Ant o directamente en la línea de comandos.

Compilación de archivos de origen MXML y ActionScript para AIR

Puede compilar los componentes Adobe® ActionScript® 3.0 y MXML de su aplicación de AIR con el compilador de la línea de comandos MXML (amxmlc). (No es necesario compilar aplicaciones basadas en HTML. Para compilar un archivo SWF en Flash Professional, simplemente publique la película en un archivo SWF.)

El patrón de línea de comandos básico para utilizar amxmlc es:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

siendo *[compiler options]* las opciones de la línea de comandos que se utilizan para compilar la aplicación de AIR.

El comando amxmlc invoca al compilador de Flex estándar mxmcl con un parámetro adicional, `+configname=air`. Este parámetro indica al compilador que utilice el archivo `air-config.xml` en lugar de `flex-config.xml`. Por lo demás, el uso de amxmlc es idéntico al uso de mxmcl.

El compilador carga el archivo de configuración `air-config.xml` especificando las bibliotecas de AIR y Flex que se suelen necesitar al compilar una aplicación de AIR. También se puede utilizar un archivo de configuración local a nivel de proyecto para suprimir o añadir opciones adicionales a la configuración global. En general la forma más fácil de crear un archivo de configuración local es mediante modificación de una copia de la versión global. El archivo local puede cargarse con la opción `-load-config`:

-load-config=project-config.xml Suprime las opciones globales.

-load-config+=project-config.xml Añade valores adicionales a las opciones globales que aceptan más de un valor, como la opción `-library-path`. Las opciones globales que tienen un solo valor se suprimen.

Si se utiliza una convención particular para el nombre del archivo de configuración local, el compilador amxmlc carga el archivo local automáticamente. Por ejemplo, si el archivo MXML principal es `RunningMan.mxml`, el nombre del archivo de configuración local es: `RunningMan-config.xml`. Para compilar la aplicación solo hace falta escribir:

```
amxmlc RunningMan.mxml
```

`RunningMan-config.xml` se carga automáticamente, dado que su nombre de archivo coincide con el del archivo MXML compilado.

Ejemplos con amxmlc

Los siguientes ejemplos demuestran el uso del compilador amxmlc. (Solo se necesitan compilar los componentes ActionScript y MXML de la aplicación).

Compile un archivo MXML de AIR:

```
amxmlc myApp.mxml
```

Compile y defina el nombre de salida:

```
amxmlc -output anApp.swf -- myApp.mxml
```

Compile un archivo ActionScript de AIR:

```
amxmlc myApp.as
```

Especifique un archivo de configuración para el compilador:

```
amxmlc -load-config config.xml -- myApp.mxml
```

Añada opciones adicionales de otro archivo de configuración:

```
amxmlc -load-config+=moreConfig.xml -- myApp.mxml
```

Añada bibliotecas en la línea de comandos (además de las bibliotecas que ya figuran en el archivo de configuración):

```
amxmlc -library-path+=/libs/libOne.swc,/libs/libTwo.swc -- myApp.mxml
```

Compile un archivo MXML de AIR sin usar archivo de configuración (Win):

```
mxmhc -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
```

Compile un archivo MXML de AIR sin usar archivo de configuración (Mac OS X o Linux):

```
mxmhc -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
```

Compile un archivo MXML de AIR para utilizar una biblioteca compartida con el motor de ejecución:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --  
myApp.mxml
```

Compile un archivo MXML de AIR para usar un archivo ANE (asegúrese de utilizar `-external-library-path` para el archivo ANE):

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

Compilando desde Java (con la ruta de clase definida para que incluya `mxmhc.jar`):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional  
compiler options] -- myApp.mxml
```

La opción `flexlib` identifica la ubicación del directorio `frameworks` del SDK de Flex, lo cual permite al compilador localizar el archivo `flex_config.xml`.

Compilando desde Java (sin ruta de clase definida):

```
java -jar [Flex SDK 2]/lib/mxmhc.jar +flexlib [Flex SDK 3]/frameworks +configname=air  
[additional compiler options] -- myApp.mxml
```

Para invocar el compilador utilizando Apache Ant (en el ejemplo se utiliza una tarea de Java para ejecutar `mxmhc.jar`):

```
<property name="SDK_HOME" value="C:/Flex46SDK"/>
<property name="MAIN_SOURCE_FILE" value="src/myApp.mxml"/>
<property name="DEBUG" value="true"/>
<target name="compile">
  <java jar="${MXMLC.JAR}" fork="true" failonerror="true">
    <arg value="-debug=${DEBUG}"/>
    <arg value="+flexlib=${SDK_HOME}/frameworks"/>
    <arg value="+configname=air"/>
    <arg value="-file-specs=${MAIN_SOURCE_FILE}"/>
  </java>
</target>
```

Compilación de una biblioteca de código o componente de AIR (Flex)

Utilice el compilador de componentes, `acompc`, para compilar bibliotecas de AIR y componentes independientes. El compilador de componentes `acompc` se comporta como el compilador `amxmlc` pero con las siguientes excepciones:

- Hay que especificar qué clases de las que figuran en el código base se han de incluir en la biblioteca o el componente.
- El compilador `acompc` no busca automáticamente un archivo de configuración local. Para utilizar un archivo de configuración de un proyecto, hay que utilizar primero la opción `-load-config`.

El comando `acompc` invoca el compilador de componentes estándar de Flex, `compc`, pero carga las opciones de configuración del archivo `air-config.xml` en lugar de las del archivo `flex-config.xml`.

Archivo de configuración del compilador de componentes

Utilice un archivo de configuración local para evitar tener que escribir (quizá con errores tipográficos) la ruta de origen y los nombres de las clases en la línea de comandos. Añada la opción `-load-config` a la línea de comandos de `acompc` para cargar el archivo de configuración local.

El siguiente ejemplo ilustra una configuración para crear una biblioteca con dos clases, `ParticleManager` y `Particle`, ambos en el paquete `com.adobe.samples.particles`. Los archivos de clase se encuentran en la carpeta `source/com/adobe/samples/particles`.

```
<flex-config>
  <compiler>
    <source-path>
      <path-element>source</path-element>
    </source-path>
  </compiler>
  <include-classes>
    <class>com.adobe.samples.particles.ParticleManager</class>
    <class>com.adobe.samples.particles.Particle</class>
  </include-classes>
</flex-config>
```

Para compilar la biblioteca con el archivo de configuración, denominado `ParticleLib-config.xml`, escriba:

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Para ejecutar el mismo comando con la totalidad en la línea de comandos, escriba:

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle  
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Escriba todo el comando en una sola línea o utilice el carácter de continuación de línea para el shell de comandos).

Ejemplos con acompc

Estos ejemplos dan por sentado que utiliza un archivo de configuración denominado `myLib-config.xml`.

Compile un componente o una biblioteca de AIR:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Compile una biblioteca compartida en tiempo de ejecución

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Obsérvese que la carpeta `lib` debe existir y estar vacía antes de ejecutar el comando).

Capítulo 11: AIR Debug Launcher (ADL)

Utilice AIR Debug Launcher (ADL) para ejecutar tanto aplicaciones basadas en SWF como las basadas en HTML durante la fase de desarrollo. Con ADL se puede ejecutar una aplicación sin primero tener que empaquetarla e instalarla. De forma predeterminada, ADL utiliza un motor de ejecución incluido con el SDK, con lo cual no se necesita instalar el motor de ejecución por separado para utilizar ADL.

ADL imprime sentencias `trace` y errores en tiempo de ejecución a la salida estándar, pero no admite puntos de corte u otras funciones de depuración. Flash Debugger (o un entorno de desarrollo integrado como Flash Builder) se puede emplear para problemas de depuración complejos.

Nota: Si las sentencias `trace()` no se visualizan en la consola, asegúrese de que no ha especificado `ErrorReportingEnable` ni `TraceOutputFileEnable` en el archivo `mm.cfg`. Para obtener más información sobre la ubicación de este archivo en cada plataforma, consulte [Edición del archivo mm.cfg](#) (en inglés).

AIR admite la depuración directamente, por lo que no es necesario depurar una versión del motor de ejecución (tal y como se haría con Adobe® Flash® Player). Para dirigir la depuración de la línea de comandos, se utiliza Flash Debugger y AIR Debug Launcher (ADL).

Flash Debugger se distribuye en el directorio del SDK de Flex. Las versiones nativas, como `fdb.exe` en Windows, se encuentran en el subdirectorio `bin`. La versión de Java está en el subdirectorio `lib`. AIR Debug Launcher, `adl.exe`, está en el directorio `bin` de la instalación del SDK de Flex. (No existen ninguna versión de Java independiente).

Nota: no es posible iniciar una aplicación de AIR directamente con `fdb`, ya que `fdb` intenta iniciarla con Flash Player. Deje que la aplicación de AIR se conecte a una sesión de `fdb` en ejecución.

Uso de ADL

Para ejecutar una aplicación con ADL, utilice el siguiente patrón:

```
adl application.xml
```

`application.xml` es el archivo descriptor de la aplicación para esta.

La sintaxis completa para ADL es la siguiente:

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screenize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(Los elementos entre paréntesis, [], son opcionales.)

-runtime runtime-directory Especifica el directorio que contiene el motor de ejecución a utilizar. Si no se especifica, se utiliza el directorio del motor de ejecución del mismo SDK que el programa ADL. Si ADL se mueve fuera su carpeta en SDK, especifique el directorio del motor de ejecución. En Windows y Linux, especifique el directorio que contiene el directorio `Adobe AIR`. En Mac OS X, especifique el directorio que contiene `Adobe AIR.framework`.

-pubid *publisher-id* Asigna el valor especificado como ID del editor de la aplicación de AIR para esta ejecución. La especificación de un ID de editor temporal permite ensayar las funciones de una aplicación de AIR, como la comunicación a través de una conexión local, que utilizan el ID del editor para ayudar a identificar una aplicación con exclusividad. A partir de AIR 1.5.3, también se puede especificar el ID de editor en el archivo descriptor de la aplicación (y no se debe utilizar este parámetro).

Nota: a partir de AIR 1.5.3, un ID de editor no se vuelve a calcular ni a asignar automáticamente a una aplicación de AIR. Se puede especificar un ID de editor al crear una actualización en una aplicación de AIR existente, pero las nuevas aplicaciones no necesitan ni deben especificar un ID de editor.

-nodebug Desactiva la compatibilidad con la depuración. Si se utiliza, el proceso de la aplicación no podrá conectar con el depurador de Flash y se suprimen los cuadros de diálogo para excepciones no controladas. (Sin embargo, las sentencias trace continúan imprimiéndose en la ventana de la consola.) Si desactivamos la depuración, el funcionamiento de la aplicación se agilizará y el modo de ejecución será más similar al de una aplicación instalada.

-atlogin Simula el inicio de la aplicación al iniciar la sesión. Este indicador permite probar la lógica de la aplicación que solo se ejecuta cuando una aplicación se establece para iniciarse cuando el usuario inicia la sesión. Cuando se utiliza `-atlogin`, la propiedad `reason` del objeto `InvokeEvent` distribuido en la aplicación será `login` en lugar de `standard` (a menos que la aplicación ya se esté ejecutando).

-profile *profileName* ADL depura la aplicación utilizando el perfil especificado. *profileName* puede ser cualquiera de los valores siguientes:

- desktop
- extendedDesktop
- mobileDevice

Si el descriptor de la aplicación incluye un elemento `supportedProfiles`, el perfil especificado con `-profile` debe ser un miembro de la lista admitida. Si no se utiliza el indicador `-profile`, el primer perfil del descriptor de la aplicación se utiliza como el perfil activo. Si el descriptor de la aplicación no incluye el elemento `supportedProfiles` y no se utiliza el indicador `-profile`, se utiliza el perfil `desktop`.

Para obtener más información, consulte [“supportedProfiles”](#) en la página 250 y [“Perfiles de dispositivo”](#) en la página 256.

Valor **-screensize** Tamaño de la pantalla simulado para utilizar cuando se ejecutan aplicaciones en el perfil `mobileDevice` en el escritorio. Especifique el tamaño de la pantalla como tipo predefinido de pantalla o como las dimensiones de píxel de la anchura y altura normal de la visualización horizontal, más la anchura y altura de la pantalla completa. Para especificar el valor por tipo, utilice uno de los siguientes tipos de pantalla predefinidos:

Tipo de pantalla	Normal (anchura x altura)	Pantalla completa (anchura x altura)
480	720 x 480	720 x 480
720	1280 x 720	1280 x 720
1080	1920 x 1080	1920 x 1080
Droid	480 x 816	480 x 854
FWQVGA	240 x 432	240 x 432
FWVGA	480 x 854	480 x 854
HVGA	320 x 480	320 x 480
iPad	768 x 1004	768 x 1024

Tipo de pantalla	Normal (anchura x altura)	Pantalla completa (anchura x altura)
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

Para especificar las dimensiones de píxel de la pantalla directamente, utilice el siguiente formato:

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Especifique siempre las dimensiones de píxel de la visualización horizontal, esto es, especifique la anchura como un valor más pequeño que el valor de la altura. Por ejemplo, la pantalla NexusOne se puede especificar con:

```
-screensize 480x762:480x800
```

-extdir extension-directory Directorio en el que el motor de ejecución debe buscar extensiones nativas. El directorio contiene un subdirectorio para cada extensión nativa que utiliza la aplicación. Cada uno de estos subdirectorios contienen el archivo ANE *unpacked* de una extensión. Por ejemplo:

```
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
```

Cuando utilice el parámetro `-extdir`, tenga en cuenta lo siguiente:

- El comando ADL requiere que cada uno de los directorios especificados tenga la extensión de nombre de archivo `.ane`. No obstante, la parte del nombre de archivo previa al sufijo `.ane` puede ser cualquier nombre de archivo válido. No tiene que coincidir con el valor del elemento `extensionID` del archivo descriptor de la aplicación.
- Puede especificar el parámetro `-extdir` más de una vez.
- El uso del parámetro `-extdir` es diferente para la herramienta ADT y la herramienta ADL. En ADT, el parámetro especifica un directorio que contiene archivos ANE.
- También se puede utilizar la variable de entorno `AIR_EXTENSION_PATH` para especificar los directorios de extensiones. Consulte [“Variables del entorno de ADT”](#) en la página 198.

application.xml Archivo descriptor de la aplicación. Consulte [“Archivos descriptores de las aplicaciones de AIR”](#) en la página 214. El descriptor de la aplicación es el único parámetro que requiere ADL y, en la mayoría de los casos, el único parámetro necesario.

root-directory Especifica el directorio raíz de la aplicación a ejecutar. Si no se especifica, se utilizará el directorio que contiene el archivo descriptor de la aplicación.

--arguments Las cadenas de caracteres que aparezcan después de `--` se pasan a la aplicación como argumentos de la línea de comandos.

Nota: cuando se intenta iniciar una aplicación de AIR que ya está ejecutándose, no se inicia una nueva instancia de la aplicación, sino que se distribuye un evento `invoke` a la instancia que está en ejecución.

Ejemplos con ADL

Ejecute una aplicación del directorio actual:

```
adl myApp-app.xml
```


Ejecute una aplicación de un subdirectorio del directorio actual:

```
adl source/myApp-app.xml release
```

Ejecute una aplicación y pase dos argumentos de la línea de comandos, "tick" y "tock":

```
adl myApp-app.xml -- tick tock
```

Ejecute una aplicación con un motor de ejecución específico:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Ejecute una aplicación sin compatibilidad de depuración:

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

Ejecute una aplicación en el perfil de dispositivo móvil y simule el tamaño de pantalla Nexus One:

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Ejecute una aplicación utilizando Apache Ant para ejecutar la aplicación (las rutas mostradas en el ejemplo son para Windows):

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADL" value="${SDK_HOME}/bin/adl.exe"/>
<property name="APP_ROOT" value="c:/dev/MyApp/bin-debug"/>
<property name="APP_DESCRIPTOR" value="${APP_ROOT}/myApp-app.xml"/>

<target name="test">
  <exec executable="${ADL}">
    <arg value="${APP_DESCRIPTOR}"/>
    <arg value="${APP_ROOT}"/>
  </exec>
</target>
```

Códigos de error y de salida de ADL

En el cuadro siguiente se describen los códigos de error o de salida que imprime ADL:

Código de salida	Descripción
0	Inicio satisfactorio. ADL se cierra después de cerrarse la aplicación de AIR.
1	Invocación satisfactoria de una aplicación de AIR que ya está en ejecución. ADL se cierra inmediatamente.
2	Error de uso. Los argumentos proporcionados a ADL son incorrectos.
3	No se puede encontrar el motor de ejecución.
4	No se puede iniciar el motor de ejecución. Esto se debe con frecuencia a que la versión que se especifica en la aplicación no coincide con la versión del motor de ejecución.
5	Se ha producido un error de causa desconocida.
6	No se puede encontrar el archivo descriptor de la aplicación.
7	El contenido del descriptor de la aplicación no es válido. Este error suele indicar que el XML no está bien conformado.

Código de salida	Descripción
8	No se puede encontrar el archivo de contenido principal de la aplicación (especificado en el elemento <content> del archivo descriptor de la aplicación).
9	El archivo de contenido principal de la aplicación no es un archivo SWF o HTML válido.
10	La aplicación no admite el perfil especificado con la opción -profile.
11	El argumento -screensize no se admite en el perfil actual.

Capítulo 12: AIR Developer Tool (ADT)

AIR Developer Tool (ADT) es una herramienta de línea de comandos multiusos para el desarrollo de aplicaciones de AIR. ADT se puede utilizar para realizar las siguientes tareas:

- Empaquetar una aplicación de AIR como archivo de instalación .air
- Empaquetar una aplicación de AIR como instalador nativo: por ejemplo, como archivo instalador .exe en Windows, .ipa en iOS o .apk en Android.
- Empaquetar una extensión nativa como archivo de extensión nativa (ANE) de AIR.
- Firmar una aplicación de AIR con un certificado digital
- Cambiar (migrar) la firma digital utilizada para las actualizaciones de la aplicación
- Determinar los dispositivos conectados a un equipo
- Crear un certificado de firma de código digital con firma automática
- Desinstalar, iniciar e instalar de forma remota una aplicación en un dispositivo móvil
- Instalar y desinstalar de forma remota el motor de ejecución de AIR en un dispositivo móvil

ADT es un programa de Java incluido en el [SDK de AIR](#). Se debe disponer de Java 1.5 o superior para utilizarlo. El SDK incluye un archivo de script para invocar ADT. Para utilizar este script, la ubicación del programa de Java se debe incluir en la variable del entorno de ruta. Si el directorio `bin` del SDK de AIR también se detecta en la variable del entorno de ruta, puede escribir `adt` en la línea de comandos, con los argumentos adecuados, para invocar ADT. (Si no sabe cómo establecer la variable del entorno de ruta, consulte la documentación del sistema operativo. Como ayuda adicional, los procedimientos para establecer la ruta en la mayoría de los sistemas informáticos se describen en [“Variables del entorno de ruta”](#) en la página 319.)

Para el uso de ADT se requieren 2 GB de memoria en el equipo como mínimo. Si se dispone de menos memoria, ADT se puede ejecutar sin ella, especialmente al empaquetar aplicaciones para iOS.

Suponiendo que tanto el directorio `bin` del SDK de AIR Y Java se incluyan en la variable de ruta, ADT se puede ejecutar utilizando la siguiente sintaxis básica:

```
adt -command options
```

Nota: *la mayor parte de los entornos de desarrollo integrados, incluidos Adobe Flash Builder y Adobe Flash Professional, pueden empaquetar y firmar aplicaciones de AIR. El uso de ADT para estas tareas comunes no suele ser necesario cuando ya se utiliza este entorno de desarrollo. No obstante, puede que aún sea necesario emplear ADT como herramienta de la línea de comandos para las funciones que no se admitan en el entorno de desarrollo integrado. Asimismo, ADT se puede utilizar como herramienta de la línea de comandos como parte de un proceso de creación automatizado.*

Comandos de ADT

El primer argumento transmitido a ADT especifica uno de los siguientes comandos.

- `-package`: empaqueta una aplicación de AIR o extensión nativa de AIR (ANE).
- `-prepare`: empaqueta una aplicación de AIR como archivo intermedio (AIRI), pero no lo firma. Los archivos de AIRI no se pueden instalar.

- `-sign`: firma un paquete de AIRI producido con el comando `-prepare`. Los comandos `-prepare` y `-sign` permiten que el empaquetado y la firma se realicen en diferentes momentos. También se puede utilizar el comando `-sign` para firmar o volver a firmar un paquete ANE.
- `-migrate`: aplica una firma de migración a un paquete firmado de AIR, que permite utilizar un certificado de firma de código nuevo o renovado.
- `-certificate`: crea un certificado de firma de código digital con firma automática.
- `-checkstore`: verifica que se pueda acceder a un certificado digital en un almacén de claves.
- `-installApp`: instala una aplicación de AIR en un dispositivo o emulador de dispositivo.
- `-launchApp`: inicia una aplicación de AIR en un dispositivo o emulador de dispositivo.
- `-appVersion`: indica la versión de una aplicación de AIR instalada actualmente en un dispositivo o emulador de dispositivo.
- `-uninstallApp`: desinstala una aplicación de AIR de un dispositivo o emulador de dispositivo.
- `-installRuntime`: instala el motor de ejecución de AIR en un dispositivo o emulador de dispositivo.
- `-runtimeVersion`: indica la versión del motor de ejecución de AIR instalado actualmente en un dispositivo o emulador de dispositivo.
- `-uninstallRuntime`: desinstala el motor de ejecución de AIR instalado actualmente de un dispositivo o emulador de dispositivo.
- `-version`: indica el número de versión de ADT.
- `-devices`: proporciona información de dispositivo para los dispositivos móviles o emuladores conectados.
- `-help`: muestra la lista de comandos y opciones.

Diversos comandos de ADT comparten conjuntos relacionados de parámetros e indicadores de opción. Estos conjuntos de opciones se describen detalladamente por separado:

- [“Opciones de firma de código de ADT”](#) en la página 189
- [“Opciones de ruta y archivo”](#) en la página 191
- [“Opciones de conexión del depurador”](#) en la página 192
- [“Opciones de extensiones nativas”](#) en la página 193

Comando `package` de ADT

El comando `-package` debería ejecutarse desde el directorio de aplicación principal. El comando utiliza la siguiente sintaxis:

Cree un paquete de AIR desde los archivos de la aplicación del componente:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Cree un paquete nativo desde los archivos de la aplicación del componente:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Cree un paquete nativo que incluya una extensión nativa de los archivos de aplicación del componente:

```
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
```

Cree un paquete nativo a partir de un archivo de AIRI o AIR:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Cree un paquete de extensiones nativas a partir de archivos de extensión nativa de componentes:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Nota: no es necesario firmar ningún archivo ANE, por lo que los parámetros `AIR_SIGNING_OPTIONS` son opcionales en este ejemplo.

AIR_SIGNING_OPTIONS Las opciones de firma de AIR identifican el certificado utilizado para firmar un archivo de instalación de AIR. Las opciones de firma se describen con detenimiento en [“Opciones de firma de código de ADT”](#) en la página 189.

-migrate Este indicador especifica que la aplicación se firma con un certificado de migración además del certificado especificado en los parámetros `AIR_SIGNING_OPTIONS`. Este indicador solo es válido si se empaqueta una aplicación de escritorio como instalador nativo y la aplicación utiliza una extensión nativa. En otros casos se produce un error. Las opciones de firma para el certificado de migración se especifican como los parámetros **MIGRATION_SIGNING_OPTIONS**. Estas opciones de firma se detallan en [“Opciones de firma de código de ADT”](#) en la página 189. El uso del indicador `-migrate` permite crear una actualización para una aplicación de instalador nativo de escritorio que use una extensión nativa y cambiar el certificado de firma de código para la aplicación, como cuando caduca el certificado original. Para obtener más información, consulte [“Firma de una versión actualizada de una aplicación de AIR”](#) en la página 209.

El indicador `-migrate` del comando `-package` está disponible a partir de AIR 3.6.

-target Tipo de paquete para crear. Los tipos de paquete admitidos son:

- **air**: paquete de AIR. “air” es el valor predeterminado y no es necesario especificar el indicador **-target** al crear archivos de AIR o AIRI.
- **airn**: paquete de la aplicación nativa para dispositivos en el perfil de televisión ampliado.
- **ane**: paquete de extensión nativa de AIR
- Destinos del paquete Android:
 - **apk**: un paquete de Android. Un paquete producido con este destino solo se puede instalar en un dispositivo Android, no en un emulador.
 - **apk-captive-runtime**: un paquete de Android que incluye tanto la aplicación como una versión captadora del motor de ejecución de AIR. Un paquete producido con este destino solo se puede instalar en un dispositivo Android, no en un emulador.
 - **apk-debug**: paquete de Android con información de depuración adicional. (Los archivos SWF de la aplicación también se deben compilar con compatibilidad de depuración.)
 - **apk-emulator**: paquete de Android para su uso en un emulador sin compatibilidad de depuración. (Utilice el destino **apk-debug** para permitir la depuración en ambos emuladores y dispositivos.)
 - **apk-profile**: paquete de Android que admite la creación de perfiles de memoria y el rendimiento de la aplicación.
- Destinos del paquete iOS:
 - **ipa-ad-hoc**: paquete de iOS para la distribución ad hoc.
 - **ipa-app-store**: paquete de iOS para la distribución de App Store de Apple.
 - **ipa-debug**: paquete de iOS con información de depuración adicional. (Los archivos SWF de la aplicación también se deben compilar con compatibilidad de depuración.)
 - **ipa-test**: paquete de iOS compilado sin información de depuración u optimización.
 - **ipa-debug-interpret**: funcionalidad equivalente a un paquete debug, pero con compilación más rápida. No obstante, el código de bytes de ActionScript se interpreta y no se transforma en código de máquina. Como resultado, la ejecución del código es más lenta en un paquete interpret.
 - **ipa-debug-interpret-simulator**: funcionalidad equivalente a **ipa-debug-interpret**, pero empaquetado para simulador de iOS. Solo para Macintosh. Si utiliza esta opción, también debe incluir la opción **-platformsdk** y especificar la ruta de acceso al SDK del simulador de iOS.
 - **ipa-test-interpret**: funcionalidad equivalente a un paquete test, pero con compilación más rápida. No obstante, el código de bytes de ActionScript se interpreta y no se transforma en código de máquina. Como resultado, la ejecución del código es más lenta en un paquete interpret.
 - **ipa-test-interpret-simulator**: funcionalidad equivalente a **ipa-test-interpret**, pero empaquetado para simulador de iOS. Solo para Macintosh. Si utiliza esta opción, también debe incluir la opción **-platformsdk** y especificar la ruta de acceso al SDK del simulador de iOS.
- **native**: instalador de escritorio nativo. El tipo de archivo generado es el formato de instalación nativo del sistema operativo en el que se ejecuta el comando:
 - **EXE**: Windows
 - **DMG**: Mac
 - **DEB**: Ubuntu Linux (AIR 2.6 o anterior)
 - **RPM** — Fedora u OpenSuse Linux (AIR 2.6 o anterior)

Para obtener más información, consulte “[Empaquetado de un instalador nativo de escritorio](#)” en la página 60.

-sampler (solo iOS, AIR 3.4 y posterior) Habilita el muestreador de ActionScript basado en telemetría en aplicaciones de iOS. Con este indicador es posible crear un perfil de la aplicación con Adobe Scout. Aunque **Scout** puede crear perfiles de cualquier contenido de la plataforma de Flash, al habilitar la telemetría detallada se puede obtener más información sobre el tiempo de las funciones de ActionScript, DisplayList, el procesamiento de Stage3D y mucho más. Tenga en cuenta que utilizar este indicador afecta al rendimiento, por lo que no debe utilizarse en aplicaciones de producción.

-hideAneLibSymbols (solo iOS, AIR 3.4 y posterior) Los desarrolladores de aplicaciones pueden usar diversas extensiones nativas de múltiples orígenes y, si los archivos ANE comparten un nombre de símbolo común, ADT genera un error de “símbolo duplicado en archivo de objeto”. En algunos casos, este error puede desembocar incluso en un bloqueo del motor de ejecución. Puede utilizar la opción `hideAneLibSymbols` para especificar si quiere o no que los símbolos de la biblioteca de archivos ANE sea visible solo para los orígenes de biblioteca (yes) o visible globalmente (no):

- **yes**: oculta los símbolos ANE; resuelve cualquier error de conflicto de símbolos no intencionado.
- **no** (valor predeterminado): no oculta ningún símbolo ANE. Este comportamiento es previo a AIR 3.4.

-embedBitcode (solo para iOS, AIR 25 y versiones posteriores) Los desarrolladores de aplicaciones pueden utilizar la opción `embedBitcode` para especificar si desean incrustar código de bits en su aplicación de iOS indicando Sí o No. El valor predeterminado de esta opción es No.

DEBUGGER_CONNECTION_OPTIONS Las opciones de conexión del depurador especifican si un paquete de depuración debe intentar conectarse a un depurador remoto que se ejecuta en otro equipo, o bien, detectar una conexión desde un depurador remoto. Este conjunto de opciones solo se admite para paquetes de depuración móviles (se aplica a apk-debug y ipa-debug). Estas opciones se describen en “[Opciones de conexión del depurador](#)” en la página 192.

-airDownloadURL Especifica una URL alternativa para descargar e instalar el motor de ejecución de AIR en los dispositivos Android. Si no se especifica, una aplicación de AIR redirigirá al usuario al motor de ejecución de AIR en Android Market si el motor de ejecución no está instalado aún.

Si la aplicación se distribuye mediante un catálogo de soluciones alternativo (distinto de Android Market administrado por Google), puede que sea necesario especificar la URL para descargar el motor de ejecución de AIR desde ese catálogo. Algunos catálogos de aplicaciones alternativos no permiten que las aplicaciones requieran una descarga fuera del catálogo. Esta opción solo se admiten para paquetes de Android.

NATIVE_SIGNING_OPTIONS Las opciones de firma nativas identifican el certificado utilizado para firmar un archivo de paquete nativo. Estas opciones de firma se utilizan para aplicar una firma empleada por el sistema operativo nativo, no el motor de ejecución de AIR. Las opciones son idénticas en AIR_SIGNING_OPTIONS y se describen detalladamente en “[Opciones de firma de código de ADT](#)” en la página 189.

Las firmas nativas se admiten en Windows y Android. En Windows, tanto las opciones de firma de AIR y como las opciones de firma nativas se deben especificar. En Android, únicamente las opciones de firma nativas se pueden especificar.

En muchos casos, se puede utilizar el mismo certificado de firma de código para aplicar tanto una firma nativa como de AIR. Sin embargo, no es así en todos los casos. Por ejemplo, la directiva de Google para las aplicaciones enviadas a Android Market indica que todas las aplicaciones se deben firmar con un certificado que sea válido hasta el año 2033 como mínimo. Esto significa que los certificados emitidos por una entidad emisora de certificados conocida, que se recomiendan al aplicar una firma de AIR, no se deben utilizar para firmar una aplicación de Android. (Ninguna entidad emisora de certificados emite un certificado de firma de código con ese periodo de validez tan largo.)

output Nombre del archivo del paquete para crear. La especificación de la extensión del archivo es opcional. Si no se especifica, se añade una extensión adecuada al valor `-target` y al sistema operativo actual.

app_descriptor La ruta al archivo descriptor de la aplicación. La ruta se puede indicar en relación con el directorio actual o como ruta absoluta. (En el archivo de AIR el archivo descriptor de la aplicación cambia de nombre a *application.xml*.)

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino:

- Android: el SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)
- iOS: el SDK de AIR se suministra con SDK de iOS captador. La opción **-platformsdk** permite empaquetar aplicaciones con un SDK externo para no sufrir las restricciones del SDK de SDK captador. Por ejemplo, si ha creado una extensión con el último SDK de iOS, puede especificar dicho SDK cuando empaquete la aplicación. Además, si utiliza ADT con el simulador de iOS, también debe incluir la opción **-platformsdk** y especificar la ruta de acceso al SDK del simulador de iOS.

-arch Los desarrolladores de aplicaciones pueden utilizar este argumento para crear APK para plataformas x86; toma los valores siguientes:

- armv7: ADT empaqueta APK para la plataforma Android armv7.
- x86: ADT empaqueta APK para la plataforma Android x86.

armv7 es el valor predeterminado si no se especifica ningún valor.

FILE_OPTIONS Identifica los archivos de la aplicación para incluir en el paquete. Las opciones del archivo se describen en su totalidad en “[Opciones de ruta y archivo](#)” en la página 191. No especifique opciones de archivo al crear un paquete nativo desde un archivo de AIR o AIRI.

input_airi Se especifica al crear un paquete nativo desde un archivo de AIRI. AIR_SIGNING_OPTIONS es necesario si el destino es *air* (o no se especifica ningún destino).

input_air Se especifica al crear un paquete nativo desde un archivo de AIR. No especifique AIR_SIGNING_OPTIONS.

ANE_OPTIONS Identifica las opciones y archivos para crear un paquete de extensión nativa. Las opciones del paquete de extensión se describen detalladamente en “[Opciones de extensiones nativas](#)” en la página 193.

Ejemplos del comando ADT -package

Empaquete los archivos de la aplicación específicos en el directorio actual para una aplicación de AIR basada en SWF:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Empaquete los archivos de la aplicación específicos en el directorio actual para una aplicación de AIR basada en HTML:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js image.gif
```

Empaquete todos los archivos y subdirectorios del directorio de trabajo actual:

```
adt -package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Nota: el archivo del almacén de claves contiene la clave privada que se utilizó para firmar la aplicación. No incluya nunca el certificado de firma en el paquete de AIR. Si utiliza comodines en el comando ADT, coloque el archivo del almacén de claves en otra carpeta para que no se incluya en el paquete. En este ejemplo el archivo del almacén de claves, cert.p12, reside en el directorio principal.

Empaquete solamente los principales archivos y un subdirectorio de imágenes:


```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Empaquete una aplicación basada en HTML y todos los archivos de los subdirectorios HTML, scripts e images:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRAliases.js  
html scripts images
```

Empaquete el archivo application.xml y el archivo SWF principal que se encuentran en un directorio de trabajo (release/bin):

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml -C  
release/bin myApp.swf
```

Empaquete componentes procedentes de más de un lugar en el sistema de archivos de construcción. En este ejemplo, los componentes de la aplicación se encuentran en las siguientes carpetas antes de que se empaquetan:

```
/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
```

La ejecución del siguiente comando ADT desde el directorio /devRoot/myApp:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp-app.xml  
-C release/bin myApp.swf (or myApp.html)  
-C ../artwork/myApp images  
-C ../libraries/release libs
```

produce un paquete con la siguiente estructura:

```
/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
```

Ejecute ADT como programa Java para una aplicación sencilla basada en SWF (sin establecer la ruta de clase):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Ejecute ADT como programa Java para una aplicación sencilla basada en HTML (sin establecer la ruta de clase):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Ejecute ADT como programa de Java (con la ruta de clase de Java definida para incluir el paquete ADT.jar):

```
java -com.adobe.air.ADT -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

Ejecute ADT como tarea de Java en Apache Ant (aunque suele ser mejor usar el comando de ADT directamente en scripts Ant). Las rutas mostradas en el ejemplo son para Windows:

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADT.JAR" value="{SDK_HOME}/lib/adt.jar"/>

target name="package">
  <java jar="{ADT.JAR}" fork="true" failonerror="true">
    <arg value="-package"/>
    <arg value="-storetype"/>
    <arg value="pkcs12"/>
    <arg value="-keystore"/>
    <arg value="../../ExampleCert.p12"/>
    <arg value="myApp.air"/>
    <arg value="myApp-app.xml"/>
    <arg value="myApp.swf"/>
    <arg value="icons/*.png"/>
  </java>
</target>
```

Nota: en algunos sistemas informáticos, los caracteres de doble de las rutas del sistema de archivos se pueden malinterpretar. Si esto sucede, intente establecer JRE utilizado para ejecutar ADT para que se utilice el conjunto de caracteres UTF-8. Esto se realiza de forma predeterminada en el script utilizado para iniciar ADT en Mac y Linux. En el archivo `adt.bat` de Windows, o cuando ADT se ejecuta directamente desde Java, especifique la opción `-Dfile.encoding=UTF-8` en la línea de comandos de Java.

Comando prepare de ADT

El comando `-prepare` crea un paquete de AIRI sin firmar. Los paquetes de AIRI no se pueden utilizar por sí mismos. Utilice el comando `-sign` para convertir un archivo de AIRI en un paquete de AIR firmado o el comando del paquete para convertir el archivo de AIRI en un paquete nativo.

El comando `-prepare` utiliza la siguiente sintaxis:

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output Nombre del archivo de AIRI creado.

app_descriptor La ruta al archivo descriptor de la aplicación. La ruta se puede indicar en relación con el directorio actual o como ruta absoluta. (En el archivo de AIR el archivo descriptor de la aplicación cambia de nombre a *application.xml*.)

FILE_OPTIONS Identifica los archivos de la aplicación para incluir en el paquete. Las opciones del archivo se describen en su totalidad en [“Opciones de ruta y archivo”](#) en la página 191.

Comando sign de ADT

El comando `-sign` firma archivos de AIRI y ANE.

El comando `-sign` emplea la siguiente sintaxis:

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Las opciones de firma de AIR identifican el certificado utilizado para firmar un archivo de paquete. Las opciones de firma se describen con detenimiento en [“Opciones de firma de código de ADT”](#) en la página 189.

input Nombre del archivo de AIRI o ANE para firmar.

output Nombre del paquete firmado para crear.

Si un archivo de ANE ya está firmado, la firma antigua se descartará. (Los archivos de AIR no se pueden volver a firmar. Para utilizar una nueva firma para una actualización de la aplicación, use el comando `-migrate`.)

Comando migrate de ADT

El comando `-migrate` aplica una firma de migración a un archivo de AIR. Se debe utilizar una firma de migración cuando se renueva o se cambia el certificado digital y es necesario actualizar las aplicaciones firmadas con el certificado antiguo.

Para obtener más información sobre el empaquetado de aplicaciones de AIR con una firma de migración, consulte [“Firma de una versión actualizada de una aplicación de AIR”](#) en la página 209.

Nota: *el certificado de migración debe aplicarse en 365 días a partir de la fecha de caducidad del certificado. Una vez transcurrido el periodo de gracia, las actualizaciones de la aplicación ya no se pueden firmar con una firma de migración. Los usuarios pueden actualizar en primer lugar a una versión de la aplicación que se haya firmado con una firma de migración y, posteriormente, instalar la actualización más reciente, o bien, pueden desinstalar la aplicación original e instalar el nuevo paquete de AIR.*

Para usar una firma de migración, en primer lugar firme la aplicación de AIR utilizando el certificado nuevo o renovado (con los comandos `-package` o `-sign`) y después aplique la firma de migración utilizando el antiguo certificado y el comando `-migrate`.

El comando `-migrate` utiliza esta sintaxis:

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Opciones de firma de AIR que identifican el certificado original que se utilizó para firmar las versiones existentes de la aplicación de AIR. Las opciones de firma se describen con detenimiento en [“Opciones de firma de código de ADT”](#) en la página 189.

input Archivo de AIR ya firmado con el certificado de la aplicación NUEVO.

output Nombre del paquete final que incluye las firmas de los certificados nuevo y antiguo.

los nombres de archivo empleados para los archivos de AIR de entrada y salida deben ser distintos.

***Nota:** El comando migrate de ADT no se puede usar con aplicaciones AIR de escritorio que incluyan extensiones nativas, ya que dichas aplicaciones están empaquetadas como instaladores nativos, no como archivos .air. Para cambiar certificados de una aplicación AIR de escritorio que incluya una extensión nativa, empaquete la aplicación utilizando el [“Comando package de ADT”](#) en la página 175 con el indicador -migrate.*

Comando checkstore de ADT

El comando -checkstore permite comprobar la validez de un almacén de claves. El comando utiliza la siguiente sintaxis:

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS Opciones de firma que identifican el almacén de claves para validar. Las opciones de firma se describen con detenimiento en [“Opciones de firma de código de ADT”](#) en la página 189.

Comando certificate de ADT

El comando -certificate permite crear un certificado de firma digital con firma automática. El comando utiliza la siguiente sintaxis:

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type  
output password
```

-cn Cadena asignada como nombre común del nuevo certificado.

-ou Cadena asignada como unidad organizativa emisora del certificado. (Opcional.)

-o Cadena asignada como organización emisora del certificado. (Opcional.)

-c Código de país de dos letras de la norma ISO-3166. Si el código suministrado no es válido, no se genera el certificado. (Opcional.)

-validityPeriod Número de años de validez del certificado. Si no se especifica ninguna validez, se asigna el valor de cinco años. (Opcional.)

key_type El tipo de clave que se va a utilizar para el certificado es 2048-RSA.

output Nombre de archivo y ruta para el archivo de certificado que se va a generar.

password La contraseña para acceder al nuevo certificado. Cuando se firmen archivos de AIR con este certificado, se necesitará la contraseña.

Comando installApp de ADT

El comando -installApp instala una aplicación en un dispositivo o emulador.

Se debe desinstalar una aplicación existente antes de volver a realizar la instalación con este comando.

El comando utiliza la siguiente sintaxis:

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package fileName
```

-platform Nombre de la plataforma del dispositivo. Especifique *ios* o *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino (opcional):

- **Android:** el SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)
- **iOS:** el SDK de AIR se suministra con SDK de iOS captador. La opción `-platformsdk` permite empaquetar aplicaciones con un SDK externo para no sufrir las restricciones del SDK de SDK captador. Por ejemplo, si ha creado una extensión con el último SDK de iOS, puede especificar dicho SDK cuando empaquete la aplicación. Además, si utiliza ADT con el simulador de iOS, también debe incluir la opción `-platformsdk` y especificar la ruta de acceso al SDK del simulador de iOS.

-device Especifica *ios_simulator*, el número de serie (Android) o el control (iOS) del dispositivo conectado. En iOS, este parámetro es obligatorio; en Android, este parámetro solo necesita especificarse cuando se asocia más de un dispositivo o emulador de Android al equipo durante la ejecución. Si el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida (Android) o dispositivo no válido especificado (iOS). Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

***Nota:** la opción de instalar un archivo IPA directamente en un dispositivo iOS está disponible en AIR 3.4 y versión posterior, y requiere tener instalado iTunes 10.5.0 o posterior.*

Utilice el comando `adt -devices` (disponible en AIR 3.4 y versiones posteriores) para determinar el control o número de serie de los dispositivos conectados. Tenga en cuenta que en iOS se utiliza el control, no el UUID del dispositivo. Para obtener más información, consulte “[Comando de dispositivos ADT](#)” en la página 188.

Además, en Android puede utilizar la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

-package Nombre del archivo del paquete que se va a instalar. En iOS, debe ser un archivo IPA. En Android, debe ser un paquete APK. Si el paquete especificado ya se ha instalado, ADT devuelve el error 14:Device de código.

Comando appVersion de ADT

El comando `-appVersion` indica la versión instalada de una aplicación en un dispositivo o emulador. El comando utiliza la siguiente sintaxis:

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Nombre de la plataforma del dispositivo. Especifique *ios* o *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino:

- Android: el SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)
- iOS: el SDK de AIR se suministra con SDK de iOS captador. La opción **-platformsdk** permite empaquetar aplicaciones con un SDK externo para no sufrir las restricciones del SDK de SDK captador. Por ejemplo, si ha creado una extensión con el último SDK de iOS, puede especificar dicho SDK cuando empaquete la aplicación. Además, si utiliza ADT con el simulador de iOS, también debe incluir la opción **-platformsdk** y especificar la ruta de acceso al SDK del simulador de iOS.

-device: especifique *ios_simulator* o el número de serie del dispositivo. El dispositivo solo se debe especificar cuando existen varios dispositivos o emuladores de Android incorporados al equipo y en ejecución. Si el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida. Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

En Android, utilice la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

-appid ID de la aplicación de AIR de la aplicación instalada. Si no hay ninguna aplicación instalada con el ID especificado, ADT devuelve el error 14: Device del código de salida.

Comando launchApp de ADT

El comando **-launchApp** ejecuta una aplicación instalada en un dispositivo o emulador. El comando utiliza la siguiente sintaxis:

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Nombre de la plataforma del dispositivo. Especifique *ios* o *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino:

- Android: el SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)
- iOS: el SDK de AIR se suministra con SDK de iOS captador. La opción **-platformsdk** permite empaquetar aplicaciones con un SDK externo para no sufrir las restricciones del SDK de SDK captador. Por ejemplo, si ha creado una extensión con el último SDK de iOS, puede especificar dicho SDK cuando empaquete la aplicación. Además, si utiliza ADT con el simulador de iOS, también debe incluir la opción **-platformsdk** y especificar la ruta de acceso al SDK del simulador de iOS.

-device: especifique *ios_simulator* o el número de serie del dispositivo. El dispositivo solo se debe especificar cuando existen varios dispositivos o emuladores de Android incorporados al equipo y en ejecución. Si el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida. Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

En Android, utilice la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

-appid ID de la aplicación de AIR de la aplicación instalada. Si no hay ninguna aplicación instalada con el ID especificado, ADT devuelve el error 14: Device del código de salida.

Comando `uninstallApp` de ADT

El comando `-uninstallApp` elimina por completo una aplicación instalada en un emulador o dispositivo remoto. El comando utiliza la siguiente sintaxis:

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Nombre de la plataforma del dispositivo. Especifique *ios* o *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino:

- Android: el SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno `AIR_ANDROID_SDK_HOME` ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)
- iOS: el SDK de AIR se suministra con SDK de iOS captador. La opción `-platformsdk` permite empaquetar aplicaciones con un SDK externo para no sufrir las restricciones del SDK de SDK captador. Por ejemplo, si ha creado una extensión con el último SDK de iOS, puede especificar dicho SDK cuando empaquete la aplicación. Además, si utiliza ADT con el simulador de iOS, también debe incluir la opción `-platformsdk` y especificar la ruta de acceso al SDK del simulador de iOS.

-device: especifique *ios_simulator* o el número de serie del dispositivo. El dispositivo solo se debe especificar cuando existen varios dispositivos o emuladores de Android incorporados al equipo y en ejecución. Si el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida. Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

En Android, utilice la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

-appid ID de la aplicación de AIR de la aplicación instalada. Si no hay ninguna aplicación instalada con el ID especificado, ADT devuelve el error 14: Device del código de salida.

Comando `installRuntime` de ADT

El comando `-installRuntime` instala el motor de ejecución de AIR en un dispositivo.

Se debe desinstalar una versión existente del motor de ejecución de AIR antes de realizar la instalación con este comando

El comando utiliza la siguiente sintaxis:

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package fileName
```

-platform Nombre de la plataforma del dispositivo. Actualmente este comando solo se admite en la plataforma Android. Utilice el nombre, *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino. Actualmente, el único SDK de la plataforma admitido es Android. El SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)

-device Número de serie del dispositivo. El dispositivo solo se debe especificar cuando existen varios dispositivos o emuladores incorporados al equipo y en ejecución. Si el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida. Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

En Android, utilice la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

-package Nombre del archivo del motor de ejecución para instalar. En Android, debe ser un paquete APK. Si no se especifica ningún paquete, el motor de ejecución adecuado para el dispositivo o emulador se selecciona de los que se encuentran disponibles en el SDK de AIR. Si el motor de ejecución ya está instalado, ADT devuelve el error 14:Device del código.

Comando runtimeVersion de ADT

El comando `-runtimeVersion` indica la versión instalada del motor de ejecución de AIR en un dispositivo o emulador. El comando utiliza la siguiente sintaxis:

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Nombre de la plataforma del dispositivo. Actualmente este comando solo se admite en la plataforma Android. Utilice el nombre, *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino. Actualmente, el único SDK de la plataforma admitido es Android. El SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)

-device Número de serie del dispositivo. El dispositivo solo se debe especificar cuando existen varios dispositivos o emuladores incorporados al equipo y en ejecución. Si el motor de ejecución no está instalado o el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida. Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

En Android, utilice la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

Comando uninstallRuntime de ADT

El comando `-uninstallRuntime` elimina por completo el motor de ejecución de AIR desde un dispositivo o emulador. El comando utiliza la siguiente sintaxis:

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```


-platform Nombre de la plataforma del dispositivo. Actualmente este comando solo se admite en la plataforma Android. Utilice el nombre, *android*.

-platformsdk Ruta al SDK de la plataforma para el dispositivo de destino. Actualmente, el único SDK de la plataforma admitido es Android. El SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Asimismo, no es necesario proporcionar la ruta del SDK de la plataforma en la línea de comandos si la variable del entorno AIR_ANDROID_SDK_HOME ya está establecida. (Si ambos están establecidos, se utiliza la ruta proporcionada en la línea de comandos.)

-device Número de serie del dispositivo. El dispositivo solo se debe especificar cuando existen varios dispositivos o emuladores incorporados al equipo y en ejecución. Si el dispositivo especificado no está conectado, ADT devuelve el error 14: Device del código de salida. Si hay varios dispositivos o emuladores conectados y no se especifica ningún dispositivo, ADT devuelve el error 2: Usage del código de salida.

En Android, utilice la herramienta Android ADB para enumerar los números de serie de los emuladores en ejecución y los dispositivos incorporados:

```
adb devices
```

Comando de dispositivos ADT

El comando `-help` de ADT muestra los identificadores de dispositivo de los dispositivos móviles y emuladores conectados actualmente:

```
adt -devices -platform ios|android
```

-platform Nombre de la plataforma que se va a verificar. Especifique `android` o `ios`.

Nota: en iOS, este comando requiere tener instalado iTunes 10.5.0 o versión posterior.

Comando help de ADT

El comando `-help` de ADT muestra un recordatorio escueto de las opciones de la línea de comandos:

```
adt -help
```

La salida de `help` utiliza las siguientes convenciones simbólicas:

- `<>`: los elementos situados entre los corchetes angulares indican información que se debe proporcionar.
- `()`: los elementos entre paréntesis indican opciones que se tratan como grupo en la salida del comando `help`.
- `ALL_CAPS`: los elementos expresados en mayúsculas indican un conjunto de opciones que se describen por separado.
- `|`: OR. Por ejemplo, `(A | B)`, significa elemento A o elemento B.
- `?`: 0 o 1. Un signo de interrogación que sigue a un elemento indica que es opcional y que solo se puede producir una instancia, si se utiliza.
- `*`: 0 o más. Un asterisco que sigue a un elemento indica que dicho elemento es opcional y que se puede producir cualquier número de instancias.
- `+`: 1 o más. Un signo más que sigue a un elemento indica que es necesario un elemento y que se pueden producir varias instancias.
- Sin símbolo: si un elemento no tiene símbolo de sufijo, se indica que dicho elemento es necesario y que solo se puede producir una instancia.

Conjuntos de opciones de ADT

Diversos comandos de ADT comparten conjuntos de opciones comunes.

Opciones de firma de código de ADT

ADT utiliza Java Cryptography Architecture (JCA) para acceder a las claves privadas y los certificados para firmar aplicaciones de AIR. Las opciones de firma identifican el almacén de claves, así como la clave privada y el certificado que en él se encuentran.

El almacén de claves debe incluir tanto la clave privada como la cadena de certificación asociada. Si el certificado de firma está encadenado con un certificado de confianza en un ordenador, el contenido del nombre común del certificado se presenta como el nombre del editor en el cuadro de diálogo de la instalación de AIR.

ADT exige que el certificado cumpla con la norma x509v3, [RFC3280](#) (en inglés), e incluya la extensión de uso mejorado de claves con los valores correspondientes para la firma de códigos. Se respetan las limitaciones definidas propias del certificado y estas podrían descartar el uso de algunos certificados para firmar aplicaciones de AIR.

***Nota:** ADT utiliza la configuración de proxy para el entorno de ejecución de Java (JRE), si procede, para conectarse a recursos de Internet con la finalidad de comprobar listas de revocación de certificados y obtener marcas de hora. Si tiene algún problema con la conexión a recursos de Internet cuando utiliza ADT y la red requiere una configuración de proxy concreta, es posible que necesite ajustar la configuración de proxy para JRE.*

Sintaxis de las opciones de firma de AIR

Las opciones de firma utilizan la siguiente sintaxis (en una sola línea de comandos):

```
-alias aliasName  
-storetype type  
-keystore path  
-storepass password1  
-keypass password2  
-providerName className  
-tsa url
```

-alias Alias de una clave en el almacén de claves. No es necesario especificar un alias si el almacén de claves contiene un solo certificado. Si no se especifica ningún alias, ADT utiliza la primera clave del almacén de claves.

No todas las aplicaciones con administración del almacén de claves permiten que se asigne un alias a los certificados. Si hace uso del almacén de claves del sistema en Windows, por ejemplo, utilice como alias el nombre distinguido del certificado. Se puede utilizar la utilidad Java Keytool para enumerar los certificados disponibles para poder determinar el alias. Por ejemplo, si se ejecuta el comando:

```
keytool -list -storetype Windows-MY
```

se produce una salida para un certificado como la siguiente:

```
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
```

Para hacer referencia a este certificado en la línea de comandos de ADT, defina el alias en:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

En Mac OS X, el alias de un certificado en la Llave es el nombre que se muestra en la aplicación Acceso a Llaves.

-storetype: tipo de almacén de claves, determinado por la implementación del almacén de claves. La implementación del almacén de claves predeterminada incluida con la mayoría de las instalaciones de Java es compatible con los tipos JKS y PKCS12. Java 5.0 ofrece compatibilidad con el tipo PKCS11, para acceder a almacenes de claves en tokens de hardware, y al tipo Keychain, para acceder a la Llave de Mac OS X. Java 6.0 ofrece compatibilidad con el tipo MSCAPI (en Windows). Si se han instalado y configurado otros proveedores de JCA, puede que se disponga además de otros tipos de almacenes de claves. Si no se especifica ningún tipo de almacén de claves, se utiliza el tipo predeterminado para el proveedor de JCA predeterminado.

Tipo de almacén	Formato del almacén de claves	Versión mínima de Java
JKS	Archivo de almacén de claves de Java (.keystore)	1.2
PKCS12	Archivo PKCS12 (.p12 o .pfx)	1.4
PKCS11	Token de hardware	1.5
KeychainStore	Llave de Mac OS X	1.5
Windows-MY o Windows-ROOT	MSCAPI	1.6

-keystore : ruta al archivo del almacén de claves para tipos de almacén basados en archivo.

-storepass : contraseña para acceder al almacén de claves. Si no se especifica, ADT la solicita.

-keypass : contraseña para acceder a la clave privada que se utiliza para firmar la aplicación de AIR. Si no se especifica, ADT la solicita.

***Nota:** si se indica una contraseña como parte del comando de ADT, los caracteres de la contraseña se guardan en el historial de la línea de comandos. Por lo tanto, el uso de las opciones -keypass o -storepass no se recomienda cuando la seguridad del certificado es importante. También se debe tener en cuenta que cuando se omiten las opciones de contraseña, los caracteres escritos en la solicitud de contraseña no se muestran (por los mismos motivos de seguridad). Simplemente escriba la contraseña y presione la tecla Intro.*

-providerName: proveedor de JCA para el tipo de almacén de claves especificado. Si no se especifica, ADT utiliza el proveedor predeterminado para ese tipo de almacén de claves.

-tsa : especifica la URL de un RFC3161 (en inglés) para marcar la hora en la firma digital. Si no se especifica una URL, se utiliza un servidor de marcas de hora predeterminado suministrado por Geotrust. Cuando la firma de una aplicación de AIR lleva una marca de hora, la aplicación puede instalarse después de caducado el certificado de firma porque la marca de hora verifica que el certificado era válido al momento de firmar.

Si ADT no puede conectarse al servidor de marcas de hora, se cancela la firma y no se produce ningún paquete. La marca de hora se puede desactivar especificando `-tsa none`. Sin embargo, una aplicación de AIR empaquetada sin marca de hora no se puede instalar una vez caducado el certificado de firma.

***Nota:** muchas de las opciones de firma equivalen a la misma opción de la utilidad Java Keytool. La utilidad Keytool sirve para examinar y administrar almacenes de claves en Windows. La utilidad de seguridad de Apple® sirve para lo mismo en Mac OS X.*

-provisioning-profile Archivo de suministro de Apple iOS. (Obligatorio únicamente para empaquetar aplicaciones de iOS.)

Ejemplos de opciones de firma

Firma con un archivo .p12:

```
-storetype pkcs12 -keystore cert.p12
```

Firma con el almacén de claves de Java predeterminado:

```
-alias AIRcert -storetype jks
```

Firma con un almacén de claves de Java específico:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Firma con la Llave de Mac OS X:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Firma con el almacén de claves del sistema de Windows:

```
-alias cn=AIRCert -storetype Windows-MY
```

Firma con un token de hardware (consulte las instrucciones del fabricante del token para ver cómo se configura Java para utilizar el token y para obtener el valor correcto de `providerName`):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Firma sin incorporar una marca de hora:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Opciones de ruta y archivo

Las opciones de ruta y archivo especifican todos los archivos que se incluyen en el paquete. Las opciones de ruta y archivo utilizan la siguiente sintaxis:

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs: archivos y directorios a empaquetar en el archivo de AIR. Se puede especificar la cantidad de archivos y directorios que se desee, delimitados por un espacio en blanco. Si se incluye un directorio en la lista, se añaden al paquete todos los archivos y subdirectorios que contenga, excepto los archivos ocultos. (Si se especifica el archivo descriptor de la aplicación, sea directamente o mediante el uso de un comodín o por expansión de un directorio, este se pasa por alto y no se añade al paquete por segunda vez). Los archivos y directorios especificados deben estar incluidos en el directorio actual o en uno de sus subdirectorios. Para cambiar el directorio actual, utilice la opción `-C`.

Importante: no se pueden utilizar comodines en los argumentos `file_or_dir` después de la opción `-C`. (Los shells de comandos expanden los comodines antes de pasar los argumentos a ADT, por lo que ADT buscaría los archivos en el lugar equivocado). Sí se puede utilizar el punto (".") para representar el directorio actual. Por ejemplo: `-C assets .` copia todo el contenido del directorio `assets`, incluidos los subdirectorios, hasta el nivel raíz del paquete de la aplicación.

`-C dir files_and_dirs` Cambia el directorio de trabajo por el valor de `dir` antes de procesar más archivos y directorios añadidos al paquete de la aplicación (especificados en `files_and_dirs`). Los archivos o directorios se añaden a la raíz del paquete de la aplicación. La opción `-C` se puede utilizar todas las veces que se desee para incluir archivos desde varios lugares del sistema de archivos. Si se especifica una ruta relativa para `dir`, la ruta siempre se resuelve desde el directorio de trabajo original.

A medida que ADT procesa los archivos y directorios incluidos en el paquete, se van guardando las rutas relativas entre el directorio actual y los archivos de destino. Estas rutas se expanden en la estructura de directorios de la aplicación cuando se instala el paquete. Por lo tanto, si se especifica `-C release/bin lib/feature.swf`, el archivo `release/bin/lib/feature.swf` se coloca en el subdirectorio `lib` de la carpeta raíz de la aplicación.

`-e file_or_dir dir` Sitúa el directorio o archivo en el directorio del paquete especificado. Esta opción no se puede usar al empaquetar un archivo ANE.

***Nota:** el elemento `<content>` del archivo descriptor de la aplicación debe especificar la ubicación final del archivo principal de la aplicación dentro del árbol de directorios del paquete de la aplicación.*

-extdir dir El valor de `dir` es el nombre de un directorio para buscar extensiones nativas (archivos ANE). Especifique una ruta absoluta o una ruta relativa al directorio actual. Puede especificar la opción `-extdir` varias veces.

El directorio especificado contiene archivos ANE para extensiones nativas utilizadas por la aplicación. Cada archivo ANE de este directorio tiene la extensión `.ane` de nombre de archivo. Sin embargo, el nombre de archivo antes de la extensión `.ane` *no* tiene que coincidir con el valor del elemento `extensionID` del archivo descriptor de la aplicación.

Por ejemplo, si utiliza `-extdir ./extensions`, el directorio `extensions` tendrá este aspecto:

```
extensions/  
  extension1.ane  
  extension2.ane
```

***Nota:** el uso de la opción `-extdir` es distinto en la herramienta ADT y en la herramienta ADL. En ADL, la opción especifica un directorio con subdirectorios, cada uno con un archivo ANE sin empaquetar. En ADT, las opciones especifican un directorio que contiene archivos ANE.*

Opciones de conexión del depurador

Si el destino del paquete es `apk-debug`, `ipa-debug` o `ipa-debug-interpret`, se pueden utilizar las opciones de conexión para especificar si la aplicación intentará conectarse a un depurador remoto (normalmente para depuración por wifi) o detectar una conexión entrante desde un depurador remoto (normalmente para depuración USB). Utilice la opción `-connect` para conectarse a un depurador; utilice la opción `-listen` para aceptar una conexión desde un depurador mediante una conexión USB. Estas opciones son excluyentes entre sí, es decir, no es posible usar las dos al mismo tiempo.

La opción `-connect` usa la siguiente sintaxis:

```
-connect hostString
```

-connect Si está presente, la aplicación intentará conectarse a un depurador remoto.

hostString Cadena que identifica el equipo que ejecuta la herramienta de depuración de Flash, FDB. Si no se especifica, la aplicación intentará conectarse a un depurador que se ejecute en el equipo en el que se crea el paquete. La cadena de `host` puede ser un nombre de dominio de equipo cualificado: `machinename.subgroup.example.com`, o bien, una dirección IP: `192.168.4.122`. Si el equipo especificado (o predeterminado) no se puede encontrar, el motor de ejecución mostrará un cuadro de diálogo solicitando un nombre de `host` válido.

La opción `-listen` emplea la siguiente sintaxis:

```
-listen port
```

-listen Si existe, el motor de ejecución espera una conexión desde un depurador remoto.

port (Opcional) Puerto desde el que se realiza la detección. De forma predeterminada, el motor de ejecución realiza la detección en el puerto 7936. Para obtener más información sobre el uso de la opción `-listen`, consulte [“Depuración remota con FDB a través de USB”](#) en la página 112.

Opciones de creación de perfiles de la aplicación de Android

Cuando el destino del paquete es `apk-profile`, las opciones del generador de perfiles se pueden utilizar para especificar qué archivo SWF precargado se utilizará para la creación de perfiles de memoria y rendimiento. Las opciones del generador de perfiles utilizan la siguiente sintaxis:

`-preloadSWFPath` *directory*

-preloadSWFPath Si está presente, la aplicación intentará buscar el archivo SWF de precarga en el directorio especificado. Si no se especifica, ADT incluye el archivo SWF de precarga del SDK de AIR.

directory El directorio contiene el archivo SWF de precarga del generador de perfiles.

Opciones de extensiones nativas

Las opciones de extensiones nativas especifican las opciones y los archivos para empaquetar un archivo ANE para una extensión nativa. Utilice estas opciones con un comando package de ADT en el que la opción `-target` sea `ane`.

```
extension-descriptor -swc swcPath
  -platform platformName
  -platformoptions path/platform.xml
  FILE_OPTIONS
```

extension-descriptor Archivo descriptor para la extensión nativa.

-swc Archivo SWC que contiene los recursos y el código ActionScript para la extensión nativa.

-platform Nombre de la plataforma que admite este archivo de ANE. Puede incluir varias opciones `-platform`, cada una con su propio `FILE_OPTIONS`.

-platformoptions La ruta de un archivo de opciones de plataforma (`platform.xml`). Utilice este archivo para especificar opciones de vinculación no predeterminadas, bibliotecas compartidas y bibliotecas estáticas de terceros empleadas por la extensión. Para obtener más información y ejemplos, consulte Bibliotecas nativas de iOS.

FILE_OPTIONS Identifica los archivos de plataforma nativos que se incluyen en el paquete, por ejemplo, bibliotecas estáticas con el paquete de extensión nativa. Las opciones del archivo se describen en su totalidad en “[Opciones de ruta y archivo](#)” en la página 191. (Tenga en cuenta que la opción `-e` no se puede utilizar al empaquetar un archivo ANE.)

Más temas de ayuda

[Empaquetado de una extensión nativa](#)

Mensajes de error de ADT

La siguiente tabla incluye los posibles errores que puede notificar el programa ADT y las causas probables:

Errores de validación del descriptor de la aplicación

Código de error	Descripción	Notas
100	El descriptor de la aplicación no se puede analizar.	Compruebe el archivo descriptor de la aplicación para buscar errores de sintaxis XML como, por ejemplo, etiquetas sin cerrar.
101	Falta el espacio de nombres.	Añada el espacio de nombres ausente.
102	Espacio de nombres no válido.	Compruebe la ortografía del espacio de nombres.

Código de error	Descripción	Notas
103	Elemento o atributo inesperado	Elimine los elementos y atributos erróneos. Los valores personalizados no se permiten en el archivo descriptor. Compruebe la ortografía de los nombres de atributos y elementos. Asegúrese de que los elementos se sitúen en el elemento principal correcto y que los atributos se utilicen con los elementos adecuados.
104	Falta un elemento o atributo	Añada el elemento o atributo necesario.
105	El elemento o atributo contienen un valor no válido.	Corrija el valor erróneo.
106	Combinación de atributo de ventana no válida.	Algunas configuraciones de ventana como, por ejemplo, <code>transparency = true</code> y <code>systemChrome = standard</code> no se pueden utilizar de forma conjunta. Cambie una de las configuraciones incompatibles.
107	El tamaño mínimo de ventana es superior al tamaño máximo de ventana.	Modifique la configuración de tamaño máximo o mínimo.
108	Atributo ya utilizado en el anterior elemento	
109	Elemento duplicado.	Elimine el elemento duplicado.
110	Al menos se requiere un elemento del tipo especificado.	Añada el elemento ausente.
111	Ninguno de los perfiles incluidos en el descriptor de la aplicación admiten extensiones nativas.	Añada un perfil a la lista <code>supportedProfiles</code> que admita las extensiones nativas de
112	El destino de AIR no admite extensiones nativas.	Seleccione un destino que admita extensiones nativas.
113	<code><nativeLibrary></code> y <code><initializer></code> se deben proporcionar de forma conjunta.	Se debe especificar una función inicializadora para todas las bibliotecas nativas en la extensión nativa.
114	<code><finalizer></code> encontrado sin <code><nativeLibrary></code> .	No especifique ningún finalizador a menos que la plataforma utilice una biblioteca nativa.
115	La plataforma predeterminada no debe contener ninguna implementación nativa.	No especifique ninguna biblioteca nativa en el elemento de la plataforma predeterminada.
116	Este destino no admite la invocación de navegador.	El elemento <code><allowBrowserInvocation></code> no puede ser <code>true</code> para el destino de empaquetado especificado.
117	Este destino requiere al menos un espacio de nombres <code>n</code> para empaquetar extensiones nativas.	Cambie el espacio de nombres de AIR en el descriptor de la aplicación por un valor admitido.

Consulte “[Archivos descriptores de las aplicaciones de AIR](#)” en la página 214 para obtener información sobre los espacios de nombres, elementos, atributos y sus valores válidos.

Errores de icono de la aplicación

Código de error	Descripción	Notas
200	El archivo de icono se puede abrir.	Compruebe que el archivo existe en la ruta especificada. Utilice otra aplicación para garantizar que el archivo se puede abrir.
201	El tamaño del icono no es correcto.	El tamaño del icono (en píxeles) debe coincidir con la etiqueta XML. Por ejemplo, con el siguiente elemento descriptor de la aplicación: <image32x32>icon.png</image32x32> La imagen en icon.png debe ser exactamente de 32x32 píxeles.
202	El archivo de icono contiene un formato de imagen no compatible.	Solo se admite el formato PNG. Convierta imágenes en otros formatos antes de empaquetar la aplicación.

Errores de archivo de aplicación

Código de error	Descripción	Notas
300	Falta el archivo o no se puede abrir.	No se puede encontrar o no se puede abrir un archivo especificado en la línea de comandos.
301	Falta el archivo descriptor de la aplicación o no se puede abrir.	El archivo descriptor de la aplicación no se encuentra en la ruta especificada o no puede abrirse.
302	Falta el archivo de contenido raíz en un paquete.	El archivo SWF o HTML al que se hace referencia en el elemento <content> del descriptor de la aplicación se debe añadir al paquete agregándolo a los archivos que se incluyen en la línea de comandos de ADT.
303	Falta el archivo de icono en el paquete.	Los archivos de icono especificado en el descriptor de la aplicación se deben añadir al paquete agregándolos a los archivos que se incluyen en la línea de comandos de ADT. Los archivos de icono no se agregan automáticamente.
304	El contenido de ventana inicial no es válido.	El archivo al que se hace referencia en el elemento <content> del descriptor de la aplicación no se reconoce como archivo HTML o SWF válido.

Código de error	Descripción	Notas
305	La versión de SWF del contenido de ventana inicial sobrepasa la versión del espacio de nombres.	La versión de SWF del archivo al que se hace referencia en el elemento <content> del descriptor de la aplicación no se admite en la versión de AIR especificada en el espacio de nombres del descriptor. Por ejemplo, si se intenta empaquetar un archivo SWF10 (Flash Player 10) como contenido inicial de una aplicación de AIR 1.1, se generará este error.
306	Perfil no admitido.	El perfil que está especificando en el archivo descriptor de la aplicación no se admite. Consulte " supportedProfiles " en la página 250.
307	El espacio de nombres debe ser al menos <i>nnn</i> .	Utilice el espacio de nombres adecuado para las funciones utilizadas en la aplicación (por ejemplo, el espacio de nombres 2.0).

Códigos de salida para otros errores

Código de salida	Descripción	Notas
2	Error de uso	Busque errores en los argumentos de la línea de comandos.
5	Error desconocido.	Este error indica una situación que no se puede explicar con condiciones de error comunes. Entre las posibles causas de raíz se incluyen la incompatibilidad entre ADT y el entorno de ejecución de Java (JRE), instalaciones dañadas de ADT o JRE y errores de programación en ADT.
6	No se puede en el directorio de salida.	Compruebe que se pueda acceder al directorio de salida especificado (o implicado) y que la unidad que lo contiene tenga suficiente espacio en disco.
7	No se puede acceder al certificado.	Compruebe que la ruta al almacén de claves se especifique correctamente. Compruebe que se pueda acceder al certificado del almacén de claves. La utilidad Java 1.6 Keytool se puede utilizar para ayudar a solucionar problemas de acceso a certificados.
8	Certificado no válido.	El archivo del certificado tiene un formato incorrecto, se ha modificado, ha caducado o se ha revocado.
9	No se pudo firmar un archivo de AIR	Compruebe las opciones de firma transmitidas a ADT.
10	No se ha podido crear la marca de hora.	ADT no ha podido establecer una conexión con el servidor de marcas de hora. Si se conecta a Internet mediante un servidor proxy, puede que sea necesario establecer la configuración de proxy de JRE.

Código de salida	Descripción	Notas
11	Error de creación del certificado.	Compruebe los argumentos de la línea de comandos para crear firmas.
12	Entrada no válida.	Compruebe las rutas de archivo y otros argumentos transmitidos a ADT en la línea de comandos.
13	Falta el SDK del dispositivo	Compruebe la configuración del SDK del dispositivo. ADT no puede localizar el SDK del dispositivo necesario para ejecutar el comando especificado.
14	Error del dispositivo	ADT no puede ejecutar el comando debido a un problema o restricción del dispositivo. Por ejemplo, el código de salida se emite al intentar desinstalar una aplicación que no esté instalada realmente.
15	Sin dispositivos	Compruebe que un dispositivo esté conectado y activado o que se esté ejecutando un emulador.
16	Componentes GPL que faltan	El SDK de AIR actual no incluye todos los componentes necesarios para realizar la operación solicitada.
17	La herramienta de empaquetado del dispositivo ha fallado.	El paquete no se ha podido crear porque faltan componentes del sistema operativo que se esperaban.

Errores de Android

Código de salida	Descripción	Notas
400	La versión actual del sdk de Android no admite el atributo.	Compruebe que el nombre del atributo se haya escrito correctamente y que sea un atributo válido para el elemento en el que aparece. Puede que sea necesario establecer el indicador -platformsdk en el comando ADT si el atributo se introdujo tras Android 2.2.
401	La versión actual del sdk de Android no admite ningún valor de atributo	Compruebe que el valor de atributo se haya escrito correctamente y que sea un valor válido para el atributo. Puede que sea necesario establecer el indicador -platformsdk en el comando ADT si el atributo se introdujo tras Android 2.2.
402	La versión actual del sdk de Android no admite ninguna etiqueta XML	Compruebe que el nombre de la etiqueta XML se haya escrito correctamente y que sea un elemento del documento de manifiesto de Android válido. Puede que sea necesario establecer el indicador -platformsdk en el comando ADT si el atributo se introdujo tras Android 2.2.

Código de salida	Descripción	Notas
403	No se permite la omisión de la etiqueta Android	La aplicación está intentando omitir un elemento de manifiesto de Android que se reserva para su uso en AIR. Consulte "Configuración de Android" en la página 81.
404	No se permite la omisión del atributo Android	La aplicación está intentando omitir un atributo de manifiesto de Android que se reserva para su uso en AIR. Consulte "Configuración de Android" en la página 81.
405	La etiqueta %1 de Android debe ser el primer elemento en la etiqueta manifestAdditions	Mueva la etiqueta especificada a la ubicación requerida.
406	El atributo %1 de la etiqueta de Android %2 tiene un valor %3 no válido.	Proporcione un valor válido para el atributo.

Variables del entorno de ADT

ADT lee los valores de las siguientes variables de entorno (si se establecen):

AIR_ANDROID_SDK_HOME especifica la ruta al directorio raíz del SDK de Android (el directorio que contiene la carpeta de herramientas). El SDK de AIR 2.6+ incluye las herramientas del SDK de Android necesarias para implementar los comandos ADT pertinentes. Únicamente establezca este valor para utilizar una versión diferente del SDK de Android. Si se establece esta variable, no es necesario especificar la opción `-platformsdk` al ejecutar los comandos de ADT que se requieran. Si se establecen tanto esta variable como la opción de la línea de comandos, se usará la ruta especificada en la línea de comandos.

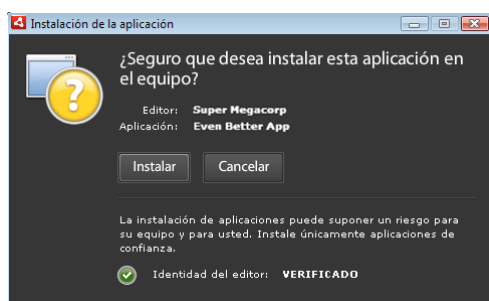
AIR_EXTENSION_PATH especifica una lista de directorios para buscar extensiones nativas requeridas por la aplicación. La lista de directorios se busca en orden según los directorios de extensiones nativas especificados en la línea de comandos de ADT. El comando ADL también utiliza esta variable de entorno.

Nota: en algunos sistemas informáticos, los caracteres de doble byte de las rutas del sistema de archivos almacenados en estas variables de entorno se pueden malinterpretar. Si esto sucede, intente establecer JRE utilizado para ejecutar ADT para que se utilice el conjunto de caracteres UTF-8. Esto se realiza de forma predeterminada en el script utilizado para iniciar ADT en Mac y Linux. En el archivo Windows `adt.bat`, o cuando ADT se ejecuta directamente desde Java, especifique la opción `-Dfile.encoding=UTF-8` en la línea de comandos de Java.

Capítulo 13: Firma de aplicaciones de AIR

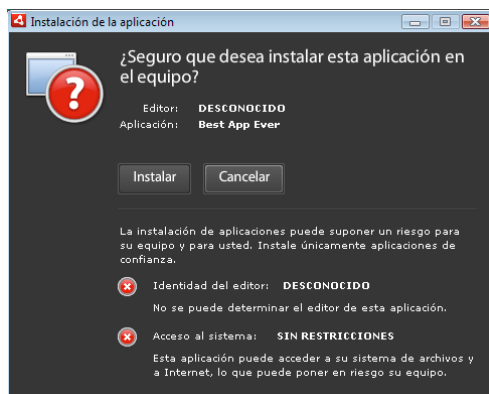
Firma digital de archivos de AIR

La firma digital de sus archivos de instalación de AIR con un certificado proporcionado por una entidad emisora de certificados (AC) reconocida ofrece seguridad a los usuarios de que la aplicación que están instalando no ha sido alterada ni de modo accidental ni malintencionado e identifica a usted como el firmante (editor). Si la aplicación de AIR está firmada con un certificado de confianza, o está *encadenada* con un certificado de confianza en el ordenador de instalación, AIR presenta el nombre del editor durante la instalación.



Cuadro de diálogo de confirmación de instalación firmado por un certificado de confianza

Si una aplicación se firma con un certificado con firma automática (o un certificado que no se vincula a un certificado de confianza), el usuario debe asumir un mayor riesgo de seguridad al instalar la aplicación. El cuadro de diálogo de instalación refleja este riesgo adicional:



Cuadro de diálogo de confirmación de instalación firmado por un certificado con firma automática

Importante: una entidad malintencionada podría falsificar un archivo de AIR con su identidad si lograra obtener su archivo de almacén de claves para la firma descubre si clave privada.

Certificados con firma de código

Las garantías de seguridad, limitaciones y obligaciones legales respecto del uso de certificados de firma de código se reseñan en la declaración de prácticas de certificación (DPC) y los acuerdos de suscriptor publicados por la autoridad de certificación emisora. Para obtener más información sobre los acuerdos para las entidades emisoras de certificados que actualmente proporcionan certificados de firma de código de AIR, consulte:

[ChosenSecurity](http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm) (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm) (en inglés)

[ChosenSecurity CPS](http://www.chosensecurity.com/resource_center/repository.htm) (http://www.chosensecurity.com/resource_center/repository.htm)

[GlobalSign](http://www.globalsign.com/code-signing/index.html) (<http://www.globalsign.com/code-signing/index.html>)

[GlobalSign CPS](http://www.globalsign.com/repository/index.htm) (<http://www.globalsign.com/repository/index.htm>) (en inglés)

[Thawte CPS](http://www.thawte.com/cps/index.html) (<http://www.thawte.com/cps/index.html>) (en inglés)

[VeriSign CPS](http://www.verisign.com/repository/CPS/) (<http://www.verisign.com/repository/CPS/>) (en inglés)

[Acuerdo de suscripción de Verisign](https://www.verisign.es/repository/subscriber/SUBAGR.html) (<https://www.verisign.es/repository/subscriber/SUBAGR.html>)

Firma de códigos de AIR

Cuando se firma un archivo de AIR se incluye una firma digital en el archivo de instalación. La firma incluye un resumen del paquete, el cual se utiliza para verificar que el archivo de AIR no se haya alterado desde que se firmó, e incluye información acerca del certificado de firma, que sirve para verificar la identidad del editor.

AIR utiliza la infraestructura de claves públicas (PKI), compatible a través del almacén de certificados del sistema operativo, para establecer si un certificado es de confianza. El ordenador en el que se instala una aplicación de AIR debe o bien confiar directamente en el certificado que se utiliza para firmar la aplicación de AIR, o bien confiar en una serie de certificados que vinculan el certificado con una entidad emisora de certificados de confianza para que se pueda verificar la información del editor.

Si se firma un archivo de AIR con un certificado que no está encadenado con uno de los certificados raíz de confianza (que normalmente incluye a todos los certificados con firma automática), no se puede verificar la información del editor. Si bien AIR puede determinar que un paquete de AIR no ha sido alterado desde que se firmó, no hay manera de saber quién creó y firmó el archivo.

***Nota:** un usuario puede optar por confiar en un certificado con firma automática y en adelante las aplicaciones de AIR firmadas con el certificado presentarán el valor del campo de nombre común en el certificado como nombre del editor. AIR no proporciona ningún mecanismo para que un usuario designe un certificado como “de confianza”. El certificado (sin incluir la clave privada) debe proporcionarse por separado al usuario, quien deberá emplear uno de los mecanismos suministrados con el sistema operativo o una herramienta adecuada para importar el certificado en el lugar correcto en el almacén de certificados del sistema.*

Identificador del editor de AIR

***Importante:** a partir de AIR 1.5.3, el ID de editor queda desfasado y no se vuelve a calcular en función del certificado de firma de código. Las nuevas aplicaciones no necesitan ni deben utilizar un ID de editor. Al actualizar las aplicaciones existentes, debe especificar el ID de editor original en el archivo descriptor de la aplicación.*

Antes de AIR 1.5.3, el instalador de aplicaciones de AIR generó un ID de editor durante la instalación de un archivo de AIR. Se trata de un identificador exclusivo del certificado que se utiliza para firmar el archivo de AIR. Si se vuelve a utilizar el mismo certificado para varias aplicaciones de AIR, todas tendrán el mismo ID de editor. La firma de una actualización de la aplicación con un certificado diferente y en ocasiones incluso con una instancia renovada del certificado original cambió el ID de editor.

En AIR 1.5.3 y versiones posteriores, un ID de editor no se asigna mediante AIR. Una aplicación publicada con AIR 1.5.3 puede especificar una cadena de ID de editor en el descriptor de la aplicación. Únicamente es necesario especificar un ID de editor cuando se publiquen actualizaciones para aplicaciones publicadas en un principio para versiones de AIR antes de 1.5.3. Si no se especifica el ID original en el descriptor de la aplicación, el nuevo paquete de AIR no se tratará como una actualización de la aplicación existente.

Para determinar el ID de editor original, localice el archivo `publisherid` en el subdirectorio META-INF/AIR donde se instaló la aplicación original. La cadena de este archivo es el ID de editor. El descriptor de la aplicación debe especificar el motor de ejecución de AIR 1.5.3 (o posterior) en la declaración del espacio de nombres del archivo descriptor de la aplicación con el fin de especificar el ID de editor manualmente.

El ID de editor, cuando se encuentra presente, se emplea para lo siguiente:

- Como parte de la clave de cifrado para el almacén local cifrado.
- Como parte de la ruta para el directorio de almacenamiento de la aplicación.
- Como parte de la cadena de conexión para conexiones locales.
- Como parte de la cadena de identidad utilizada para invocar una aplicación la API en navegador de AIR.
- Como parte de OSID (utilizado al crear programas personalizados de instalación y desinstalación).

Cuando cambia un ID de editor, el comportamiento de cualquier función de AIR basada en el ID también cambia. Por ejemplo, ya no se puede acceder a los datos del almacén local cifrado existente y cualquier instancia de Flash o AIR que cree una conexión local con la aplicación debe utilizar el nuevo ID en la cadena de conexión. El ID de editor para una aplicación instalada no puede cambiar en AIR 1.5.3 ni en versiones posteriores. Si se utiliza un ID de editor diferente al publicar un paquete de AIR, el instalador trata el nuevo paquete como una aplicación distinta en lugar de como una actualización.

Formatos de certificados

Las herramientas de firma de AIR aceptan cualquier almacén de claves accesibles a través de la arquitectura de criptografía de Java (Java Cryptography Architecture o JCA). En esto se incluyen los almacenes de claves basados en archivos, como archivos de formato PKCS12 (que suelen tener la extensión de archivo `.pfx` o `.p12`), archivos `.keystore` de Java, almacenes de claves de hardware PKCS11 y los almacenes de claves del sistema. Los formatos de almacenes de claves a los que tiene acceso ADT depende de la versión y configuración del motor de ejecución de Java que se utilice para ejecutar ADT. El acceso a algunos tipos de almacén de claves, como los token de hardware PKCS11, pueden necesitar que se instalen y configuren plugins de JCA y controladores de software adicionales.

Para firmar archivos de AIR, puede utilizar la mayoría de los certificados de firma de código existentes, o bien, puede obtener un certificado nuevo emitido expresamente para firmar aplicaciones de AIR. Se pueden utilizar, por ejemplo, cualquiera de los siguientes tipos de certificados de Verisign, Thawte, GlobalSign o ChosenSecurity:

- [ChosenSecurity](#) (en inglés)
 - TC Publisher ID para Adobe AIR
- [GlobalSign](#)
 - Certificado de firma de código ObjectSign
- [Thawte](#):
 - Certificado de AIR Developer
 - Certificado de Apple Developer
 - Certificado de JavaSoft Developer
 - Certificado de Microsoft Authenticode

- [VeriSign](#) (en inglés):
 - ID digital de Adobe AIR
 - Microsoft Authenticode Digital ID
 - Sun Java Signing Digital ID

Nota: el certificado se debe crear para realizar la firma de código. No se puede utilizar un certificado SSL ni otro tipo de certificado para firmar archivos de AIR.

Marcas de hora

Cuando se firma un archivo de AIR, la herramienta de empaquetado consulta al servidor de una autoridad de marcas de hora para obtener una fecha y hora de la firma que sean verificables independientemente. La marca de hora que se obtiene se incorpora en el archivo de AIR. Siempre y cuando el certificado de firma sea válido en el momento de firmarlo, se podrá instalar el archivo de AIR, incluso después de caducado el certificado. Por otro lado, si no se obtiene la marca de hora, el archivo de AIR dejará de poder instalarse cuando caduque o se revoque el certificado.

La opción predeterminada es que las herramientas de empaquetado de AIR obtienen una marca de hora. No obstante, para que las aplicaciones puedan empaquetarse cuando no se dispone del servicio de marcas de hora, el marcado de hora se puede desactivar. Adobe recomienda que todo archivo de AIR que se distribuya al público incluya una marca de hora.

La autoridad de marcas de hora predeterminada que utilizan las herramientas de empaquetado de AIR es Geotrust.

Obtención de un certificado

Para obtener un certificado, normalmente visitaría el sitio web de la entidad emisora de certificados y llevaría a cabo el proceso de adquisición de la empresa en cuestión. Las herramientas que se utilizan para producir el archivo del almacén de claves que necesitan las herramientas de AIR dependen del tipo de certificado que se compre, cómo se guarda el certificado en el ordenador receptor y, en algunos casos, el navegador que se utilizó para obtener el certificado. Por ejemplo, para obtener y exportar un certificado de Adobe Developer desde Thawte, debe utilizar Mozilla Firefox. El certificado podrá entonces exportarse como archivo con prefijo .pfx o .p12 directamente desde la interfaz de usuario de Firefox.

Nota: las versiones de Java 1.5 y posteriores no aceptan caracteres ASCII superior en contraseñas utilizadas para proteger archivos de certificado PKCS12. Las herramientas de desarrollo de AIR utilizan Java para crear paquetes firmados de AIR. Si el certificado se exporta como archivo .p12 o .pfx, utilice únicamente caracteres ASCII normales en la contraseña.

Se puede generar un certificado con firma automática con Air Development Tool (ADT) que se utiliza para empaquetar los archivos de instalación de AIR. También pueden utilizarse algunas herramientas de terceros.

Para ver las instrucciones sobre cómo generar un certificado autofirmado, así como instrucciones para firmar un archivo de AIR, consulte “[AIR Developer Tool \(ADT\)](#)” en la página 174. También se pueden exportar y firmar archivos de AIR con Flash Builder, Dreamweaver y la actualización de AIR para Flash.

El siguiente ejemplo describe cómo obtener un certificado de desarrollador de AIR de la autoridad de certificación Thawte y prepararlo para el uso con ADT.

Ejemplo: Cómo obtener un certificado de desarrollador de AIR con Thawte

Nota: este ejemplo muestra una sola de las diversas formas de obtener y preparar un certificado de firma de código para su uso. Cada entidad emisora de certificados cuenta con sus propias directivas y procedimientos.

Para adquirir un certificado de desarrollador de AIR, el sitio web de Thawte exige el uso del navegador Mozilla Firefox. La clave privada para el certificado se guarda en el almacén de claves del navegador. Asegúrese de que el almacén de claves de Firefox esté protegido con una contraseña maestra y que el ordenador mismo sea seguro desde el punto de vista físico. (Podrá exportar y eliminar el certificado y la clave privada del almacén de claves del navegador una vez finalizado el proceso de adquisición).

Como parte del proceso de inscripción para el certificado se genera un par de claves: pública y privada. La clave privada se guarda automáticamente en el almacén de claves de Firefox. Deberá utilizar el mismo ordenador y navegador para solicitar y recuperar el certificado del sitio web de Thawte.

- 1 Visite el sitio web de Thawte y vaya a la [página de productos para Certificados para firma de códigos](#).
- 2 En la lista de certificados para firma de códigos, seleccione el certificado de Adobe AIR Developer.
- 3 Complete el proceso de inscripción de tres pasos. Necesitará facilitar información sobre su organización, así como datos de contacto. A continuación, Thawte lleva a cabo su proceso de verificación de identidad y es posible que solicite información suplementaria. Una vez finalizada la verificación, Thawte le enviará un correo electrónico con instrucciones sobre cómo recuperar el certificado.

Nota: para obtener más información (en inglés) sobre el tipo de documentación que se requiere, haga clic en: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.

- 4 Recupere el certificado emitido del sitio de Thawte. El certificado se guarda automáticamente en el almacén de claves de Firefox.
- 5 Exporte del almacén de claves un archivo que contenga la clave privada y el certificado siguiendo los pasos que se indican a continuación:

Nota: la clave privada y el certificado de Firefox se exportan en un formato .p12 (pfx) compatible con ADT, Flex, Flash y Dreamweaver.

- a Abra el cuadro de diálogo *Administrador de certificados* de Firefox:
 - b En Windows: abra Herramientas -> Opciones -> Opciones avanzadas -> Cifrado -> Ver certificados.
 - c En Mac OS: abra Firefox -> Preferencias -> Avanzado -> Cifrado -> Ver certificados.
 - d En Linux: abra Editar -> Preferencias -> Avanzado -> Cifrado -> Ver certificados
 - e Seleccione el certificado para firma de códigos de Adobe AIR en la lista de certificados y haga clic en el botón **Hacer copia**.
 - f Escriba un nombre de archivo y el lugar al que se debe exportar el archivo del almacén de claves, y haga clic en **Guardar**.
 - g Si utilizó la contraseña maestra de Firefox, se le solicitará escribir la contraseña para el dispositivo de seguridad del software a fin de poder exportar el archivo. (Esta contraseña es de uso exclusivo en Firefox).
 - h En el cuadro de diálogo para la *contraseña para la copia de seguridad del certificado*, cree una contraseña para el archivo del almacén de claves.
Importante: esta contraseña protege el archivo del almacén de claves y se necesita cuando se utiliza el archivo para firmar aplicaciones de AIR. Conviene elegir una contraseña segura.
 - i Haga clic en OK (Aceptar). Debería recibir un mensaje confirmando la creación de la contraseña para copias de seguridad. El archivo del almacén contiene la clave privada y el certificado se guarda con una extensión .p12 (en formato PKCS12).
- 6 Utilice el archivo de almacén de claves exportado con ADT, Flash Builder, Flash Professional o Dreamweaver. Hay que utilizar la contraseña creada para el archivo cada vez que se firme una aplicación de AIR.

Importante: la clave privada y el certificado se siguen guardando en el almacén de claves de Firefox. Mientras que esto permite exportar otra copia del archivo del certificado, también facilita otro punto de acceso que debe protegerse para mantener la seguridad de su certificado y su clave privada.

Cambio de certificado

En algunos casos, es necesario cambiar el certificado utilizado para firmar actualizaciones de la aplicación de AIR. Entre estos casos se encuentran:

- Renovación del certificado de firma original.
- La actualización de un certificado con firma automática a un certificado emitido por una entidad emisora de certificados.
- El cambio de un certificado con firma automática que va a caducar por otro.
- El cambio de un certificado comercial a otro (por ejemplo, cuando cambia la identidad de la empresa).

Para que AIR reconozca un archivo de AIR como actualización, es necesario firmar tanto los archivos de AIR originales como de actualización con el mismo certificado, o bien, aplicar una firma de migración de certificados a la actualización. Una firma de migración es una segunda firma aplicada al paquete de AIR de actualización utilizando el certificado original. La firma de migración utiliza el certificado original, que establece que el firmante es el editor original de la aplicación.

Una vez instalado un archivo de AIR con una firma de migración, el nuevo certificado pasa a ser el certificado principal. Las actualizaciones posteriores no requieren una firma de migración. No obstante, las firmas de migración se deben aplicar tanto tiempo como sea posible a los usuarios que suelen omitir las actualizaciones.

Importante: se debe cambiar el certificado y aplicar una firma de migración a la actualización con el certificado original antes de que caduque. De lo contrario, los usuarios deben desinstalar su versión existente de la aplicación antes de instalar una nueva versión. Para AIR 1.5.3 o posterior, se puede aplicar una firma de migración utilizando un certificado caducado en un periodo de gracia de 365 días posteriores a la fecha de caducidad. Sin embargo, el certificado caducado no se puede utilizar para aplicar la firma de la aplicación principal.

Para cambiar el certificado:

- 1 Cree una actualización de la aplicación.
- 2 Empaquete y firme el archivo de AIR de actualización con el certificado **nuevo**.
- 3 Vuelva a firmar el archivo de AIR con el certificado **original** (con el comando `-migrate` de ADT).

Un archivo de AIR con firma de migración es, en otros aspectos, un archivo de AIR normal. Si se instala la aplicación en un sistema que no tiene la versión original, AIR la instala de la forma habitual.

Nota: antes de AIR 1.5.3, la firma de una aplicación de AIR con un certificado renovado no siempre requería una firma de migración. Con el inicio de AIR 1.5.3, una firma de migración siempre es necesaria para los certificados renovados.

Para aplicar una firma de migración, use el “Comando migrate de ADT” en la página 182, como se describe en “Firma de una versión actualizada de una aplicación de AIR” en la página 209.

Nota: El comando migrate de ADT no se puede usar con aplicaciones AIR de escritorio que incluyan extensiones nativas, ya que dichas aplicaciones están empaquetadas como instaladores nativos, no como archivos .air. Para cambiar certificados de una aplicación AIR de escritorio que incluya una extensión nativa, empaquete la aplicación utilizando el “Comando package de ADT” en la página 175 con el indicador `-migrate`.

Cambios de la identidad de la aplicación

Antes de AIR 1.5.3, la identidad de una aplicación de AIR cambiaba cuando se instalaba una actualización firmada con una firma de migración. El cambio de identidad de una aplicación tiene varias repercusiones, entre las que se incluyen:

- La nueva versión de la aplicación no tiene acceso a los datos que están en el almacén local cifrado existente.
- Cambia la ubicación del directorio de almacenamiento de la aplicación. Los datos de la ubicación anterior no se copian en el nuevo directorio. (Pero la nueva aplicación puede localizar el directorio original con base en el ID del editor anterior).
- La aplicación ya no puede abrir conexiones locales con el ID del editor anterior.
- La cadena de identidad utilizada para acceder a una aplicación desde una página web cambia.
- El OSID de la aplicación cambia. (El OSID se utiliza al escribir programas de instalación o desinstalación personalizados.)

Al publicar una actualización con AIR 1.5.3 o posterior, la identidad de la aplicación no puede cambiar. Los ID de editor y la aplicación original se deben especificar en el descriptor de la aplicación del archivo de AIR de actualización. De lo contrario, el nuevo paquete no se reconocerá como actualización.

Nota: al publicar una nueva aplicación de AIR con AIR 1.5.3 o posterior, no se debe especificar un ID de editor.

Terminología

Esta sección contiene un glosario de algunos de los principales términos que necesita conocer al tomar decisiones sobre cómo firmar la aplicación para su distribución pública.

Término	Descripción
Entidad emisora de certificados (AC)	Entidad de una red de infraestructura de claves públicas que interviene como tercero de confianza y que, en última instancia, certifica la identidad del propietario de una clave pública. Una AC emite certificados digitales firmados por su propia clave privada para constatar que ha verificado la identidad del titular del certificado.
Declaración de prácticas de certificación (DPC)	Plantea las prácticas y políticas de la entidad emisora de certificados respecto de la emisión y verificación de certificados. La DPC forma parte del contrato entre la AC y sus suscriptores y partes confiantes. Reseña también las políticas de verificación de identidad y el nivel de garantía que ofrecen los certificados emitidos.
Lista de revocación de certificados (LRC)	Lista de certificados emitidos que han sido revocados y en los cuales ya no se debe confiar. AIR comprueba la LRC en el momento de firmarse una aplicación de AIR y, si no hay marca de hora, nuevamente cuando se instala la aplicación.
Cadena de certificación	Secuencia de certificados en la que cada certificado de la cadena ha sido firmado por el certificado siguiente.
Certificado digital	Documento digital que contiene información acerca de la identidad del propietario, la clave pública del propietario y la identidad del propio certificado. Un certificado emitido por una autoridad de certificación está firmado a su vez por un certificado de propiedad de la AC emisora.
Firma digital	Mensaje o resumen cifrado que solo puede descifrarse con la clave pública de un par clave pública-clave privada. En una PKI la firma digital contiene un certificado digital (o más) que en última instancia llevan hasta la entidad emisora de certificados. Una firma digital sirve para validar que un mensaje (o un archivo informático) no ha sido alterado desde que se firmó (dentro de los límites de garantía que ofrezca el algoritmo criptográfico que se utilizó) y, suponiendo que se confía en la entidad emisora de certificados, para validar la identidad del firmante.
Almacén de claves	Base de datos que contiene certificados digitales y, en algunos casos, las claves privadas asociadas.
Java Cryptography Architecture (JCA)	Arquitectura de criptografía de Java: arquitectura ampliable para administrar y acceder a los almacenes de claves. Para obtener más información, consulte la guía de consulta de Java Cryptography Architecture (en inglés).

Término	Descripción
PKCS #11	Cryptographic Token Interface Standard (norma de interfaces de tokens criptográficos) de RSA Laboratories. Un almacén de claves basado en tokens de hardware.
PKCS #12	Personal Information Exchange Syntax Standard (norma para la sintaxis de intercambio de información personal) de RSA Laboratories. Un almacén de claves basado en archivo que suele contener una clave privada y su certificado digital asociado.
Clave privada	La mitad privada de un sistema criptográfico asimétrico que consta de clave pública y clave privada. La clave privada es confidencial y no debe nunca transmitirse a través de una red. Los mensajes con firma digital son cifrados con la clave privada por parte del firmante.
Clave pública	La mitad pública de un sistema criptográfico asimétrico que consta de clave pública y clave privada. La clave pública se distribuye libremente y se utiliza para descifrar los mensajes cifrados con la clave privada.
Infraestructura de claves públicas (PKI)	Infraestructura de claves públicas: sistema de confianza en el que las autoridades de certificación autentican la identidad de los propietarios de claves públicas. Los clientes de la red confían en los certificados digitales emitidos por una AC de confianza para verificar la identidad del firmante de un mensaje o archivo digital.
Marca de hora	Dato firmando digitalmente que contiene la fecha y hora en que se produjo un evento. ADT puede incluir en un paquete de AIR una marca de hora provista por un servidor de hora en conformidad con la norma RFC 3161 . Cuando la hay, AIR utiliza la marca de hora para establecer la validez de un certificado en el momento de firmar. Esto permite instalar una aplicación de AIR una vez caducado su certificado de firma.
Autoridad de marcas de hora	Autoridad que emite marcas de hora. Para que AIR la reconozca, la marca de hora debe estar en conformidad con la norma RFC 3161 y la firma de la marca de hora debe encadenarse con un certificado raíz de confianza en la máquina en que se instale la aplicación.

Certificados de iOS

Los certificados de firma de código emitidos por Apple se utilizan para firmar aplicaciones de iOS, incluyendo las desarrolladas con Adobe AIR. La aplicación de una firma utilizando un certificado de desarrollo de Apple es necesario para instalar una aplicación en los dispositivos de prueba. La aplicación de una firma utilizando un certificado de distribución es necesaria para distribuir la aplicación finalizada.

Para firmar una aplicación, ADT necesita acceder al certificado de firma de código y a la clave privada asociada. El propio archivo de certificado no incluye la clave privada. Se debe crear un almacén de claves en el formulario de un archivo de intercambio de información personal (Personal Information Exchange file) (.p12 o .pfx) que contenga el certificado y la clave privada. Consulte [“Conversión de un certificado de desarrollador en un archivo de almacén de claves P12”](#) en la página 207.

Generación de una solicitud de firma de certificado

Para obtener un certificado de desarrollador, puede generar una solicitud de firma de certificado para enviarlo al Portal de suministro de Apple iOS (Apple iOS Provisioning Portal).

El proceso de solicitud de firma de certificado genera un par de claves públicas y privadas. La clave privada se conserva en el equipo. Se envía la solicitud de firma que contiene la clave pública y la información de identificación a Apple, que actúa como entidad emisora de certificados. Apple firma el certificado con su propio certificado de relaciones del desarrollador en todo el mundo (World Wide Developer Relations).

Generación de una solicitud de firma de certificado en Mac OS

En Mac OS, puede utilizar la aplicación Acceso a Llaveros para generar una solicitud de firma de código. La aplicación Acceso a Llaveros se encuentra en el subdirectorio Utilidades de la carpeta Aplicaciones. Las instrucciones para generar la solicitud de firma del certificado están disponibles en el portal Apple iOS Provisioning Portal.

Generación de una solicitud de firma de certificado en Windows

Para los desarrolladores de Windows es más fácil obtener el certificado de desarrollador de iPhone directamente en un ordenador Mac. No obstante, es posible obtener un certificado también en un ordenador con Windows. En primer lugar, cree una petición de firma de certificado (archivo CSR) con OpenSSL:

- 1 Instale OpenSSL en el ordenador con Windows. (Vaya a <http://www.openssl.org/related/binaries.html>.)

Si lo desea, puede instalar también los archivos de Visual C++ 2008 Redistributable que se muestran en la página de descargas de Open SSL. (NO es necesario tener Visual C++ instalado en el ordenador.)

- 2 Abra una sesión de comandos de Windows y la línea de comandos para abrir el directorio bin de OpenSSL (por ejemplo `c:\OpenSSL\bin\`).

- 3 Para crear la clave privada, introduzca lo siguiente en la línea de comandos:

```
openssl genrsa -out mykey.key 2048
```

Guarde este archivo de clave privada. Lo utilizará más adelante.

Cuando utilice OpenSSL, tenga en cuenta los mensajes de error. Si OpenSSL genera un mensaje de error, los archivos pueden seguir produciéndose. Sin embargo, dichos archivos no se podrán utilizar. Si aparecen errores, revise la sintaxis y vuelva a ejecutar el comando.

- 4 Para crear el archivo CSR, introduzca lo siguiente en la línea de comandos:

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest -subj  
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

Reemplace los valores de dirección de correo electrónico, CN (nombre del certificado) y C (país) por los suyos propios.

- 5 Cargue el archivo CSR a Apple en el [centro de desarrollo de iPhone](#). (Consulte “Solicitud de un certificado de desarrollador de iPhone” y “Creación de un archivo de suministro”.)

Conversión de un certificado de desarrollador en un archivo de almacén de claves P12

Para crear un almacén de claves P12, se debe combinar el certificado de desarrollador de Apple y la clave privada asociada en un solo archivo. El proceso para crear el archivo del almacén de claves depende del método utilizado para generar la solicitud de firma del certificado original y del lugar donde se almacena la clave privada.

Conversión del certificado de desarrollador de iPhone en un archivo P12 en Mac OS

Una vez descargado el certificado de iPhone de Apple, expórtelo a formato de almacén de claves P12. Para hacerlo en Mac® OS:

- 1 Abra la aplicación Acceso a Llaveros (en la carpeta Aplicaciones/Utilidades).
- 2 Si aún no ha añadido el certificado al Llavero, seleccione Archivo > Importar. Vaya al archivo de certificado (archivo .cer) que ha obtenido de Apple.

- 3 Seleccione la categoría Llaves en Acceso a Llaveros.

- 4 Seleccione la clave privada asociada a su certificado de desarrollo de iPhone.

La clave privada se identifica mediante el certificado de desarrollador de iPhone público <Nombre> <Apellido> emparejado con ella.

- 5 Presione comando y haga clic en el certificado de desarrollador de (iPhone Developer) y seleccione *Export “iPhone Developer: Name...”*.

- 6 Guarde la clave en formato de archivo de intercambio de información personal (.p12).

- 7 Se solicitará que cree una contraseña que se utilice cuando el almacén de claves se use para firmar aplicaciones o transferir la clave y el certificado de este almacén de claves a otro almacén.

Conversión de un certificado de desarrollador de Apple en un archivo P12 en Windows

Para desarrollar AIR para aplicaciones de iOS, se debe utilizar un archivo de certificado P12. Podrá generar este certificado a partir del archivo de certificado de desarrollador de iPhone de Apple cuando lo reciba de Apple.

- 1 Convierta el archivo de certificador de desarrollador recibido de Apple en un archivo de certificado PEM. Ejecute la siguiente declaración en la línea de comandos desde el directorio bin de OpenSSL:

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```

- 2 Si utiliza una clave privada del Llavero de un equipo Mac, conviértala en una clave PEM:

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```

- 3 Ahora puede generar un archivo P12 válido a partir de la clave y de la versión PEM del certificado de desarrollador de iPhone:

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

Si utiliza una clave del Llavero de Mac OS, utilice la versión PEM que generó en el paso anterior. En caso contrario, utilice la clave OpenSSL generada anteriormente (en Windows).

Creación de archivos intermedios sin firmar de AIR con ADT

Utilice el comando `-prepare` para crear un archivo intermedio de AIR sin firmar. Para producir un archivo de instalación de AIR válido, el archivo intermedio de AIR debe firmarse con el comando `-sign` de ADT.

El comando `-prepare` acepta los mismos indicadores y parámetros que el comando `-package` (a excepción de las opciones de firma). La única diferencia es que no se firma el archivo de salida. El archivo intermedio se genera con la extensión del nombre de archivo `airi`.

Para firmar un archivo intermedio de AIR, utilice el comando `-sign` de ADT. (Consulte “[Comando prepare de ADT](#)” en la página 182.)

Ejemplo del comando `-prepare` de ADT

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Firma de un archivo intermedio de AIR con ADT

Para firmar un archivo intermedio de AIR con ADT, utilice el comando `-sign`. El comando "sign" solo funciona con archivos intermedios de AIR (con la extensión `airi`). Un archivo de AIR no se puede firmar por segunda vez.

Para crear un archivo intermedio de AIR, utilice el comando de ADT `-sign`. (Consulte “[Comando prepare de ADT](#)” en la página 182.)

Firma de archivos AIRI

- ❖ Utilice el comando `-sign` de ADR con la sintaxis siguiente:

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS Las opciones de firma identifican la clave privada y el certificado con que se firma el archivo de AIR. Estas opciones se describen en “[Opciones de firma de código de ADT](#)” en la página 189.

airi_file La ruta al archivo intermedio sin firmar de AIR que se va a firmar.

air_file El nombre del archivo de AIR que se va a crear.

Ejemplo del comando -sign de ADT

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Para obtener más información, consulte “[Comando sign de ADT](#)” en la página 182.

Firma de una versión actualizada de una aplicación de AIR

Cada vez que cree una versión actualizada de una aplicación de AIR existente, debe firmar la aplicación actualizada. Lo más conveniente es usar el certificado que se utilizó para firmar la versión anterior. En ese caso, la firma es como al firmar la aplicación por primera vez.

Si el certificado utilizado para firmar la anterior versión de la aplicación se hubiera renovado o sustituido después de caducar, puede usar el certificado renovado o nuevo (sustitutorio) para firmar la versión actualizada. Para ello, debe firmar la aplicación con el certificado nuevo y aplicar una firma de migración usando el certificado original. La firma de migración valida que el propietario del certificado original haya publicado la actualización.

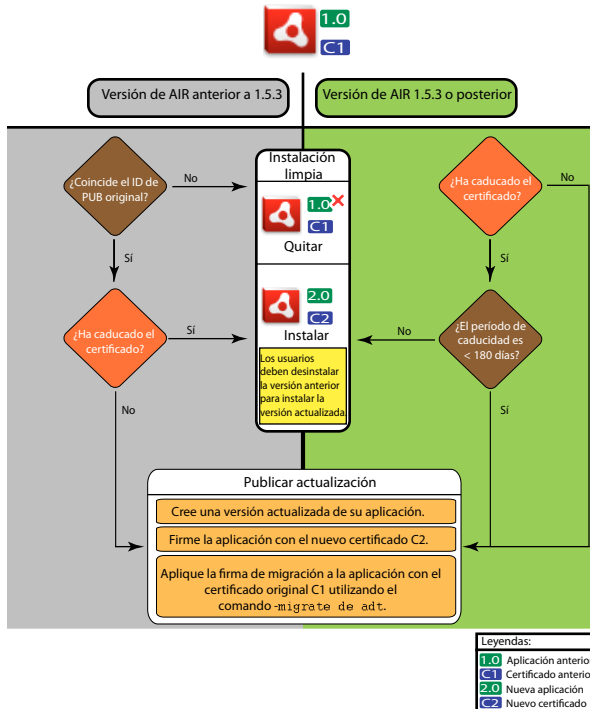
Antes de aplicar una firma de migración, se deben tener en cuenta los siguientes puntos:

- Para poder aplicar una firma de migración, el certificado original debe ser aún válido o haber caducado en los últimos 365 días. El periodo se denomina "periodo de gracia" y la duración puede cambiar en el futuro.
Nota: hasta AIR 2.6, el periodo de gracia era de 180 días.
- No es posible aplicar una firma de migración una vez que caduque el certificado y transcurra el periodo de gracia de 365 días. De ser así, los usuarios tendrán que desinstalar la versión existente antes de instalar la actualizada.
- El periodo de gracia de 365 días solo se aplica a las aplicaciones que especifican la versión de AIR 1.5.3 o posterior en el espacio de nombres del descriptor de la aplicación.

Importante: la firma de actualizaciones con firmas de migración a partir de certificados caducados es una solución temporal. Para obtener una solución amplia, cree un flujo de trabajo de firma estandarizado para administrar la implementación de actualizaciones de la aplicación. Por ejemplo, firme cada actualización con el certificado más reciente y aplique un certificado de migración usando el mismo certificado que para firmar la actualización anterior (de ser aplicable). Cargue cada actualización en su propia URL, desde la que los usuarios podrán descargar la aplicación. Para obtener más información, consulte “[Flujo de trabajo de firma para actualizaciones de la aplicación](#)” en la página 272.

En la tabla y la figura siguientes se resume el flujo de trabajo para el uso de firmas de migración:

Escenario	Estado del certificado original	Acción del desarrollador	Acción del usuario
Aplicación basada en la versión del motor de ejecución de Adobe AIR 1.5.3 o superior	Válido	Publicación de la versión más reciente de la aplicación de AIR	No se requiere ninguna acción La aplicación se actualiza automáticamente
	Caducado pero dentro del periodo de gracia de 365 días	Firme la aplicación con el certificado nuevo. Aplique una firma de migración con el certificado caducado.	No se requiere ninguna acción La aplicación se actualiza automáticamente
	Caducado y no en periodo de gracia	No se puede aplicar la firma de migración a la actualización de la aplicación de AIR. En lugar de ello, debe publicar otra versión de la aplicación de AIR utilizando un nuevo certificado. Los usuarios pueden instalar la nueva versión tras desinstalar su versión existente de la aplicación de AIR.	Desinstale la versión actual de la aplicación de AIR e instale la versión más reciente.
<ul style="list-style-type: none"> • Aplicación basada en la versión del motor de ejecución de Adobe AIR 1.5.2 o anterior • El ID de editor en el descriptor de la aplicación de la actualización coincide con el ID de editor de la versión anterior 	Válido	Publicación de la versión más reciente de la aplicación de AIR	No se requiere ninguna acción La aplicación se actualiza automáticamente
	Caducado y no en periodo de gracia	No se puede aplicar la firma de migración a la actualización de la aplicación de AIR. En lugar de ello, debe publicar otra versión de la aplicación de AIR utilizando un nuevo certificado. Los usuarios pueden instalar la nueva versión tras desinstalar su versión existente de la aplicación de AIR.	Desinstale la versión actual de la aplicación de AIR e instale la versión más reciente.
<ul style="list-style-type: none"> • Aplicación basada en la versión del motor de ejecución de Adobe AIR 1.5.2 o anterior • El ID de editor en el descriptor de la aplicación de la actualización no coincide con el ID de editor de la versión anterior 	Cualquiera	Firme la aplicación de AIR usando un certificado válido y publique la versión más reciente de la aplicación de AIR	Desinstale la versión actual de la aplicación de AIR e instale la versión más reciente.



Flujo de trabajo de firma para actualizaciones

Migración de una aplicación de AIR para utilizar un nuevo certificado

Para migrar una aplicación de AIR a un certificado nuevo mientras se actualiza la aplicación:

- 1 Cree una actualización de la aplicación.
- 2 Empaque y firme el archivo de AIR de actualización con el certificado **nuevo**.
- 3 Vuelva a firmar el archivo de AIR con el certificado **original** utilizando el comando `-migrate`.

También se puede usar un archivo AIR firmado con el comando `-migrate` para instalar una nueva versión de la aplicación, además de para actualizar cualquier versión anterior firmada con el certificado antiguo.

Nota: al actualizar una aplicación publicada para una versión de AIR anterior a 1.5.3, es necesario especificar el ID de editor original en el descriptor de la aplicación. De lo contrario, los usuarios de la aplicación deben desinstalar la versión anterior antes de la instalar la actualización.

Utilice el comando `-migrate` de ADT con la sintaxis siguiente:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** Las opciones de firma identifican la clave privada y el certificado con que se firma el archivo de AIR. Estas opciones deben identificar el certificado de firma **original** y se describen en “Opciones de firma de código de ADT” en la página 189
- **air_file_in** El archivo de AIR para la actualización, firmado con el certificado **nuevo**.
- **air_file_out** El archivo de AIR que se va a crear.

Nota: los nombres de archivo utilizados para los archivos de AIR de entrada y salida deben ser diferentes.

El ejemplo siguiente demuestra una llamada a ADT con el indicador `-migrate` para aplicar una firma de migración a una versión actualizada de una aplicación de AIR:


```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Nota: el comando `-migrate` se añadió a ADT en la versión 1.1 de AIR.

Migración de una aplicación de AIR como instalador nativo para usar un certificado nuevo

Una aplicación de AIR publicada como instalador nativo (por ejemplo, una aplicación con la extensión nativa `api`) no se puede firmar utilizando el comando `-migrate` de ADT por ser una aplicación nativa específica de plataforma, no un archivo `.air`. El procedimiento para migrar a un certificado nuevo una aplicación de AIR publicada como extensión nativa es el siguiente:

- 1 Cree una actualización de la aplicación.
- 2 Asegúrese de que en el archivo del descriptor de la aplicación (`app.xml`) la etiqueta `<supportedProfiles>` incluya tanto el perfil de escritorio como el perfil `extendedDesktop` (o elimine la etiqueta `<supportedProfiles>` del descriptor de la aplicación).
- 3 Empaquete y firme la aplicación de actualización **como archivo .air** usando el comando `-package` de ADT con el certificado **nuevo**.
- 4 Aplique el certificado de migración al archivo `.air` usando el comando `-migrate` de ADT con el certificado **original** (como se ha descrito en “[Migración de una aplicación de AIR para utilizar un nuevo certificado](#)” en la página 211).
- 5 Empaquete el archivo `.air` en un instalador nativo usando el comando `-package` de ADT con el indicador `-target native`. No hace falta especificar un certificado de firma en este paso porque la aplicación ya está firmada.

El ejemplo siguiente ilustra los pasos 3-5 de este proceso. El código llama a ADT con el comando `-package`, llama a ADT con el comando `-migrate` y vuelve a llamar a ADT con el comando `-package` para empaquetar una versión actualizada de una aplicación de AIR como instalador nativo:

```
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
```

Migración de una aplicación de AIR que usa una extensión nativa para usar un nuevo certificado

Una aplicación de AIR que usa una extensión nativa no se puede firmar usando un comando `-migrate` de ADT. Tampoco se puede migrar mediante el procedimiento usado para una aplicación de AIR de instalador nativo, ya que no se puede publicar como archivo `.air` intermedio. El procedimiento para migrar a un certificado nuevo una aplicación de AIR que usa una extensión nativa es el siguiente:

- 1 Cree una actualización de la aplicación.
- 2 Empaquete y firme el instalador nativo de la actualización usando el comando `-package` de ADT. Empaquete la aplicación con el certificado **nuevo** e incluya el indicador `-migrate` especificando el certificado **original**.

Use la siguiente sintaxis para llamar al comando `-package` de ADT con el indicador `-migrate`:

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** Las opciones de firma identifican la clave privada y el certificado con que se firma el archivo de AIR. Estas opciones identifican el certificado de firma **nuevo** y se describen en “[Opciones de firma de código de ADT](#)” en la página 189.

- **MIGRATION_SIGNING_OPTIONS** Las opciones de firma identifican la clave privada y el certificado con que se firma el archivo de AIR. Estas opciones identifican el certificado de firma **original** y se describen en “[Opciones de firma de código de ADT](#)” en la página 189.
- Las otras opciones son las mismas que las usadas para empaquetar una aplicación de AIR como instalador nativo y se describen en “[Comando package de ADT](#)” en la página 175.

El ejemplo siguiente demuestra una llamada a ADT con el comando `-package` y el indicador `-migrate` para empaquetar una versión actualizada de una aplicación de AIR que usa una extensión nativa y aplicar una firma de migración a la actualización:

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Nota: El indicador `-migrate` del comando `-package` está disponible en ADT a partir de AIR 3.6.

Creación de certificados con firma automática con ADT

Los certificados con firma automática se pueden utilizar para generar un archivo de instalación de AIR válido. No obstante, estos certificados solo proporcionan garantías de seguridad limitadas para los usuarios. La autenticidad de los certificados con firma automática no se puede verificar. Cuando se instala un archivo de AIR con firma automática, los datos del editor se presentan al usuario como "Desconocidos". Un certificado generado por ADT tiene una validez de cinco años.

Si se crea una actualización para una aplicación de AIR que se firmó con un certificado autogenerado, habrá que utilizar el mismo certificado para firmar tanto los archivos de AIR originales como los de la actualización. Los certificados que produce ADT son siempre únicos, aunque se utilicen los mismos parámetros. Por lo tanto, si desea firmar automáticamente las actualizaciones con un certificado generado por ADT, conserve el certificado original en un lugar seguro. Por otra parte, no podrá producir un archivo de AIR actualizado después de haber caducado el certificado generado por ADT. (Sí podrá publicar nuevas aplicaciones con otro certificado, pero no las nuevas versiones de la misma aplicación).

Importante: *dadas las limitaciones de los certificados con firma automática, Adobe recomienda enfáticamente utilizar un certificado comercial procedente de una entidad emisora de certificados de renombre para firmar aplicaciones de AIR que se distribuyen de forma pública.*

El certificado y la clave privada asociada generados por ADT se guardan en un archivo de almacén de claves del tipo PKCS12. La contraseña especificada se establece en la propia clave y no en el almacén de claves.

Ejemplos de generación de certificados

```
adt -certificate -cn SelfSign -ou QE -o "Example, Co" -c US 2048-RSA newcert.p12 39#wnetx3t1
adt -certificate -cn ADigitalID 1024-RSA SigningCert.p12 39#wnetx3t1
```

Cuando se utilizan estos certificados para firmar archivos de AIR, hay que usar las siguientes opciones de firma con los comandos `-package` o `-prepare` de ADT:

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3t1
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3t1
```

Nota: *las versiones de Java 1.5 y posteriores no aceptan caracteres ASCII superior en contraseñas utilizadas para proteger archivos de certificado PKCS12. Utilice únicamente caracteres ASCII normales en la contraseña.*

Capítulo 14: Archivos descriptores de las aplicaciones de AIR

Todas las aplicaciones de AIR requieren un archivo descriptor de la aplicación. El archivo descriptor de la aplicación es un documento XML que define las propiedades básicas de la aplicación.

Muchos entornos de desarrollo que admiten AIR generan automáticamente un descripción de la aplicación cuando se crea un proyecto. De lo contrario, debe crear su propio archivo descriptor. Un archivo descriptor de ejemplo, `descriptor-sample.xml`, se puede encontrar en el directorio `samples` de los kits de desarrollo de software (SDK) de AIR y Flex.

El archivo descriptor puede tener cualquier nombre de archivo. Al combinar la aplicación en un paquete, el nombre del archivo descriptor de la aplicación cambia a `application.xml` y el archivo se coloca en un directorio especial en el paquete de AIR.

Descriptor de la aplicación de ejemplo

El siguiente documento descriptor de la aplicación establece las propiedades básicas utilizadas por la mayoría de aplicaciones de AIR:

```
<?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>
```

Si la aplicación utiliza un archivo HTML como su contenido raíz en lugar de un archivo SWF, únicamente el elemento `<content>` es diferente:

```
<content>
  HelloWorld.html
</content>
```

Cambios en el descriptor de la aplicación

El descriptor de la aplicación de AIR ha cambiado en las siguientes versiones de AIR.

Cambios del descriptor de AIR 1.1

Los elementos `name` y `description` permitidos de la aplicación se localizarán utilizando el elemento `text`.

Cambios del descriptor de AIR 1.5

`contentType` pasó a ser un elemento necesario de `fileType`.

Cambios del descriptor de AIR 1.5.3

Se ha añadido el elemento `publisherID` para que las aplicaciones especifiquen un valor de ID del editor.

Cambios del descriptor de AIR 2.0

Añadidos:

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (consulte `imageNxN`)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

Cambios del descriptor de AIR 2.5

Eliminado: `version`

Añadidos:

- `android`
- `extensionID`
- `extensions`
- `image36x36` (consulte `imageNxN`)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

Cambios del descriptor de AIR 2.6

Añadidos:

- `image114x114` (consulte `imageNxN`)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Cambios del descriptor de AIR 3.0

Añadidos:

- `colorDepth`
- `direct` como valor válido de `renderMode`
- `renderMode` ya no se omite en plataformas de escritorio
- Se puede especificar el elemento `<uses-sdk>` de Android. (Previamente no era posible.)

Cambios del archivo descriptor de AIR 3.1

Añadidos:

- “`Entitlements`” en la página 229

Cambios del descriptor de AIR 3.2

Añadidos:

- `depthAndStencil`
- `supportedLanguages`

Cambios del descriptor de AIR 3.3

Añadidos:

- `aspectRatio` ahora incluye la opción `ANY`.

Cambios del archivo descriptor de AIR 3.4

Añadidos:

- `image50x50` (consulte “`imageNxN`” en la página 237)
- `image58x58` (consulte “`imageNxN`” en la página 237)
- `image100x100` (consulte “`imageNxN`” en la página 237)
- `image1024x1024` (consulte “`imageNxN`” en la página 237)

Cambios del archivo descriptor de AIR 3.6

Añadidos:

- “`requestedDisplayResolution`” en la página 247 en el elemento “`iPhone`” en la página 241 incluye ahora un atributo `excludeDevices` que permite especificar qué destinos de iOS utilizan resolución estándar o alta

- el nuevo elemento “`requestedDisplayResolution`” en la página 247 en “`initialWindow`” en la página 238 especifica si se debe usar resolución estándar o alta en plataformas de escritorio como, por ejemplo, ordenadores Mac con pantallas de alta resolución

Cambios del archivo descriptor de AIR 3.7

Añadidos:

- Ahora, el elemento “`iPhone`” en la página 241 proporciona un elemento “`externalSwfs`” en la página 230, que permite especificar una lista de los archivos SWF que se deben cargar en tiempo de ejecución.
- Ahora, el elemento “`iPhone`” en la página 241 proporciona un elemento “`forceCPURenderModeForDevices`” en la página 234, que permite forzar el modo de representación de CPU para un conjunto de dispositivos específicos.

Estructura del archivo descriptor de la aplicación

El archivo descriptor de la aplicación es un documento XML con la siguiente estructura:

```
<application xmlns="http://ns.adobe.com/air/application/3.0">
  <allowBrowserInvocation>...</allowBrowserInvocation>
  <android>
    <colorDepth>...</colorDepth>
    <manifestAdditions>
      <manifest>...</manifest>
    ]]>
  </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" en la página 249>...</"supportedLanguages" en la página 249>
<supportedProfiles>...</supportedProfiles>
<versionNumber>...</versionNumber>
<versionLabel>...</versionLabel>
</application>
```

Elementos del descriptor de la aplicación de AIR

El siguiente diccionario de elementos describe cada uno de los elementos válidos de un archivo descriptor de la aplicación de AIR.

allowBrowserInvocation

Adobe AIR 1.0 y posterior: Opcional

Permite que la API en navegador de AIR detecte e inicie la aplicación.

Si este valor se define en `true`, asegúrese de tener en cuenta las posibles consecuencias para la seguridad. Estos aspectos se describen en [Invocación de una aplicación de AIR desde el navegador](#) (para desarrolladores de ActionScript) e [Invocación de una aplicación de AIR desde el navegado](#) (para desarrolladores de HTML).

Para obtener más información, consulte [“Inicio desde el navegador de una aplicación de AIR instalada”](#) en la página 268.

Elemento principal: [“application”](#) en la página 219

Elementos secundarios: ninguno

Contenido

true o false (predeterminado)

Ejemplo

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 y posterior: Opcional

Permite añadir elementos al archivo de manifiesto de Android. AIR crea el archivo AndroidManifest.xml para cada paquete APK. El elemento android en el descriptor de la aplicación de AIR se puede utilizar para añadir elementos adicionales al mismo. Se omite en todas las plataformas excepto en Android.

Elemento principal: [“application”](#) en la página 219

Elementos secundarios:

- [“colorDepth”](#) en la página 224
- [“manifestAdditions”](#) en la página 242

Contenido

Elementos que define las propiedades específicas de Android para añadirse en el manifiesto de la aplicación de Android.

Ejemplo

```
<android>
  <manifestAdditions>
    ...
  </manifestAdditions>
</android>
```

Más temas de ayuda

[“Configuración de Android”](#) en la página 81

[Archivo AndroidManifest.xml](#)

application

Adobe AIR 1.0 y posterior: Obligatorio

Elemento raíz de un documento descriptor de la aplicación de AIR.

Elemento principal: ninguno

Elementos secundarios:

- [“allowBrowserInvocation”](#) en la página 218
- [“android”](#) en la página 219
- [“copyright”](#) en la página 226
- [“customUpdateUI”](#) en la página 226
- [“description”](#) en la página 227
- [“extensions”](#) en la página 230
- [“filename”](#) en la página 231
- [“fileTypes”](#) en la página 232
- [“icon”](#) en la página 235
- [“id”](#) en la página 236
- [“initialWindow”](#) en la página 238
- [“installFolder”](#) en la página 240
- [“iPhone”](#) en la página 241
- [“name”](#) en la página 244
- [“programMenuFolder”](#) en la página 246
- [“publisherID”](#) en la página 246
- [“supportedLanguages”](#) en la página 249
- [“supportedProfiles”](#) en la página 250
- [“version”](#) en la página 252
- [“versionLabel”](#) en la página 253
- [“versionNumber”](#) en la página 253

Atributos

`minimumPatchLevel`: nivel de revisión mínimo del motor de ejecución de AIR que requiere esta aplicación.

`xmlns`: el atributo del espacio de nombres XML determina la versión del motor de ejecución de AIR necesaria de la aplicación.

El espacio de nombres cambia con cada edición principal de AIR (pero no con las revisiones menores). El último segmento del espacio de nombres, por ejemplo “3.0”, indica la versión del motor de ejecución que requiere la aplicación.

Los valores `xmlns` para las versiones principales de AIR son:

```
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"
```

Para las aplicaciones basadas en SWF, la versión del motor de ejecución de AIR especificada en el descriptor de la aplicación determina la versión de SWF máxima que se puede cargar como contenido inicial de la aplicación. Las aplicaciones que especifican AIR 1.0 o 1.1 solo pueden utilizar archivos SWF9 (Flash Player 9) como contenido inicial, aunque se ejecuten utilizando el motor de ejecución de AIR 2. Las aplicaciones que especifican AIR 1.5 (o posterior) pueden usar archivos SWF9 o SWF10 (Flash Player 10) como contenido inicial.

La versión de SWF determina qué versión de las API de AIR y Flash Player están disponibles. Si un archivo SWF9 se utiliza como contenido inicial de una aplicación de AIR 1.5, esa aplicación solo tendrá acceso a las API de AIR 1.1 y Flash Player 9. Asimismo, no se aplicarán los cambios de comportamiento realizados en las API existentes en AIR 2.0 o Flash Player 10.1. (Los cambios importantes relacionados con la seguridad en las API constituyen una excepción en este principio y se pueden aplicar de forma retroactiva en revisiones actuales o posteriores del motor de ejecución.)

Para las aplicaciones basadas en HTML, la versión del motor de ejecución especificada en el descriptor de la aplicación determina qué versión de las API de AIR y Flash Player están disponibles en la aplicación. Los comportamientos de HTML, CSS y JavaScript siempre están determinados por la versión del Webkit utilizado en el motor de ejecución de AIR instalado, y no por el descriptor de la aplicación.

Si una aplicación de AIR carga contenido SWF, la versión de las API de AIR y Flash Player disponible para dicho contenido depende del modo en que se cargue el contenido. En ocasiones la versión eficaz se determina mediante el espacio de nombres del descriptor de la aplicación, otras veces mediante la versión del contenido de carga y a veces mediante la versión del contenido cargado. La siguiente tabla muestra cómo la versión de la API está determinada según el método de carga:

Forma de carga del contenido	Forma en que la versión de la API se ve determinada
Contenido inicial, aplicación basada en SWF	Versión SWF del archivo cargado
Contenido inicial, aplicación basada en HTML	Espacio de nombres del descriptor de la aplicación
SWF cargado mediante contenido SWF	Versión del contenido de carga

Forma de carga del contenido	Forma en que la versión de la API se ve determinada
Biblioteca SWF cargada mediante contenido HTML utilizando la etiqueta <script>	Espacio de nombres del descriptor de la aplicación
SWF cargado mediante contenido HTML utilizando las API de AIR o Flash Player (por ejemplo, flash.display.Loader)	Espacio de nombres del descriptor de la aplicación
SWF cargado mediante contenido HTML utilizando etiquetas <object> o <embed> (o las API equivalentes de JavaScript)	Versión SWF del archivo cargado

Al cargar un archivo SWF de una versión distinta al contenido de carga, puede encontrarse con los siguientes problemas:

- Carga de un SWF de versión reciente mediante un SWF de versión anterior. Las referencias a las API añadidas a las versiones más recientes de AIR y Flash Player en el contenido cargado quedarán si resolver
- Carga de un SWF de versión anterior mediante un SWF de versión reciente. Las API modificadas en las versiones más recientes de AIR y Flash Player pueden tener un comportamiento no previsto por el contenido cargado.

Contenido

El elemento de la aplicación contiene los elementos secundarios que define las propiedades de una aplicación de AIR.

Ejemplo

```
<?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>
  </icon>
</application>
```

```
<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 y posterior, iOS and Android: opcional

Especifica la relación de aspecto de la aplicación.

Si no se especifica, la aplicación se abre en la proporción de aspecto “natural” del dispositivo. La orientación natural varía de dispositivo a dispositivo. Generalmente, esta es la proporción de aspecto vertical en los dispositivos de pantalla pequeña como, por ejemplo, teléfonos. En algunos dispositivos como, por ejemplo, el tablet iPad, la aplicación se abre en la orientación actual. En AIR 3.3 y versiones posteriores, esto se aplica a toda la aplicación, no solo a la visualización inicial.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

portrait, landscape o any

Ejemplo

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

autoOrients

Adobe AIR 2.0 y posterior, iOS and Android: opcional

Especifica si la orientación del contenido se ajusta automáticamente al cambiar la orientación física del dispositivo. Para obtener más información, consulte [Orientación del escenario](#).

Al utilizar la orientación automática, considere el establecimiento de las propiedades `align` y `scaleMode` del objeto Stage en los siguientes parámetros:

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Esta configuración permite que la aplicación gire sobre la esquina superior izquierda y evita que el contenido de la aplicación se escale automáticamente. Mientras los demás modos de escala se ajustan al contenido para que se ajuste a las dimensiones del escenario girado, también recortan, distorsionan o reducen excesivamente el contenido. Los mejores resultados se pueden obtener casi siempre volviendo a dibujar o a diseñar por sí mismo el contenido.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

true o false (predeterminado)

Ejemplo

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

colorDepth

Adobe AIR 3 y posterior: Opcional

Especifica si se utiliza color de 16 bits o 32 bits.

Con color de 16 bits se puede aumentar el rendimiento de procesamiento a costa de la fidelidad de los colores. Hasta AIR 3, el color de 16 bits siempre se ha utilizado en Android. En AIR 3, se utiliza color de 32 bits de forma predeterminada.

***Nota:** si la aplicación usa la clase `StageVideo`, debe utilizar color de 32 bits.*

Elemento principal: “[android](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Uno de los valores siguientes:

- 16 bits
- 32 bits

Ejemplo

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

containsVideo

Especifica si la aplicación debe contener cualquier contenido de vídeo o no.

Elemento principal: “[android](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Uno de los valores siguientes:

- true
- false

Ejemplo

```
<android>  
  <containsVideo>true</containsVideo>  
  <manifestAdditions>...</manifestAdditions>  
</android>
```

content

Adobe AIR 1.0 y posterior: Obligatorio

El valor especificado para el elemento `content` es la URL para el archivo principal de contenido de la aplicación. Este será un archivo SWF o un archivo HTML. La URL se especifica relativa a la raíz de la carpeta de instalación de la aplicación. (Al ejecutar una aplicación de AIR con ADL, la URL se indica relativa a la carpeta que contiene el archivo descriptor de la aplicación. Se puede utilizar el parámetro `root-dir` de ADL para especificar otro directorio raíz).

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Una URL relativa al directorio de la aplicación. al tratarse como una URL el valor del elemento `content`, los caracteres del nombre del archivo de contenido deben codificarse como URL de acuerdo con las reglas definidas en [RFC 1738](#). Los caracteres de espacio, por ejemplo, deben codificarse como `%20`.

Ejemplo

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0 a 1.1: opcional; AIR 1.5 y versiones posteriores: necesario

`contentType` es necesario desde AIR 1.5 (era opcional en AIR 1.0 y 1.1). La propiedad ayuda a algunos sistemas operativos a localizar la mejor aplicación para abrir un archivo. El valor debe ser el tipo MIME del contenido del archivo. Se debe tener en cuenta que el valor se omite en Linux si el tipo de archivo ya se ha registrado y dispone de un tipo MIME asignado.

Elemento principal: “[fileType](#)” en la página 232

Elementos secundarios: ninguno

Contenido

Subtipo y tipo MIME. Consulte [RFC2045](#) para obtener más información sobre los tipos MIME.

Ejemplo

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 y posterior: Opcional

Información de copyright para la aplicación de AIR. En Mac OS el aviso de copyright aparece en el cuadro de diálogo Acerca de para la aplicación instalada. En Mac OS, la información de copyright también se utiliza en el campo NSHumanReadableCopyright del archivo Info.plist para la aplicación.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Cadena que contiene la información de derechos de autor de la aplicación.

Ejemplo

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 y posterior: Opcional

Indica si una aplicación proporcionará sus propios cuadros de diálogo de actualización. Si el valor es `false`, AIR presenta cuadros diálogo de actualización estándar para el usuario. Solo las aplicaciones distribuidas como archivos de AIR pueden utilizar el sistema de actualización integrado de AIR.

Cuando la versión instalada de la aplicación tiene el elemento `customUpdateUI` definido en `true` y el usuario hace doble clic en el archivo de AIR para una nueva versión o instala una actualización de la aplicación utilizando la función de instalación integrada, el motor de ejecución abre la versión instalada de la aplicación. El motor de ejecución no abre el instalador de aplicaciones de AIR predeterminado. La lógica de la aplicación determina entonces cómo continuar con la operación de actualización. (Para que se lleve a cabo una actualización, el ID de la aplicación y el ID del editor que figuran en el archivo de AIR deben coincidir con los de la aplicación instalada).

***Nota:** el mecanismo `customUpdateUI` solo entra en juego si la aplicación ya está instalada y el usuario hace doble clic en el archivo de instalación de AIR que contiene una actualización, o si instala una actualización de la aplicación utilizando la función de instalación integrada. Puede descargar una actualización e iniciarla a través de la lógica de su propia aplicación, visualizando la interfaz de usuario personalizada según convenga, esté o no `customUpdateUI` definido en `true`.*

Para obtener más información, consulte “[Actualización de aplicaciones de AIR](#)” en la página 270.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

`true` o `false` (predeterminado)

Ejemplo

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 y posterior: Opcional

Indica que la aplicación requiere el uso del búfer de estencil o de profundidad. Normalmente estos búferes se utilizan al trabajar con contenido en 3D. De forma predeterminada, el valor de este elemento es `false` para desactivar los búferes de estencil y de profundidad. Este elemento es necesario, ya que los búferes deben estar asignados en el inicio de la aplicación antes de que se cargue ningún otro contenido.

El ajuste de este elemento debe coincidir con el valor transferido por el argumento `enableDepthAndStencil` al método `Context3D.configureBackBuffer()`. Si los valores no coinciden, AIR emite un error.

Este elemento solo se puede aplicar si `renderMode = direct`. Si `renderMode` no es igual a `direct`, ADT emite el error 118:

```
<depthAndStencil> element unexpected for render mode cpu. It requires "direct" render mode.
```

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

`true` o `false` (predeterminado)

Ejemplo

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

description

Adobe AIR 1.0 y posterior: Opcional

Descripción de la aplicación. Se muestra en el instalador de aplicaciones de AIR.

Si especifica un solo nodo de texto (en lugar de varios elementos para `text`), el instalador de aplicaciones de AIR utiliza este nombre, cualquiera que sea el idioma seleccionado para el sistema: De lo contrario, el instalador de aplicaciones de AIR utiliza la descripción que es más similar al idioma de la interfaz de usuario del sistema operativo del usuario. Por ejemplo, tomemos una instalación en la que el elemento `description` del archivo descriptor de la aplicación incluye un valor para la configuración regional “es” (española). El instalador de aplicaciones de AIR utiliza la descripción “es” si el sistema del usuario identifica “es” (español) como el idioma de la interfaz de usuario. También utiliza la descripción “es” si el idioma de la interfaz de usuario del sistema es es-ES (español de España). Sin embargo, si el idioma de la interfaz de usuario del sistema es es-ES y el archivo descriptor de la aplicación define tanto el nombre es-ES como el nombre es-UV, el instalador de aplicaciones de AIR utiliza el valor es-ES. Si la aplicación no define ninguna descripción que coincida con el idioma de la interfaz de usuario del sistema, el instalador de aplicaciones de AIR utiliza el primer valor de `description` que se define en el archivo descriptor de la aplicación.

Para obtener más información sobre el desarrollo de aplicaciones multilingües, consulte “[Localización de aplicaciones de AIR](#)” en la página 307.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: “[text](#)” en la página 251

Contenido

El esquema del descriptor de aplicaciones de AIR 1.0 solo permite definir un nodo de texto para el nombre (y no varios elementos de `text`).

En AIR 1.1 (o superior) se pueden especificar varios idiomas en el elemento `description`. El atributo `xml:lang` para cada elemento de texto especifica un código de idioma, de conformidad con lo definido en [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Ejemplo

Descripción el nodo de texto simple:

```
<description>This is a sample AIR application.</description>
```

Descripción con elementos de texto localizados para inglés, francés y español (válidos en AIR 1.1 y posterior):

```
<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 y posterior: Obligatorio

La descripción del tipo de archivo se muestra al usuario mediante el sistema operativo. La descripción del tipo de archivo no se localiza.

Consulte también: “[description](#)” en la página 227 como elemento secundario del elemento `application`

Elemento principal: “[fileType](#)” en la página 232

Elementos secundarios: ninguno

Contenido

Cadena que describe el contenido del archivo.

Ejemplo

```
<description>PNG image</description>
```

embedFonts

Permite utilizar fuentes personalizadas en StageText en la aplicación AIR. Este elemento es opcional.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: “[fuente](#)” en la página 233

Contenido

El elemento `embedFonts` puede contener cualquier elemento de fuente.

Ejemplo

```
<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 y posterior: solo iOS, opcional

iOS utiliza propiedades denominadas derechos (entitlements) para proporcionar acceso a la aplicación a recursos y funcionalidades adicionales. Utilice el elemento Entitlements para especificar esta información en una aplicación de iOS móvil.

Elemento principal: “[iPhone](#)” en la página 241

Elementos secundarios: elementos Entitlements.plist de iOS

Contenido

Contiene elementos secundarios que especifican pares de clave-valor para usarlos como parámetros de Entitlements.plist en la aplicación. El contenido del elemento Entitlements se debe incluir en un bloque CDATA. Para obtener más información, consulte la [Referencia de claves de derechos](#) en la biblioteca del desarrollador de iOS.

Ejemplo

```
<iphone>
...
  <Entitlements>
    <![CDATA[
      <key>aps-environment</key>
      <string>development</string>
    ]]>
  </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 y posterior: Obligatorio

La cadena de extensión de un tipo de archivo.

Elemento principal: “[fileType](#)” en la página 232

Elementos secundarios: ninguno

Contenido

Cadena que identifique los caracteres de la extensión de archivo (sin el punto, “.”).

Ejemplo

```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 y posterior

Especifica el ID de una extensión utilizada por la aplicación. El ID se define en el documento descriptor de la extensión.

Elemento principal: “[extensions](#)” en la página 230

Elementos secundarios: ninguno

Contenido

Cadena que identifica el ID de la extensión.

Ejemplo

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

extensions

Adobe AIR 2.5 y posterior: Opcional

Identifica las extensiones utilizadas por una aplicación.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: “[extensionID](#)” en la página 230

Contenido

Elementos `extensionID` que contienen los ID de extensión de ActionScript desde el archivo descriptor de la extensión.

Ejemplo

```
<extensions>
  <extensionID>extension.first</extensionID>
  <extensionID>extension.next</extensionID>
  <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 y posterior, solo iOS: Opcional

Especifica el nombre de un archivo de texto que contiene una lista de los archivos SWF que ADT debe configurar para alojamiento remoto. Puede minimizar el tamaño de descarga inicial de la aplicación mediante el empaquetamiento de un subconjunto de los archivos SWF usados por su aplicación y la carga del resto (solo activos) de archivos SWF externos en tiempo de ejecución con el método `Loader.load()`. Para utilizar esta función, se debe empaquetar la aplicación de tal manera que ADT mueva todo el código de bytes ActionScript (ABC) de los archivos SWF cargados de forma externa al SWF de la aplicación principal, dejando un archivo SWF que contenga únicamente activos. Esto se hace para cumplir con la regla de la Apple Store que prohíbe descargar código después de que una aplicación esté instalada.

Para obtener más información, consulte [“Reducir tamaño de descarga mediante la carga externa, los archivos SWF solo de recurso”](#) en la página 92.

Elemento principal: “[iPhone](#)” en la página 241, “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Nombre de un archivo de texto que contiene una lista delimitada por líneas de los archivos SWF que se deben alojar de forma remota.

Atributos:

Ninguno

Ejemplos

iOS:

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

filename

Adobe AIR 1.0 y posterior: Obligatorio

Cadena que se deberá utilizar como nombre de archivo de la aplicación (sin la extensión) al instalar esta. El archivo de la aplicación inicia la aplicación de AIR en el motor de ejecución. Si no se facilita ningún valor para `name`, el nombre de archivo (valor de `filename`) se utilizará también como nombre de la carpeta de instalación.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

La propiedad `filename` puede contener cualquier carácter Unicode (UTF-8) salvo los siguientes, cuyo uso en nombres de archivo está prohibido en diversos sistemas de archivos:

Carácter	Código hexadecimal
<i>diversos</i>	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
	x7C

El valor de `filename` no puede terminar con un punto.

Ejemplo

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 y posterior: Opcional

Describe un solo tipo de archivo que la aplicación puede registrar.

Elemento principal: “[fileTypes](#)” en la página 232

Elementos secundarios:

- “[contentType](#)” en la página 225
- “[description](#)” en la página 228
- “[extension](#)” en la página 229
- “[icon](#)” en la página 235
- “[name](#)” en la página 245

Contenido

Elementos que describen un tipo de archivo.

Ejemplo

```
<fileType>
  <name>foo.example</name>
  <extension>foo</extension>
  <description>Example file type</description>
  <contentType>text/plain</contentType>
  <icon>
    <image16x16>icons/fooiIcon16.png</image16x16>
    <image48x48>icons/fooiIcon48.png</image48x48>
  </icon>
</fileType>
```

fileTypes

Adobe AIR 1.0 y posterior: Opcional

El elemento `fileTypes` permite declarar los tipos de archivos con que se puede asociar una aplicación de AIR.

Cuando se instala una aplicación de AIR, los tipos de archivos declarados se registran en el sistema operativo y, si estos tipos de archivos aún no se encuentran asociados con ninguna otra aplicación, se asocian con la aplicación de AIR. Para suprimir una asociación existente entre un tipo de archivo y otra aplicación, utilice el método en tiempo de ejecución `NativeApplication.setAsDefaultApplication()` (preferentemente con el permiso del usuario).

Nota: los métodos del motor de ejecución solo pueden manejar asociaciones para los tipos de archivos declarados en el descriptor de la aplicación.

El elemento `fileTypes` es opcional.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: “[fileType](#)” en la página 232

Contenido

El elemento `fileTypes` puede contener cualquier número de elementos `fileType`.

Ejemplo

```
<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>
```

fuentes

Describe una única fuente personalizada que se puede utilizar en la aplicación AIR.

Elemento principal: “[embedFonts](#)” en la página 228

Elementos secundarios: “[fontName](#)” en la página 233, “[fontPath](#)” en la página 234

Contenido

Elementos que especifican el nombre de fuente personalizada y su ruta.

Ejemplo

```
<font>
  <fontPath>ttf/space age.ttf</fontPath>
  <fontName>space age</fontName>
</font>
```

fontName

Especifica el nombre de la fuente personalizada.

Elemento principal: “[fuente](#)” en la página 233

Elementos secundarios: ninguno

Contenido

El nombre de la fuente personalizada que se especificará en `StageText.fontFamily`

Ejemplo

```
<fontName>space age</fontName>
```

fontPath

Proporciona la ubicación del archivo de la fuente personalizada.

Elemento principal: “[fuente](#)” en la página 233

Elementos secundarios: ninguno

Contenido

Ruta del archivo de fuente personalizada (con respecto al origen).

Ejemplo

```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 y posterior, solo iOS: Opcional

Forzar el modo de representación de CPU para un conjunto de dispositivos específicos. Esta función permite activar de forma selectiva el modo de representación de GPU para los restantes dispositivos con iOS.

Esta etiqueta se agrega como un elemento secundario de la etiqueta `iPhone` y se especifica una lista de nombres de modelos de dispositivos separados por espacios. Los nombres de modelos de dispositivos válidos incluyen los siguientes:

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		

Elemento principal: “[iPhone](#)” en la página 241, “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Lista separada por espacios de nombres de modelos de dispositivos.

Atributos:

Ninguno

Ejemplos

iOS:

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

fullScreen

Adobe AIR 2.0 y posterior, iOS and Android: opcional

Especifica si la aplicación se inicia en modo de pantalla completa.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

true o false (predeterminado)

Ejemplo

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 y posterior: Opcional

Altura inicial de la ventana principal de la aplicación.

Si no se ve una altura, esta se determina mediante la configuración del archivo raíz SWF o, en el caso de una aplicación de AIR basada en HTML, mediante el sistema operativo.

Altura máxima de una ventana modificada de 2048 píxeles a 4096 píxeles en AIR 2.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Entero positivo con un valor máximo de 4095.

Ejemplo

```
<height>4095</height>
```

icon

Adobe AIR 1.0 y posterior: Opcional

La propiedad `icon` especifica un archivo de icono (o varios) a utilizar para la aplicación. Los iconos son opcionales. Si no se define la propiedad `icon`, el sistema operativo muestra un icono predeterminado.

La ruta se indica relativa al directorio raíz de la aplicación. Los archivos de iconos deben tener el formato PNG. Se pueden especificar todos los tamaños de icono siguientes:

Si hay un elemento para un tamaño determinado, la imagen que contiene el archivo debe ser exactamente del tamaño especificado. Si no se proporcionan todos los tamaños, el sistema operativo ajusta el tamaño más parecido para un uso determinado del icono.

***Nota:** los iconos especificados no se añaden automáticamente al paquete de AIR. Los archivos de iconos deben estar incluidos en sus lugares relativos correctos cuando se empaqueta de la aplicación.*

Para obtener el mejor resultado posible, proporcione una imagen para cada uno de los tamaños disponibles. Asegúrese también de que los iconos sean de buen aspecto tanto en el modo de color de 16 bits como en el de 32 bits.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: “[imageNxN](#)” en la página 237

Contenido

Elemento `imageNxN` para cada tamaño de icono deseado.

Ejemplo

```
<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 y posterior: Obligatorio

Cadena de identificación de la aplicación, denominada ID de la aplicación. En ocasiones se utiliza el Identificador de estilo DNS inverso, pero este estilo no es necesario.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

El valor de ID se limita a los siguientes caracteres:

- 0–9
- a–z
- A–Z
- . (punto)
- - (guion)

El valor debe contener entre 1 y 212 caracteres. Este elemento es obligatorio.

Ejemplo

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 y posterior: Opcional

Define la ruta a un icono relativa al directorio de la aplicación.

Se pueden utilizar las siguientes imágenes de icono; cada una de ellas especifica un tamaño de icono diferente:

- 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+)

El icono debe ser un gráfico PNG que sea del mismo tamaño indicado por el elemento de imagen. Los archivos de icono se deben incluir en el paquete de la aplicación; los iconos a los que se hace referencia en el documento descriptor de la aplicación no se incluyen automáticamente.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

La ruta del archivo al icono puede contener cualquier carácter Unicode (UTF-8) salvo los siguientes, cuyo uso en nombres de archivo está prohibido en diversos sistemas de archivos:

Carácter	Código hexadecimal
<i>diversos</i>	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E

Carácter	Código hexadecimal
?	x3F
\	x5C
	x7C

Ejemplo

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 y posterior: Opcional

Permite especificar las propiedades adicionales de una aplicación de iOS.

Elemento principal: “[iPhone](#)” en la página 241

Elementos secundarios: elementos Info.plist iOS

Contenido

Contiene elementos secundarios que especifican pares de clave-valor para usarlos como parámetros de Info.plist en la aplicación. El contenido del elemento InfoAdditions se debe incluir en un bloque CDATA.

Consulte [Information Property List Key Reference](#) (Referencia clave de la lista de propiedades de información; en inglés) en Apple iPhone Reference Library para obtener información sobre las parejas de valores clave y cómo expresarlas en XML.

Ejemplo

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

Más temas de ayuda

“[Configuración de iOS](#)” en la página 87

initialWindow

Adobe AIR 1.0 y posterior: Obligatorio

Define el archivo de contenido principal y el aspecto de la aplicación inicial.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: todos los elementos siguientes pueden aparecer como secundarios del elemento initialWindow. Sin embargo, algunos elementos se omiten dependiendo de si AIR admite las ventanas en una plataforma:

Elemento	Escritorio	Móvil
"aspectRatio" en la página 223	se omitirá	utilizado
"autoOrients" en la página 223	se omitirá	utilizado
"content" en la página 225	utilizado	utilizado
"depthAndStencil" en la página 227	utilizado	utilizado
"fullScreen" en la página 235	se omitirá	utilizado
"height" en la página 235	utilizado	se omitirá
"maximizable" en la página 243	utilizado	se omitirá
"maxSize" en la página 243	utilizado	se omitirá
"minimizable" en la página 244	utilizado	se omitirá
"minSize" en la página 244	utilizado	se omitirá
"renderMode" en la página 247	utilizado (AIR 3.0 y posterior)	utilizado
"requestedDisplayResolution" en la página 247	utilizado (AIR 3.6 y posterior)	se omitirá
"resizable" en la página 248	utilizado	se omitirá
"softKeyboardBehavior" en la página 249	se omitirá	utilizado
"systemChrome" en la página 251	utilizado	se omitirá
"title" en la página 252	utilizado	se omitirá
"transparent" en la página 252	utilizado	se omitirá
"visible" en la página 254	utilizado	se omitirá
"width" en la página 254	utilizado	se omitirá
"x" en la página 254	utilizado	se omitirá
"y" en la página 255	utilizado	se omitirá

Contenido

Elementos secundarios que definen el comportamiento y la apariencia de la aplicación.

Ejemplo

```
<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 y posterior: Opcional

Identifica el subdirectorio del directorio de instalación predeterminado.

En Windows el subdirectorio de instalación predeterminado es el directorio Archivos de programa. En Mac OS es el directorio /Aplicaciones. En Linux, es /opt/. Por ejemplo, si la propiedad `installFolder` está definida en "Acme" y una aplicación lleva el nombre "EjemploApl", la aplicación se instala en C:\Archivos de programa\Acme\ExampleApp en Windows, en /Aplicaciones/Acme/Example.app en MacOS y /opt/Acme/ExampleApp en Linux.

La propiedad `installFolder` es opcional. Si no se especifica ninguna propiedad para `installFolder`, la aplicación se instala en un subdirectorio del directorio de instalación predeterminado basado en la propiedad `name`.

Elemento principal: "application" en la página 219

Elementos secundarios: ninguno

Contenido

La propiedad `installFolder` puede contener cualquier carácter Unicode (UTF-8) excepto aquellos cuyo uso en nombres de archivo está prohibido en diversos sistemas de archivos (para ver la lista de excepciones, consulte la anterior propiedad `filename`).

Utilice la barra diagonal (/) como carácter separador entre directorios si desea especificar un subdirectorio anidado.

Ejemplo

```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, solo iOS: opcional

Define propiedades de la aplicación específica de iOS.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios:

- “[Entitlements](#)” en la página 229
- “[externalSwfs](#)” en la página 230
- “[forceCPURenderModeForDevices](#)” en la página 234
- “[InfoAdditions](#)” en la página 238
- “[requestedDisplayResolution](#)” en la página 247

Más temas de ayuda

“[Configuración de iOS](#)” en la página 87

manifest

Adobe AIR 2.5 y posterior, solo Android: Opcional

Especifica información para añadir al archivo de manifiesto de Android para la aplicación.

Elemento principal: “[manifestAdditions](#)” en la página 242

Elementos secundarios: Definidos por el SDK de Android.

Contenido

El elemento manifiesto no es, técnicamente hablando, parte del esquema del descriptor de la aplicación de AIR. Se trata de la raíz del documento XML de manifiesto de Android. Todo el contenido que se sitúe en el elemento manifest se debe ajustar al esquema de AndroidManifest.xml. Cuando se genera un archivo APK con las herramientas de AIR, la información del elemento manifest se copia en la parte correspondiente del archivo AndroidManifest.xml generado de la aplicación.

Si especifica valores de manifest en Android que solamente están disponibles en una versión del SDK más reciente que la admitida directamente en AIR, deberá establecer el indicador `-platformsdk` como ADT al empaquetar la aplicación. Establezca el indicador en la ruta del sistema de archivos en una versión del SDK de Android que admita los valores que está añadiendo.

El propio elemento manifest se debe incluir en un bloque CDATA en el descriptor de la aplicación de AIR.

Ejemplo

```
<![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>  
]]>
```

Más temas de ayuda

“[Configuración de Android](#)” en la página 81

[Archivo AndroidManifest.xml](#)

manifestAdditions

Adobe AIR 2.5 y posterior, solo Android

Especifica la información para añadir al archivo de manifiesto de Android.

Todas las aplicaciones de Android incluyen un archivo de manifiesto que define las propiedades básicas de la aplicación. El archivo de manifiesto de Android es similar en concepto al descriptor de la aplicación de AIR. Una aplicación de AIR para Android cuenta con un descriptor de la aplicación y archivo de manifiesto de Android generado automáticamente. Cuando se empaqueta una aplicación de AIR para Android, la información en este elemento `manifestAdditions` se añade a las partes correspondientes del documento de manifiesto de Android.

Elemento principal: “[android](#)” en la página 219

Elementos secundarios: “[manifest](#)” en la página 241

Contenido

La información del elemento `manifestAdditions` se añade al documento XML `AndroidManifest`.

AIR establece varias entrada de manifiesto en el documento de manifiesto de Android generado para garantizar que las funciones del motor de ejecución y la aplicación funcionan correctamente. La siguiente configuración no puede omitirse:

Los siguientes atributos del elemento de manifiesto no se pueden establecer:

- `package`
- `android:versionCode`
- `android:versionName`

Los siguientes atributos del elemento de actividad principal no se pueden establecer:

- `android:label`
- `android:icon`

Los siguientes atributos del elemento de la aplicación no se pueden establecer:

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

Ejemplo

```
<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>
```

Más temas de ayuda

“Configuración de Android” en la página 81

[Archivo AndroidManifest.xml](#)

maximizable

Adobe AIR 1.0 y posterior: Opcional

Especifica si la ventana se puede maximizar.

Nota: en los sistemas operativos como Mac OS X en los cuales la maximización de las ventanas es una operación de redimensionamiento, para que la ventana no cambie de tamaño, tanto "maximizable" como "resizable" deben definirse en *false*.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

true (predeterminado) o false

Ejemplo

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 y posterior: Opcional

Tamaños máximos de la ventana. Si ni se establece un tamaño máximo, este se determina mediante el sistema operativo.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Dos enteros que representan la altura y la anchura máximas, separados por espacios en blanco.

Nota: el tamaño de ventana máximo admitido en AIR ha aumentado de 2048x2048 píxeles a 4096x4096 píxeles en AIR 2. (Debido a que las coordenadas de la pantalla se basan en cero, el valor máximo se puede utilizar para la anchura y la altura es de 4095.)

Ejemplo

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 y posterior: Opcional

Especifica si la ventana se puede minimizar.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

true (predeterminado) o false

Ejemplo

```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 y posterior: Opcional

Especifica el tamaño mínimo permitido para la ventana.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Dos enteros que representan la altura y la anchura mínimas, separados por espacios en blanco. Se debe tener en cuenta que el tamaño mínimo impuesto por el sistema operativo tiene precedencia sobre el valor establecido en el descriptor de la aplicación.

Ejemplo

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 y posterior: Opcional

El título de la aplicación mostrado por el instalador de la aplicación de AIR.

Si no se especifica ningún elemento `name`, el instalador de aplicaciones de AIR muestra el nombre de archivo definido para `filename` como el nombre de la aplicación.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: “[text](#)” en la página 251

Contenido

Si especifica un solo nodo de texto (en lugar de varios elementos `<text>`), el instalador de aplicaciones de AIR utiliza este nombre, cualquiera que sea el idioma seleccionado para el sistema.

El esquema del descriptor de aplicaciones de AIR 1.0 solo permite definir un nodo de texto para el nombre (y no varios elementos de `text`). En AIR 1.1 (o posterior) se pueden especificar varios idiomas en el elemento `name`.

El atributo `xml:lang` para cada elemento de texto especifica un código de idioma, de conformidad con lo definido en [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

El instalador de aplicaciones de AIR utiliza el nombre que mejor se corresponde con el idioma de la interfaz de usuario del sistema operativo del usuario. Por ejemplo, tomemos una instalación en la que el elemento `name` del archivo descriptor de la aplicación incluye un valor para la configuración regional "es" (español). El instalador de aplicaciones de AIR utiliza el nombre "es" si el sistema operativo identifica "es" (español) como el idioma de la interfaz de usuario. También utiliza el nombre "es" si el idioma de la interfaz de usuario del sistema es es-ES (español de España). Sin embargo, si el idioma de la interfaz de usuario es es-ES y el archivo descriptor de la aplicación define tanto el nombre es-ES como el nombre es-UV, el instalador de aplicaciones de AIR utiliza el valor es-ES. Si la aplicación no define ningún nombre que coincida con el idioma de la interfaz de usuario del sistema, el instalador de aplicaciones de AIR utiliza el primer valor de `name` que se define en el archivo descriptor de la aplicación.

El elemento `name` solo define el título de la aplicación que se utiliza en el instalador de aplicaciones de AIR. El instalador de aplicaciones de AIR admite varios idiomas: chino tradicional, chino simplificado, checo, neerlandés, inglés, francés, alemán, italiano, japonés, coreano, polaco, portugués brasileño, ruso, español, sueco y turco. El instalador de aplicaciones de AIR selecciona el idioma visualizado (para texto que no sea el título y la descripción de la aplicación) con base en el idioma de la interfaz de usuario del sistema. Esta selección de idioma es independiente de la configuración del archivo descriptor de la aplicación.

El elemento `name` define las configuraciones regionales disponibles para la aplicación instalada y en funcionamiento. Para obtener más información sobre el desarrollo de aplicaciones multilingües, consulte [“Localización de aplicaciones de AIR”](#) en la página 307.

Ejemplo

El siguiente ejemplo define un nombre con un nodo de texto simple:

```
<name>Test Application</name>
```

El siguiente ejemplo, válido para AIR 1.1 y posterior, especifica el nombre en tres idiomas (inglés, francés y español) utilizando los nodos del elemento `<text>`:

```
<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 y posterior: Obligatorio

Identifica el nombre de un tipo de archivo.

Elemento principal: “fileType” en la página 232

Elementos secundarios: ninguno

Contenido

Cadena que representa el nombre del tipo de archivo.

Ejemplo

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 y posterior: Opcional

Identifica dónde deben guardarse los accesos directos a la aplicación en el menú Todos los programas del sistema operativo Windows o en el menú de aplicaciones de Linux. (En otros sistemas operativos, en la actualidad se hace caso omiso a esta opción).

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

La cadena utilizada para el valor `programMenuFolder` puede contener cualquier carácter Unicode (UTF-8) excepto aquellos cuyo uso en nombres de archivo está prohibido en diversos sistemas de archivos (para ver la lista de excepciones, consulte el elemento `filename`). No utilice una barra diagonal (/) como último carácter de este valor.

Ejemplo

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 y posterior: Opcional

Identifica el ID del editor para actualizar una aplicación de AIR creada en un principio con una versión de AIR 1.5.2 o posterior.

Especifique un ID de editor únicamente al crear una actualización de la aplicación. El valor de elemento `publisherID` debe coincidir con el ID del editor generado por AIR en la versión anterior de la aplicación. Para una aplicación instalada, el ID del editor se puede encontrar en la carpeta en la que se instala en una aplicación, en el archivo `META-INF/AIR/publisherid`.

Las nuevas aplicaciones creadas con AIR 1.5.3 o posterior no deben especificar ningún ID de editor.

Para obtener más información, consulte “[Identificador del editor de AIR](#)” en la página 200.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Cadena del ID de editor.

Ejemplo

```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 y posterior: Opcional

Especifica si utilizar la aceleración con la unidad de procesamiento de gráficos (GPU), si se admite en el dispositivo informático actual.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Uno de los valores siguientes:

- `auto` (predeterminado): actualmente se sitúa en modo CPU.
- `cpu`: la aceleración de hardware no se utiliza.
- `direct`: la composición del procesamiento se produce en la CPU; la fusión utiliza la GPU. Disponible en AIR 3+.

***Nota:** para poder aprovechar la aceleración de GPU del contenido de Flash con plataformas de AIR para móviles, Adobe recomienda utilizar `renderMode="direct"` (es decir, Stage3D) en vez de `renderMode="gpu"`. Adobe oficialmente admite y recomienda las siguientes arquitecturas basadas en Stage3D: Starling (2D) y Away3D (3D). Para obtener más información sobre Stage3D y Starling/Away3D, consulte <http://gaming.adobe.com/getstarted/>.*

- `gpu`: la aceleración de hardware se usa, si está disponible.

***Importante:** no utilice el modo de procesamiento de GPU para las aplicaciones de Flex.*

Ejemplo

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

requestedDisplayResolution

Adobe AIR 2.6 y versiones posteriores, solo iOS; Adobe AIR 3.6 y versiones posteriores, OS X — Opcional

Especifica si la aplicación desea utilizar la resolución estándar o alta en un dispositivo o un monitor de ordenador con una pantalla de alta resolución. Cuando se establece en *estándar* (valor predeterminado), la pantalla aparecerá en la aplicación como pantalla de resolución estándar. Cuando se establece en *alta*, la aplicación puede gestionar cada píxel de alta resolución.

Por ejemplo, en una pantalla de iPhone con alta resolución 640x960, si el valor es *estándar*, las dimensiones de escenario a pantalla completa serán 320x480 y cada píxel de aplicación se representa mediante cuatro píxeles de pantalla. Si el valor es *alta*, las dimensiones de escenario a pantalla completa serán 640x960.

En los dispositivos con pantallas de resolución estándar, las dimensiones del escenario coinciden con las dimensiones de la pantalla independientemente de la configuración empleada.

Si el elemento `requestedDisplayResolution` está anidado en el elemento `iPhone`, se aplica a dispositivos iOS. En ese caso, el atributo `excludeDevices` se puede usar para especificar dispositivos a los que no se aplica el valor.

Si el elemento `requestedDisplayResolution` está anidado en el elemento `initialWindow`, se aplica a aplicaciones de AIR de escritorio en ordenadores MacBook Pro que admitan presentaciones de alta resolución. El valor especificado se aplica a todas las ventanas nativas usadas en la aplicación. El anidado del elemento `requestedDisplayResolution` en el elemento `initialWindow` es posible a partir de AIR 3.6.

Elemento principal: “[iPhone](#)” en la página 241, “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

standard (valor predeterminado) o *high*.

Atributo:

`excludeDevices`: lista de prefijos de nombres de modelo o nombres de modelo de iOS, separados por espacios. Permite al desarrollador especificar que ciertos dispositivos usen alta resolución y otros resolución estándar. Este atributo solo está disponible en iOS (el elemento `requestedDisplayResolution` está anidado en el elemento `iPhone`). El atributo `excludeDevices` está disponible a partir de AIR 3.6.

Para dispositivos cuyo nombre de modelo se especifique en este atributo, el valor de `requestedDisplayResolution` es el opuesto al especificado. Es decir, si el valor de `requestedDisplayResolution` es *alta*, los dispositivos excluidos usan resolución estándar. Si el valor de `requestedDisplayResolution` es *estándar*, los dispositivos excluidos usan resolución alta.

Los valores son prefijos de nombres de modelo o nombres de modelo de dispositivos iOS. Por ejemplo, el valor `iPad3,1` hace referencia específica a un iPad con Wi-Fi de tercera generación (pero no a iPads GSM o CDMA de tercera generación). El valor `iPad3` también se puede referir a cualquier iPad de tercera generación. Encontrará una lista no oficial de nombres de modelo de iOS en [la página de modelos de iPhone wiki](#).

Ejemplos

Escritorio:

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

iOS:

```
<iPhone>
  <requestedDisplayResolution excludeDevices="iPad3
iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 y posterior: Opcional

Especifica si la ventana se puede cambiar de tamaño.

Nota: en los sistemas operativos como Mac OS X en los cuales la maximización de las ventanas es una operación de redimensionamiento, para que la ventana no cambie de tamaño, tanto "maximizable" como "resizable" deben definirse en *false*.

Elemento principal: "`initialWindow`" en la página 238

Elementos secundarios:

Contenido

true (predeterminado) o *false*

Ejemplo

```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 y posterior, perfil móvil: Opcional

Especifica el comportamiento predeterminado de la aplicación cuando se muestra un teclado virtual. El comportamiento predeterminado es desplazar la aplicación hacia arriba. El motor de ejecución mantiene el campo de texto u objeto interactivo seleccionados en pantalla. Utilice la opción *pan* si la aplicación no proporciona su propia lógica de administración de teclado.

También puede desactivar el comportamiento automático estableciendo el elemento `softKeyboardBehavior` en *none*. En este caso, los campos de texto y los objetos interactivos distribuyen un evento `SoftKeyboardEvent` cuando el teclado de software se activa, pero el motor de ejecución no desplaza ni cambia el tamaño de la aplicación. Es responsabilidad de la aplicación mantener el área de entrada de texto visible.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

none o *pan*. El valor predeterminado es *pan*.

Ejemplo

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Más temas de ayuda

[SoftKeyboardEvent](#)

supportedLanguages

Adobe AIR 3.2 y posterior: Opcional

Identifica los idiomas admitidos en la aplicación. Este elemento solamente se utiliza en iOS, el motor de ejecución captador de Mac y en aplicaciones de Android. Este elemento se omite en todos los demás tipos de aplicaciones.

Si no especifica este elemento, el empaquetador lleva a cabo de forma predeterminada las siguientes acciones en función del tipo de aplicación:

- iOS: todos los idiomas admitidos por el motor de ejecución de AIR se enumeran en el App Store de iOS como idiomas admitidos de la aplicación.
- Motor de ejecución captador de Mac: la aplicación empaquetada con el paquete captador no contiene información de localización.
- Android: el paquete de la aplicación tiene los recursos para todos los idiomas admitidos por el motor de ejecución de AIR.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Una lista de los idiomas admitidos, delimitada con espacios. Los valores de idioma válidos son los valores ISO 639-1 de los idiomas admitidos por el motor de ejecución de AIR: en, de, es, fr, it, ja, ko, pt, ru, cs, nl, pl, sv, tr, zh, da, nb, iw.

El empaquetador genera un error en caso de encontrar un valor vacío en el elemento `<supportedLanguages>`.

Nota: las etiquetas localizadas (por ejemplo, el nombre de etiqueta) omiten el valor del idioma i se utiliza la etiqueta `<supportedLanguages>` y no contiene el idioma. Si una extensión nativa tiene recursos para un idioma no especificado por la etiqueta `<supportedLanguages>`, se emite una advertencia y los recursos se omiten para dicho idioma.

Ejemplo

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 y posterior: Opcional

Identifica los perfiles compatibles con la aplicación.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Es posible incluir cualquiera de estos valores en el elemento `supportedProfiles`:

- `desktop`: el perfil de escritorio define las aplicaciones de AIR que están instaladas en un equipo de escritorio utilizando un archivo de AIR. Estas aplicaciones no tienen acceso a la clase `NativeProcess` (que proporciona comunicación con las aplicaciones nativas).
- `extendedDesktop`: el perfil de escritorio ampliado define las aplicaciones de AIR que se encuentran instaladas en un equipo de escritorio utilizando un instalador de aplicaciones nativas. Estas aplicaciones tienen acceso a la clase `NativeProcess` (que proporciona comunicación con las aplicaciones nativas).
- `mobileDevice`: perfil del dispositivo móvil para las aplicaciones móviles.
- `extendedMobileDevice`: el perfil de dispositivo móvil ampliado no está en uso actualmente.

La propiedad `supportedProfiles` es opcional. Si no se incluye este elemento en el archivo descriptor de la aplicación, esta se puede compilar e implementar para cualquier perfil.

Para especificar varios perfiles, separe cada uno de ellos con carácter de espacio. Por ejemplo, la siguiente configuración especifica que la aplicación solo está disponible en los perfiles de escritorio y ampliados:

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

Nota: cuando se ejecuta una aplicación con ADL y no se especifica ningún valor para la opción `-profile` de ADL, se utiliza el primer perfil en el descriptor de la aplicación. (Si tampoco se especifican perfiles es el descriptor de la aplicación, se utilizará el perfil de escritorio.)

Ejemplo

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

Más temas de ayuda

“[Perfiles de dispositivo](#)” en la página 256

“[Perfiles admitidos](#)” en la página 80

systemChrome

Adobe AIR 1.0 y posterior: Opcional

Especifica si la ventana de la aplicación inicial se crea con la barra de título estándar, los bordes y los controles que proporciona el sistema operativo.

El fondo cromático de una ventana no se puede cambiar en tiempo de ejecución.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Uno de los valores siguientes:

- `none`: no se proporciona ningún fondo cromático. La aplicación (o un marco de la aplicación como Flex) es responsable de mostrar el fondo cromático de la ventana.
- `standard` (valor predeterminado): el fondo cromático del sistema lo proporciona el sistema operativo.

Ejemplo

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 y posterior: Opcional

Especifica una cadena localizada.

El atributo `xml:lang` para cada elemento de texto especifica un código de idioma, de conformidad con lo definido en [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

El instalador de la aplicación de AIR utiliza el elemento `text` con el valor de atributo `xml:lang` que más coincide con el idioma de la interfaz de usuario del sistema operativo del usuario.

Por ejemplo, considere una instalación en la que un elemento `text` incluye un valor para la configuración local de inglés en (inglés). El instalador de aplicaciones de AIR utiliza el nombre "es" si el sistema operativo identifica "es" (español) como el idioma de la interfaz de usuario. También utiliza el nombre "es" si el idioma de la interfaz de usuario del sistema es es-ES (español de España). Sin embargo, si el idioma de la interfaz de usuario es es-ES y el archivo descriptor de la aplicación define tanto el nombre es-ES como el nombre es-UV, el instalador de aplicaciones de AIR utiliza el valor es-ES.

Si la aplicación no define ningún elemento `text` que coincide con los idiomas de la interfaz de usuario del sistema, el instalador de aplicaciones de AIR utiliza el primer valor `name` definido en el archivo descriptor de la aplicación.

Elementos principales:

- “[name](#)” en la página 244
- “[description](#)” en la página 227

Elementos secundarios: ninguno

Contenido

Un atributo `xml:lang` que especifica una configuración regional y una cadena de texto localizado.

Ejemplo

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 y posterior: Opcional

Especifica el título mostrado en la barra de título de la ventana de la aplicación.

Un título solo se muestra si el elemento `systemChrome` se define como `standard`.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Cadena que contiene el título de la ventana.

Ejemplo

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 y posterior: Opcional

Especifica si la ventana de la aplicación inicial es de mezcla alfa con el escritorio.

Una ventana con transparencia activada puede dibujarse más lentamente y necesitar más memoria. La opción de transparencia no puede modificarse durante el tiempo de ejecución.

Importante: solo se puede definir `transparent` en `true` si `systemChrome` tiene el valor.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

`true` o `false` (predeterminado)

Ejemplo

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

version

Adobe AIR 1.0 a 2.0: obligatorio; No se permite en AIR 2.5 y posterior

Especifica la información sobre la versión para la aplicación.

La cadena de la versión es un designador definido por la aplicación. AIR no interpreta la cadena de versión de ningún modo. Por ejemplo, no supone que la versión “3.0” es más actualizada que la versión “2.0”. Ejemplos: “1.0”, “.4”, “0.5”, “4.9”, “1.3.4a”.

En AIR 2.5 y posterior, el elemento `version` se sustituye por los elementos `versionNumber` y `versionLabel`.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Cadena que contiene la versión de la aplicación.

Ejemplo

```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 y posterior: Opcional

Especifica una cadena de versión legible para el usuario.

El valor de la etiqueta de la versión se muestra en los cuadros de diálogo de instalación en lugar del valor del elemento `versionNumber`. Si no se utiliza `versionLabel`, `versionNumber` se usa para ambos.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

Cadena que contiene el texto de la versión mostrada públicamente.

Ejemplo

```
<versionLabel>0.9 Beta</versionLabel>
```

versionNumber

Adobe AIR 2.5 y posterior: Obligatorio

Número de versión de la aplicación.

Elemento principal: “[application](#)” en la página 219

Elementos secundarios: ninguno

Contenido

El número de versión puede contener una secuencia de hasta tres enteros separados por puntos. Cada entero debe ser un número entre 0 y 999 (incluido).

Ejemplos

```
<versionNumber>1.0.657</versionNumber>
```

```
<versionNumber>10</versionNumber>
```

```
<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 y posterior: Opcional

Especifica si la ventana de la aplicación inicial se puede ver tras su creación.

Las ventanas de AIR, incluyendo la ventana inicial, se crean en un estado visible de forma predeterminada. Se podrá mostrar una ventana llamando al método `activate()` del objeto `NativeWindow` o cambiando la propiedad `visible` a `true`. Puede convenir dejar la ventana principal oculta al principio para que no se muestren los ajustes de la posición y el tamaño de la ventana y la disposición del contenido.

El componente `mx:WindowedApplication` de Flex muestra y activa la ventana de forma automática inmediatamente antes de distribuir el evento `applicationComplete`, a menos que el atributo `visible` esté definido en `false` en la definición MXML.

En los dispositivos en el perfil móvil, que no admite ventanas, la configuración visible se omite.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

`true` o `false` (predeterminado)

Ejemplo

```
<visible>true</visible>
```

width

Adobe AIR 1.0 y posterior: Opcional

Anchura inicial de la ventana principal de la aplicación.

Si no se establece una anchura, esta se determina mediante los ajustes en el archivo raíz SWF, o bien, en el caso de una aplicación de AIR basada en HTML, mediante el sistema operativo.

Anchura máxima de una ventana modificada de 2048 píxeles a 4096 píxeles en AIR 2.

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Entero positivo con un valor máximo de 4095.

Ejemplo

```
<width>1024</width>
```

X

Adobe AIR 1.0 y posterior: Opcional

Posición horizontal de la ventana de la aplicación inicial.

En la mayoría de los casos, es mejor permitir que el sistema operativo determine la posición inicial de la ventana en lugar de asignar un valor fijo.

El origen del sistema de coordenadas de pantalla (0,0) es la esquina superior izquierda de la pantalla del escritorio principal (tal y como se determina mediante el sistema operativo).

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Valor entero.

Ejemplo

```
<x>120</x>
```

y

Adobe AIR 1.0 y posterior: Opcional

Posición vertical de la ventana de la aplicación inicial.

En la mayoría de los casos, es mejor permitir que el sistema operativo determine la posición inicial de la ventana en lugar de asignar un valor fijo.

El origen del sistema de coordenadas de pantalla (0,0) es la esquina superior izquierda de la pantalla del escritorio principal (tal y como se determina mediante el sistema operativo).

Elemento principal: “[initialWindow](#)” en la página 238

Elementos secundarios: ninguno

Contenido

Valor entero.

Ejemplo

```
<y>250</y>
```

Capítulo 15: Perfiles de dispositivo

Adobe AIR 2 y posterior

Los perfiles son un mecanismo para definir las clases de dispositivos informáticos en los que funciona la aplicación. Un perfil define un conjunto de APIs y capacidades que suelen admitirse en una clase concreta de dispositivo. Entre los perfiles disponibles se incluyen:

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Se pueden definir los perfiles para la aplicación en el descriptor de la aplicación. Los usuarios de los equipos y dispositivos en los perfiles incluidos pueden instalar la aplicación; los usuarios de otros equipos y dispositivos no. Por ejemplo, si solo se incluye el perfil de escritorio en el descriptor de la aplicación, los usuarios pueden instalar y ejecutar la aplicación únicamente en los equipos de escritorio.

Si se incluye un perfil que la aplicación no admita realmente, la experiencia del usuario en estos entornos puede no ser adecuada. Si no se especifica ningún perfil en el descriptor de la aplicación, AIR no limita la aplicación. La aplicación se puede empaquetar en cualquiera de los formatos admitidos y los usuarios con dispositivos de cualquier perfil pueden instalarla. Sin embargo, puede que no funcione adecuadamente en tiempo de ejecución.

Si es posible, las restricciones de perfil se aplican cuando se empaqueta la aplicación. Por ejemplo, si solo se incluye el perfil `extendedDesktop`, no se podrá empaquetar la aplicación como archivo de AIR; solo como instalador nativo. Del mismo modo, si nos e incluye el perfil `mobileDevice`, la aplicación no se puede empaquetar como APK de Android.

Un solo dispositivo informático puede admitir varios perfiles. Por ejemplo, AIR en los equipos de escritorio admite aplicaciones para los perfiles `extendedDesktop` y de escritorio. Sin embargo, una aplicación de perfil de escritorio ampliada puede comunicarse con procesos nativos y DEBE empaquetarse como instalador nativo (`exe`, `dmg`, `deb` o `rpm`). Por otra parte, una aplicación de perfil de escritorio no se puede comunicar con un proceso nativo. Una aplicación de perfil de escritorio se puede empaquetar como archivo de AIR o instalador nativo.

La inclusión de una función en un perfil indica que la compatibilidad con dicha función es común en la clase de dispositivos para los que se define ese perfil. Sin embargo, no significa que todos los dispositivos de un perfil admitan todas las funciones. Por ejemplo la mayoría de los teléfonos móviles, pero no todos, incluyen un acelerómetro. Las clases y las funciones que presentan compatibilidad universal suelen tener una propiedad booleana que se puede comprobar antes de utilizar la función. En el caso del acelerómetro, por ejemplo, se puede probar la propiedad estática `Accelerometer.isSupported` con el fin de determinar si el dispositivo actual cuenta con un acelerómetro compatible.

Los siguientes perfiles se pueden asignar a la aplicación de AIR utilizando el elemento `supportedProfiles` en el descriptor de la aplicación:

Escritorio El perfil de escritorio define un conjunto de capacidades para aplicaciones de AIR que se instalan como archivos de AIR en un equipo de escritorio. Estas aplicaciones se instalarán y ejecutarán en plataformas de escritorio compatibles (Mac OS, Windows y Linux). Las aplicaciones de AIR desarrolladas en versiones de AIR anteriores a AIR 2, se pueden considerar dentro del perfil de escritorio. Algunas APIs no funcionan en este perfil. Por ejemplo, las aplicaciones de escritorio no se pueden comunicar con procesos nativos.

Escritorio ampliado El perfil de escritorio ampliado define un conjunto de capacidades para aplicaciones de AIR que están empaquetadas e instaladas con un instalador nativo. Estos instaladores nativos son archivos EXE en Windows, archivos DMG en Mac OS y archivos BIN, DEB o RPM en Linux. Las aplicaciones de escritorio ampliadas cuentan con capacidades adicionales que no están disponibles en las aplicaciones de perfil de escritorio. Para obtener más información, consulte “[Empaquetado de un instalador nativo de escritorio](#)” en la página 60.

Dispositivo móvil El perfil de dispositivo móvil define un conjunto de capacidades para las aplicaciones que se encuentran instaladas en dispositivos móviles como, por ejemplo, teléfonos móviles y tablets. Estas aplicaciones se instalan y se ejecutan en plataformas móviles admitidas, entre las que se incluyen Android, Blackberry Tablet OS e iOS.

Dispositivo móvil ampliado El perfil de dispositivo móvil extendido define un conjunto ampliado de capacidades para las aplicaciones están instaladas en dispositivos móviles. Actualmente, no existen dispositivos que admitan este perfil.

Restricción de perfiles de destino en el archivo descriptor de la aplicación

Adobe AIR 2 y posterior

Desde AIR 2, el archivo descriptor de la aplicación incluye un elemento `supportedProfiles`, que permite restringir perfiles de destino. Por ejemplo, la siguiente configuración especifica que la aplicación solo está disponible en los perfiles de escritorio:

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

Cuando se establece este elemento, la aplicación solo se puede empaquetar en los perfiles especificados. Utilice los siguientes valores:

- `desktop`: perfil de escritorio.
- `extendedDesktop`: perfil de escritorio ampliado.
- `mobileDevice`: perfil de dispositivo móvil.

El elemento `supportedProfiles` es opcional. Si no se incluye este elemento en el archivo descriptor de la aplicación, esta se puede empaquetar e implementar para cualquier perfil.

Para especificar varios perfiles en el elemento `supportedProfiles`, separe cada uno de ellos con un carácter de espacio, tal y como se indica a continuación:

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

Capacidades en diferentes perfiles

Adobe AIR 2 y posterior

En la siguiente tabla se incluyen las clases y las funciones que no se admiten en todos los perfiles.

Clase o Función	desktop	extendedDesktop	mobileDevice
Acelerómetro (Accelerometer.isSupported)	No	No	Activar
Accesibilidad (Capabilities.hasAccessibility)	Sí	Sí	No
Cancelación de eco acústico (Microphone.getEnhancedMicrophone())	Sí	Sí	No
ActionScript 2	Sí	Sí	No
CacheAsBitmap matrix	No	No	Sí
Camera (Camera.isSupported)	Sí	Sí	Sí
CameraRoll	No	No	Sí
CameraUI (CameraUI.isSupported)	No	No	Sí
Paquetes de motor de ejecución captadores	Sí	Sí	Sí
ContextMenu (ContextMenu.isSupported)	Sí	Sí	No
DatagramSocket (DatagramSocket.isSupported)	Sí	Sí	Sí
DockIcon (NativeApplication.supportsDockIcon)	Activar	Comprobar	No
Drag-and-drop (NativeDragManager.isSupported)	Sí	Sí	Comprobar
EncryptedLocalStore (EncryptedLocalStore.isSupported)	Sí	Sí	Sí
Flash Access (DRMManager.isSupported)	Sí	Sí	No
GameInput (GameInput.isSupported)	No	No	No
Geolocation (Geolocation.isSupported)	No	No	Activar
HTMLLoader (HTMLLoader.isSupported)	Sí	Sí	No
IME (IME.isSupported)	Sí	Sí	Comprobar
LocalConnection (LocalConnection.isSupported)	Sí	Sí	No
Microphone (Microphone.isSupported)	Sí	Sí	Comprobar
Audio multicanal (Capabilities.hasMultiChannelAudio())	No	No	No
Extensiones nativas	No	Sí	Sí
NativeMenu (NativeMenu.isSupported)	Sí	Sí	No
NativeProcess (NativeProcess.isSupported)	No	Sí	No
NativeWindow (NativeWindow.isSupported)	Sí	Sí	No
NetworkInfo (NetworkInfo.isSupported)	Sí	Sí	Comprobar

Clase o Función	desktop	extendedDesktop	mobileDevice
Apertura de archivos con la aplicación predeterminada	Limitado	Sí	No
PrintJob (PrintJob.isSupported)	Sí	Sí	No
SecureSocket (SecureSocket.isSupported)	Sí	Sí	Comprobar
ServerSocket (ServerSocket.isSupported)	Sí	Sí	Sí
Sombreado	Sí	Sí	Limitado
Stage3D (Stage.stage3Ds.length)	Sí	Sí	Sí
Stage orientation (Stage.supportsOrientationChange)	No	No	Sí
StageVideo	No	No	Activar
StageWebView (StageWebView.isSupported)	Sí	Sí	Sí
Iniciar la aplicación en el inicio de sesión (NativeApplication.supportsStartAtLogin)	Sí	Sí	No
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Sí	Sí	No
Modo de inactividad del sistema	No	No	Sí
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Activar	Comprobar	No
Entrada de Text Layout Framework	Sí	Sí	No
Updater (Updater.isSupported)	Sí	No	No
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Sí	Sí	No

Las entradas de la tabla tienen los siguientes significados:

- *Check*: la función no se admite en algunos, pero no todos, dispositivos en el perfil. Se debe comprobar en tiempo de ejecución si la función se admite antes de utilizarla.
- *Limited*: la función se admite, pero tiene limitaciones importantes. Consulte la documentación correspondiente para obtener más información.
- *No*: la función no se admite en el perfil.
- *Yes*: la función se admite en el perfil. Se debe tener en cuenta que los dispositivos informáticos independientes pueden carecer del hardware necesario para una función. Por ejemplo, no todos los teléfonos tienen cámaras.

Especificación de perfil al depurar con ADL

Adobe AIR 2 y posterior

ADL comprueba si se especifican los perfiles admitidos en el elemento `supportedProfiles` del archivo descriptor de la aplicación. Si es así, ADL utilizará de forma predeterminada el primer perfil admitido incluido como perfil en el proceso de depuración.

Es posible especificar un perfil para la sesión de depuración de ADL utilizando el argumento de la línea de comandos `-profile`. (Consulte “[AIR Debug Launcher \(ADL\)](#)” en la página 168.) Este argumento se puede utilizar tanto si se especifica como si no un perfil en el elemento `supportedProfiles` en el archivo descriptor de la aplicación. No obstante, si se especifica un elemento `supportedProfiles`, es necesario incluir el perfil en la línea de comandos. De lo contrario, ADL genera un error

Capítulo 16: API en navegador AIR.SWF

Personalización del archivo badge.swf de instalación integrada

Además de utilizar el archivo badge.swf suministrado con el SDK, puede crear su propio archivo SWF para usarlo en una página del navegador. El archivo SWF personalizado puede interactuar con el motor de ejecución de cualquiera de las formas siguientes:

- Puede instalar una aplicación de AIR. Consulte [“Instalación de una aplicación de AIR desde el navegador”](#) en la página 267.
- Puede comprobar si hay instalada una aplicación de AIR en particular. Consulte [“Cómo comprobar desde una página web si una aplicación de AIR está instalada”](#) en la página 266.
- Puede comprobar si está instalado el motor de ejecución. Consulte [“Cómo comprobar si está instalado el motor de ejecución”](#) en la página 265.
- Puede iniciar una aplicación de AIR instalada en el sistema del usuario. Consulte [“Inicio desde el navegador de una aplicación de AIR instalada”](#) en la página 268.

Estas capacidades se proporcionan al llamar a las API de un archivo SWF alojado en adobe.com: air.swf. Puede personalizar el archivo badge.swf y llamar a las API air.swf desde el propio archivo SWF.

Además, un archivo SWF que se ejecuta en el navegador puede comunicarse con una aplicación de AIR en curso utilizando la clase LocalConnection. Para obtener más información, consulte [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de ActionScript) o [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de HTML).

Importante: las funciones que se describen en esta sección (y las API del archivo air.swf) requieren que el usuario tenga Adobe® Flash® Player 9 actualización 3 instalado en Windows o Mac OS. En Linux, la función de instalación integrada requiere Flash Player 10 (versión 10,0,12,36 o posterior). Se puede escribir código para comprobar la versión instalada de Flash Player y proveer una interfaz alternativa para el usuario si no está instalada la versión de Flash Player que se requiere. Por ejemplo: si hay una versión anterior de Flash Player instalada, podría proporcionar un vínculo a la versión de descarga del archivo de AIR (en lugar de utilizar el archivo badge.swf o la API air.swf para instalar una aplicación).

Utilización del archivo badge.swf para instalar una aplicación de AIR

Los SDK de AIR y Flex incluyen un archivo badge.swf que permite utilizar fácilmente la función de instalación integrada. El archivo badge.swf puede instalar el motor de ejecución y una aplicación de AIR desde un vínculo en una página web. El archivo badge.swf y su código fuente se le proporcionan para que los distribuya a través de su sitio web.

Incorporación del archivo badge.swf en una página web

- 1 Localice los siguientes archivos, incluidos en el directorio samples/badge de los SDK de AIR y Flex y añádalos a su servidor web.
 - badge.swf

- default_badge.html
- AC_RunActiveContent.js

2 Abra la página default_badge.html en un editor de textos.

3 En la página default_badge.html, en la función de JavaScript `AC_FL_RunContent()`, ajuste las definiciones del parámetro `FlashVars` para lo siguiente:

Parámetro	Descripción
appname	El nombre de la aplicación que muestra el archivo SWF si no está instalado el motor de ejecución.
appurl	(Obligatorio). La URL del archivo de AIR a descargar. Hay que utilizar una URL absoluta (y no una relativa).
airversion	(Obligatorio). Para la versión 1.0 del motor de ejecución, defina esto en 1.0.
imageurl	La URL de la imagen (opcional) a mostrar como logotipo.
buttoncolor	El color del botón de descargar (especificado como valor hexadecimal; por ejemplo, FFCC00).
messagecolor	El color del mensaje de texto que aparece debajo del botón si no está instalado el motor de ejecución (especificado como valor hexadecimal; por ejemplo, FFCC00).

4 El tamaño mínimo del archivo badge.swf es de 217 píxeles de anchura por 180 píxeles de altura. Ajuste los valores de los parámetros `width` y `height` de la función `AC_FL_RunContent()` de acuerdo con sus necesidades.

5 Cambie el nombre del archivo default_badge.html y ajuste su código (o inclúyalo en otra página HTML) para adaptarlo a sus necesidades.

Nota: para la etiqueta `embed HTML` que carga el archivo badge.swf, no establezca el atributo `wmode`; déjelo definido en el valor predeterminado ("`window`"). Otras configuraciones de `wmode` evitarán la instalación en algunos sistemas. Asimismo, el uso de otras configuraciones de `wmode` genera un error: "Error #2044: ErrorEvent no controlado.: text=Error #2074: El escenario es demasiado pequeño para la interfaz de usuario de descarga."

También se puede editar y recompilar el archivo badge.swf. Para obtener más información, consulte "[Modificación del archivo badge.swf](#)" en la página 263.

Instalación de la aplicación de AIR desde un vínculo de instalación integrada en una página web

Una vez que haya añadido a una página el vínculo de instalación integrada, el usuario podrá instalar la aplicación de AIR con solo hacer clic en el vínculo del archivo SWF.

1 Examine la página HTML en un navegador web que disponga de Flash Player instalado (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux).

2 En la página web, haga clic en el vínculo del archivo badge.swf.

- Si ha instalado el motor de ejecución, vaya al paso siguiente.
- Si no ha instalado el motor de ejecución, aparece un cuadro de diálogo que le pregunta si desea instalarlo. Instale el motor de ejecución (consulte "[Instalación de Adobe AIR](#)" en la página 3) y continúe con el siguiente paso.

3 En la ventana Instalación, deje seleccionada la configuración predeterminada y haga clic en Continuar.

En un ordenador con Windows, AIR realiza automáticamente lo siguiente:

- Instala la aplicación en `c:\Archivos de programa\`
- Crea un acceso directo para la aplicación en el escritorio
- Crea un acceso directo en el menú Inicio

- Añade una entrada para la aplicación en Agregar o quitar programas, en el Panel de control

En Mac OS el instalador añade la aplicación al directorio de aplicaciones (por ejemplo, en el directorio /Aplicaciones de Mac OS).

En un ordenador con Linux, AIR realiza automáticamente lo siguiente:

- Instala la aplicación en /opt.
- Crea un acceso directo para la aplicación en el escritorio
- Crea un acceso directo en el menú Inicio
- Añade una entrada para la aplicación en el administrador del paquete del sistema.

4 Seleccione las opciones que desee y haga clic en el botón Instalar.

5 Una vez concluida la instalación, haga clic en Finalizar.

Modificación del archivo badge.swf

El SDK de Flex y AIR proporciona los archivos de origen para el archivo badge.swf. Estos archivos están incluidos en la carpeta samples/badge del SDK:

Archivos de origen	Descripción
badge fla	El archivo de origen de Flash se utiliza para compilar el archivo badge.swf. El archivo badge fla se compila en un archivo SWF 9 (que se puede cargar en Flash Player).
AIRBadge.as	Una clase de ActionScript 3.0 que define la clase de base que se utiliza en el archivo badge fla.

Se puede utilizar Flash Professional para rediseñar la interfaz visual del archivo badge fla.

La función constructora `AIRBadge()`, definida en la clase `AIRBadge`, carga el archivo `air.swf` alojado en <http://airdownload.adobe.com/air/browserapi/air.swf>. El archivo `air.swf` incluye código para utilizar la función de instalación integrada.

El método `onInit()` (en la clase `AIRBadge`) se invoca una vez satisfactoriamente cargado el archivo `air.swf`:

```
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;
    }
}
```

El código define la variable global `_air` en la clase principal del archivo `air.swf`. Esta clase incluye los siguientes métodos públicos, a los que tiene acceso el archivo `badge.swf` para llamar a la función de instalación integrada:

Método	Descripción
<code>getStatus()</code>	<p>Determina si el motor de ejecución está instalado (o puede instalarse) en el ordenador. Para obtener más información, consulte "Cómo comprobar si está instalado el motor de ejecución" en la página 265.</p> <ul style="list-style-type: none"> <code>runtimeVersion</code>: una cadena que indica la versión del motor de ejecución (por ejemplo: "1.0.M6") que requiere la aplicación a instalarse.
<code>installApplication()</code>	<p>Instala la aplicación especificada en el equipo del usuario. Para obtener más información, consulte "Instalación de una aplicación de AIR desde el navegador" en la página 267.</p> <ul style="list-style-type: none"> <code>url</code>: una cadena que define la URL. Hay que utilizar una ruta de URL absoluta (y no una relativa). <code>runtimeVersion</code>: cadena que indica la versión del motor de ejecución (por ejemplo: "2.5.") que requiere la aplicación que se va a instalar. <code>arguments</code>: argumentos a pasarse a la aplicación si se inicia en cuanto se haya instalado. La aplicación se inicia en el momento de instalarla si el elemento <code>allowBrowserInvocation</code> está definido en <code>true</code> en el archivo descriptor de la aplicación. (Para obtener más información sobre el archivo descriptor de la aplicación, consulte "Archivos descriptores de las aplicaciones de AIR" en la página 214.) Si la aplicación se inicia a resultas de una instalación integrada desde el navegador (habiendo el usuario seleccionado iniciarla en cuanto se instalara), el objeto <code>NativeApplication</code> de la aplicación distribuye un objeto <code>BrowserInvokeEvent</code> solo si se pasan argumentos. Tenga en cuenta las consecuencias para la seguridad que pueden tener los datos que pase a la aplicación. Para obtener más información, consulte "Inicio desde el navegador de una aplicación de AIR instalada" en la página 268.

Los valores para `url` y `runtimeVersion` se pasan al archivo SWF a través de las opciones de FlashVars en la página HTML contenedora.

Si la aplicación se inicia automáticamente al instalarla, se puede utilizar la comunicación por LocalConnection para que la aplicación instalada se ponga en contacto con el archivo badge.swf al invocarse. Para obtener más información, consulte [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de ActionScript) o [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de HTML).

También se puede llamar al método `getApplicationVersion()` del archivo `air.swf` para comprobar si una aplicación está instalada. Se puede llamar a este método antes del proceso de instalación de la aplicación o una vez iniciada la instalación. Para obtener más información, consulte [“Cómo comprobar desde una página web si una aplicación de AIR está instalada”](#) en la página 266.

Carga del archivo air.swf

Puede crear su propio archivo SWF que utilice las API del archivo `air.swf` para interactuar con el motor de ejecución y las aplicaciones de AIR desde una página web en un navegador. El archivo `air.swf` está alojado en <http://airdownload.adobe.com/air/browserapi/air.swf>. Para hacer referencia a las API de `air.swf` desde el archivo SWF, cargue el archivo `air.swf` en el mismo dominio de la aplicación que el archivo SWF. El código siguiente muestra un ejemplo de cargar el archivo `air.swf` en el dominio de la aplicación del archivo SWF de carga:

```
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;
}
```

Una vez cargado el archivo `air.swf` (cuando el objeto `contentLoaderInfo` de `Loader` distribuye el evento `init`), se puede llamar a cualquiera de las API de `air.swf`, descritas en las secciones siguientes.

Nota: el archivo `badge.swf` suministrado con los SDK de AIR y Flex carga automáticamente el archivo `air.swf`. Consulte [“Utilización del archivo badge.swf para instalar una aplicación de AIR”](#) en la página 261. Las instrucciones que aparecen en esta sección son para crear su propio archivo SWF que cargue el archivo `air.swf`.

Cómo comprobar si está instalado el motor de ejecución

Un archivo SWF puede comprobar si el motor de ejecución está instalado, llamando al método `getStatus()` en el archivo `air.swf` cargado desde <http://airdownload.adobe.com/air/browserapi/air.swf>. Para obtener más información, consulte [“Carga del archivo air.swf”](#) en la página 265.

Una vez cargado el archivo `air.swf`, el archivo SWF puede llamar al método `getStatus()` del archivo `air.swf` como en el ejemplo siguiente:

```
var status:String = airSWF.getStatus();
```

El método `getStatus()` devuelve uno de los siguientes valores de cadena, basado en el estado del motor de ejecución en el ordenador:

Valor de la cadena	Descripción
"available"	El motor de ejecución puede instalarse en este ordenador pero ahora no está instalado.
"unavailable"	El motor de ejecución no puede instalarse en este ordenador.
"installed"	El motor de ejecución está instalado en este ordenador.

El método `getStatus()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

Cómo comprobar desde una página web si una aplicación de AIR está instalada

Un archivo SWF puede comprobar si una aplicación de AIR (con ID de la aplicación e ID del editor que coincidan) está instalada, llamando al método `getApplicationVersion()` en el archivo `air.swf` cargado desde <http://airdownload.adobe.com/air/browserapi/air.swf>. Para obtener más información, consulte “[Carga del archivo air.swf](#)” en la página 265.

Una vez cargado el archivo `air.swf`, el archivo SWF puede llamar al método `getApplicationVersion()` del archivo `air.swf` como en el ejemplo siguiente:

```
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.
    }
}
```

El método `getApplicationVersion()` utiliza los siguientes parámetros:

Parámetros	Descripción
appID	El ID de la aplicación. Para obtener más información, consulte "id" en la página 236.
pubID	El ID del editor de la aplicación. Para obtener más información, consulte "publisherID" en la página 246. Si la aplicación en cuestión no dispone de un ID de editor, establezca el parámetro pubID en una cadena vacía ("").
callback	Una función callback que cumple la función de controlador. El método <code>getApplicationVersion()</code> funciona de modo asíncrono, y al detectar la versión instalada (o la falta de una), se invoca este método callback. La definición del método callback debe incluir un parámetro, una cadena de caracteres, que se establece como la cadena de la versión de la aplicación instalada. Si la aplicación no está instalada, se pasa un valor nulo a la función, como se muestra en el ejemplo de código anterior.

El método `getApplicationVersion()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

Nota: a partir de AIR 1.5.3, el ID de editor queda desfasado. Los ID de editor no se vuelven a asignar a una aplicación de forma automática. Por compatibilidad con versiones anteriores, las aplicaciones pueden continuar para especificar un ID de editor.

Instalación de una aplicación de AIR desde el navegador

Un archivo SWF puede instalar una aplicación de AIR, llamando al método `installApplication()` en el archivo `air.swf` cargado desde <http://airdownload.adobe.com/air/browserapi/air.swf>. Para obtener más información, consulte "Carga del archivo `air.swf`" en la página 265.

Una vez cargado el archivo `air.swf`, el archivo SWF puede llamar al método `installApplication()` del archivo `air.swf` como en el código siguiente:

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

El método `installApplication()` instala la aplicación especificada en el equipo del usuario. Este método utiliza los siguientes parámetros:

Parámetro	Descripción
url	Una cadena de caracteres que define la URL del archivo de AIR a instalar. Hay que utilizar una ruta de URL absoluta (y no una relativa).
runtimeVersion	Una cadena que indica la versión del motor de ejecución (por ejemplo: "1.0") que requiere la aplicación a instalarse.
arguments	<p>Un conjunto de argumentos a pasarse a la aplicación si se inicia en cuanto se haya instalado. Únicamente los caracteres alfanuméricos se reconocen en los argumentos. Si necesita transmitir otros valores, considere el uso de un esquema de codificación.</p> <p>La aplicación se inicia en el momento de instalarla si el elemento <code>allowBrowserInvocation</code> está definido en <code>true</code> en el archivo descriptor de la aplicación. (Para obtener más información sobre el archivo descriptor de la aplicación, consulte "Archivos descriptores de las aplicaciones de AIR" en la página 214.) Si la aplicación se inicia a resultas de una instalación integrada desde el navegador (habiendo el usuario seleccionado iniciarla en cuanto se instalara), el objeto <code>NativeApplication</code> de la aplicación distribuye un objeto <code>BrowserInvokeEvent</code> solo si se pasan argumentos. Para obtener más información, consulte "Inicio desde el navegador de una aplicación de AIR instalada" en la página 268.</p>

El método `installApplication()` solo funciona cuando se le llama en el controlador de eventos para un evento de usuario, por ejemplo al hacer clic con el ratón.

El método `installApplication()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

En Mac OS, para instalar una versión actualizada de una aplicación el usuario debe contar con privilegios del sistema adecuados para instalar programas en el directorio de aplicaciones (y privilegios de administrador si la aplicación actualiza el motor de ejecución). En Windows, el usuario debe contar con privilegios de administrador.

También se puede llamar al método `getApplicationVersion()` del archivo `air.swf` para comprobar si una aplicación ya está instalada. Se puede llamar a este método antes de que empiece el proceso de instalación de la aplicación o una vez iniciada la instalación. Para obtener más información, consulte [“Cómo comprobar desde una página web si una aplicación de AIR está instalada”](#) en la página 266. Una vez en ejecución la aplicación, esta puede comunicarse con el contenido SWF en el navegador utilizando la clase `LocalConnection`. Para obtener más información, consulte [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de ActionScript) o [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de HTML).

Inicio desde el navegador de una aplicación de AIR instalada

Para utilizar la función de invocación desde el navegador (lo que permite iniciar la aplicación desde el navegador), el archivo descriptor de la aplicación en cuestión debe incluir la siguiente definición:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Para obtener más información sobre el archivo descriptor de la aplicación, consulte [“Archivos descriptores de las aplicaciones de AIR”](#) en la página 214.

Un archivo SWF en el navegador puede iniciar una aplicación de AIR, llamando al método `launchApplication()` en el archivo `air.swf` cargado desde <http://airdownload.adobe.com/air/browserapi/air.swf>. Para obtener más información, consulte [“Carga del archivo air.swf”](#) en la página 265.

Una vez cargado el archivo `air.swf`, el archivo SWF puede llamar al método `launchApplication()` del archivo `air.swf` como en el código siguiente:

```
var appID:String = "com.example.air.myTestApplication";  
var pubID:String = "02D88EED35F84C264A183921344EEA353A629FD.1";  
var arguments:Array = ["launchFromBrowser"]; // Optional  
airSWF.launchApplication(appID, pubID, arguments);
```

El método `launchApplication()` se define en el nivel superior del archivo `air.swf` (que se carga en el dominio de aplicación de la interfaz de usuario del archivo SWF). Al llamar a este método, AIR inicia la aplicación especificada (si está instalada y se permite la invocación desde el navegador mediante la opción `allowBrowserInvocation` del archivo descriptor de la aplicación). El método utiliza los siguientes parámetros:

Parámetro	Descripción
appID	El ID de la aplicación que se va a iniciar. Para obtener más información, consulte "id" en la página 236.
pubID	El ID del editor de la aplicación que se va a iniciar. Para obtener más información, consulte "publisherID" en la página 246. Si la aplicación en cuestión no dispone de un ID de editor, establezca el parámetro pubID en una cadena vacía ("").
arguments	Un conjunto de argumentos que se pasan a la aplicación. El objeto NativeApplication de la aplicación distribuye un evento BrowserInvokeEvent que tiene una propiedad "arguments" definida en este conjunto.. Únicamente los caracteres alfanuméricos se reconocen en los argumentos. Si necesita transmitir otros valores, considere el uso de un esquema de codificación.

El método `launchApplication()` solo funciona cuando se le llama en el controlador de eventos para un evento de usuario, por ejemplo al hacer clic con el ratón.

El método `launchApplication()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

Si el elemento `allowBrowserInvocation` está definido en `false` en el archivo descriptor de la aplicación, la llamada al método `launchApplication()` no surtirá efecto.

Antes de presentar la interfaz de usuario para iniciar la aplicación, puede ser conveniente llamar al método `getApplicationVersion()` en el archivo `air.swf`. Para obtener más información, consulte ["Cómo comprobar desde una página web si una aplicación de AIR está instalada"](#) en la página 266.

Cuando se invoca la aplicación a través de la función de invocación desde el navegador, el objeto `NativeApplication` de la aplicación distribuye un objeto `BrowserInvokeEvent`. Para obtener información, consulte [Invocación de una aplicación de AIR desde el navegador](#) (para desarrolladores de ActionScript), o bien, [Invocación de una aplicación de AIR desde el navegador](#) (para desarrolladores de HTML).

Si utiliza la función de invocación desde el navegador, asegúrese de tener en cuenta las posibles consecuencias para la seguridad. Estas implicaciones se describen en [Invocación de una aplicación de AIR desde el navegador](#) (para desarrolladores de ActionScript) e [Invocación de una aplicación de AIR desde el navegador](#) (para desarrolladores de HTML).

Una vez en ejecución la aplicación, esta puede comunicarse con el contenido SWF en el navegador utilizando la clase `LocalConnection`. Para obtener más información, consulte [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de ActionScript) o [Comunicación con otras instancias de Flash Player y AIR](#) (para desarrolladores de HTML).

Nota: a partir de AIR 1.5.3, el ID de editor queda desfasado. Los ID de editor no se vuelven a asignar a una aplicación de forma automática. Por compatibilidad con versiones anteriores, las aplicaciones pueden continuar para especificar un ID de editor.

Capítulo 17: Actualización de aplicaciones de AIR

Los usuarios pueden instalar o actualizar cualquier aplicación de AIR haciendo doble clic en el archivo de AIR de su equipo o desde un navegador (mediante la perfeccionada función de instalación). El instalador de Adobe® AIR™ gestiona la instalación y avisa al usuario si está actualizando una aplicación previa existente.

Sin embargo, también es posible permitir que las propias aplicaciones se actualicen solas mediante la clase Updater. (Una aplicación instalada puede detectar nuevas versiones disponibles para su descarga e instalación.) La clase Updater incluye un método `update()` que permite al usuario apuntar a un archivo de AIR de un equipo y actualizar a dicha versión. La aplicación se debe empaquetar como archivo de AIR para poder utilizar la clase Updater. Las aplicaciones empaquetadas como paquetes o ejecutables nativos deben utilizar las facilidades de actualización proporcionadas por la plataforma nativa.

Tanto el ID de aplicación como el ID de editor de un archivo de actualización de AIR deben coincidir para que la aplicación se actualice. El ID de editor proviene del certificado de firma. Tanto la actualización como la aplicación que va a actualizarse deben estar firmadas con el mismo certificado.

Para AIR 1.5.3 o posterior, el archivo descriptor de la aplicación incluye un elemento `<publisherID>`. Este elemento debe usarse si existen versiones de la aplicación desarrolladas utilizando AIR 1.5.2 o una versión anterior. Para obtener más información, consulte “[publisherID](#)” en la página 246.

En AIR 1.1 y posterior, es posible migrar una aplicación para utilizar un nuevo certificado de firma para el código. Para migrar una aplicación y utilizar una nueva firma, es preciso firmar el archivo de actualización de AIR con el certificado nuevo y con el original. La migración de certificados es un proceso que no se puede invertir. Una vez concluida la migración, solo se reconocerán como actualizaciones de la instalación existente aquellos archivos de AIR firmados con el nuevo certificado (o con ambos certificados).

La administración de las actualizaciones de aplicaciones puede resultar un proceso complicado. AIR 1.5 incluye el nuevo *marco de actualización para las aplicaciones de Adobe AIR*. Este marco proporciona las API que ayudan a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR.

Puede utilizar la migración de certificados para pasar de un certificado firmado automáticamente a un certificado comercial de firma de código, o de uno firmado automáticamente a otro del mismo tipo. Si no migra el certificado, los usuarios existentes deberán quitar su versión actual de la aplicación para poder instalar la nueva versión. Para obtener más información, consulte “[Cambio de certificado](#)” en la página 204.

Se recomienda incluir un mecanismo de actualización en la aplicación. Si se crea una nueva versión de la aplicación, el mecanismo de actualización puede indicar al usuario que instale la nueva versión.

El instalador de aplicaciones de AIR crea archivos de registro cuando se instala, se actualiza o se elimina una aplicación de AIR. Puede consultar estos registros para ayudar a determinar la causa de cualquier problema de instalación. Consulte [Installation logs](#) (Registros de instalación; en inglés).

Nota: las nuevas versiones del motor de ejecución Adobe AIR puede incluir versiones actualizadas de WebKit. Una versión actualizada de WebKit puede implicar cambios inesperados en el contenido HTML de una aplicación implementada de AIR. Estos cambios pueden requerir la actualización de la aplicación. Un mecanismo de actualización puede informar al usuario de la nueva versión de la aplicación. Para obtener más información, consulte [Entorno HTML](#) (para desarrolladores de ActionScript) o [Entorno HTML](#) (para desarrolladores HTML).

Actualización de aplicaciones

La clase `Updater` (del paquete `flash.desktop`) incluye un método, `update()`, que se puede utilizar para actualizar la aplicación actualmente en ejecución a una versión distinta. Por ejemplo, si el usuario tiene una versión del archivo de AIR ("Sample_App_v2.air") en el escritorio, el siguiente código actualizaría la aplicación:

Ejemplo de ActionScript:

```
var updater:Updater = new Updater();
var airFile:File = File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version:String = "2.01";
updater.update(airFile, version);
```

Ejemplo de JavaScript:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Antes de que una aplicación utilice la clase `Updater`, el usuario o la aplicación deben descargar la versión actualizada del archivo de AIR en el equipo. Para obtener más información, consulte [“Descarga de un archivo de AIR en el equipo del usuario”](#) en la página 273.

Resultados de la llamada al método `Updater.update()`

Cuando una aplicación del motor de ejecución llama al método `update()`, este cierra la aplicación y, a continuación, intenta instalar la nueva versión del archivo de AIR. Se comprueba que el ID de aplicación y el ID de editor especificados en el archivo de AIR coinciden con el ID de aplicación y de editor de la aplicación que llama al método `update()`. (Para obtener más información sobre el ID de aplicación y el ID de editor, consulte [“Archivos descriptores de las aplicaciones de AIR”](#) en la página 214.) También comprueba si la cadena de versión coincide con la cadena `version` transferida al método `update()`. Si la instalación concluye correctamente, el motor de ejecución abre la nueva versión de la aplicación. En caso contrario (si la instalación no concluye correctamente), vuelve a abrir la versión existente de la aplicación (previa a la instalación).

En Mac OS, para instalar una versión actualizada de una aplicación, el usuario debe contar con adecuados privilegios del sistema para instalar en el directorio de la aplicación. En Windows y Linux, el usuario debe disponer de privilegios de administrador.

Si la versión actualizada de la aplicación requiere una versión actualizada del motor de ejecución, se instala la versión más reciente del motor de ejecución. Para actualizar el motor de ejecución, el usuario debe tener privilegios administrativos para el equipo.

Al verificar una aplicación con ADL, llamar al método `update()` produce una excepción de tiempo de ejecución.

Cadena de versión

Para que el archivo de AIR se pueda instalar, la cadena que se especifica como parámetro `version` del método `update()` debe coincidir con el elemento `version` o `versionNumber` del archivo descriptor de la aplicación para el archivo de AIR que va a instalarse. Es preciso especificar el parámetro `version` por motivos de seguridad. Al solicitar a la aplicación que verifique el número de versión en el archivo de AIR, la aplicación no instalará por error una versión anterior. (Una versión anterior de la aplicación puede presentar una vulnerabilidad de seguridad que se haya solucionado en la aplicación instalada actualmente.) La aplicación también comprueba la cadena de versión en el archivo de AIR y la compara con la de la aplicación instalada para evitar desactualizaciones.

Antes de AIR 2.5, la cadena de la versión puede ser de cualquier formato. Por ejemplo, "2.01" o "versión 2". En AIR 2.5, o posterior, la cadena de versión debe ser una secuencia de hasta números de tres dígitos separados por puntos. Por ejemplo, ".0", "1.0" y "67.89.999" sin números válidos de versión. Se debe validar la cadena de la versión de actualización antes de actualizar la aplicación.

Si una aplicación de Adobe AIR descarga un archivo de AIR por Internet, se recomienda disponer de un mecanismo que permite al servicio web notificar a la aplicación de Adobe AIR sobre la descarga actual de la versión. La aplicación puede utilizar esta cadena como el parámetro `version` del método `update()`. Si el archivo de AIR se obtiene por otros medios que impiden conocer la versión del archivo de AIR, la aplicación de AIR puede examinar el archivo de AIR para extraer la información sobre su versión. (Un archivo de AIR es un archivo ZIP comprimido y el archivo descriptor de la aplicación es el segundo registro del archivo.)

Para obtener más información sobre el archivo descriptor de la aplicación, consulte ["Archivos descriptores de las aplicaciones de AIR"](#) en la página 214.

Flujo de trabajo de firma para actualizaciones de la aplicación

La publicación de actualizaciones de forma ad-hoc complica las tareas de administración de varias versiones de la aplicación y dificulta el seguimiento de las fechas de caducidad del certificado. Los certificados pueden caducar antes de que se pueda publicar una actualización.

El motor de ejecución de Adobe AIR trata una actualización de la aplicación publicada sin una firma de migración como una nueva aplicación. Los usuarios deben desinstalar su aplicación actual de AIR antes de que puedan instalar la actualización de la aplicación.

Para resolver el problema, cargue cada aplicación actualizada con el certificado más reciente en una URL de implementación independiente. Incluya un mecanismo que le recuerde aplicar firmas de migración cuando el certificado esté dentro del periodo de gracia de 180 días. Consulte ["Firma de una versión actualizada de una aplicación de AIR"](#) en la página 209 para obtener más información.

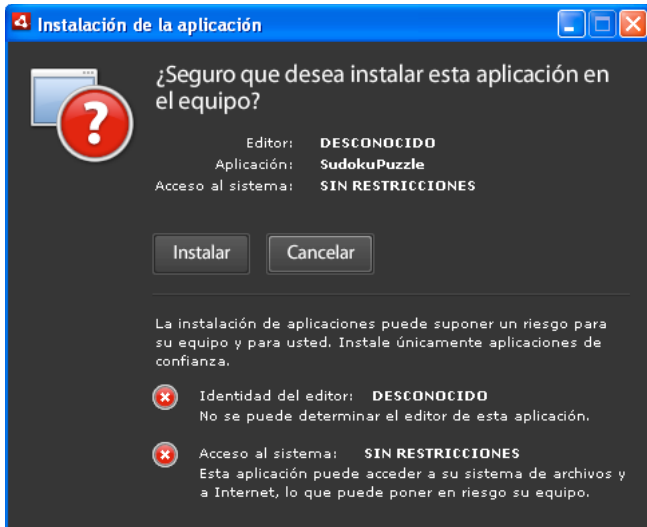
Consulte ["Comandos de ADT"](#) en la página 174 para obtener información sobre cómo aplicar firmas.

Realice las siguientes tareas para simplificar el proceso de aplicación de firmas de migración:

- Cargue todas las aplicaciones actualizadas en una URL de implementación independiente.
- Cargue el archivo XML del descriptor de actualización y el certificado más reciente para la actualización en la misma URL.
- Firme la aplicación actualizada con el certificado más reciente.
- Aplique una firma de migración a la aplicación actualizada con el certificado para firmar la versión anterior ubicada en una URL diferente.

Presentación de una interfaz de usuario personalizada para las actualizaciones de la aplicación

AIR incluye una interfaz de actualización predeterminada:



Esta interfaz siempre se utiliza la primera vez que el usuario instala la versión de una aplicación en un ordenador. Sin embargo, es posible definir una interfaz propia para utilizarla en el futuro. Si su aplicación define una interfaz de actualización personalizada, especifique un elemento `customUpdateUI` en el archivo descriptor de la aplicación para la aplicación instalada actualmente:

```
<customUpdateUI>true</customUpdateUI>
```

Cuando la aplicación se instale y el usuario abra un archivo de AIR con un ID de aplicación e ID de editor que coincidan con los de la aplicación instalada, será el motor de ejecución el encargado de abrir la aplicación, no el archivo de instalación predeterminado de la aplicación de AIR. Para obtener más información, consulte “[customUpdateUI](#)” en la página 226.

La aplicación puede decidir, cuando se ejecute (cuando el objeto `NativeApplication.nativeApplication` distribuye un evento `load`), si actualiza o no la aplicación (con la clase `Updater`). Si decide actualizarse, puede presentar su propia interfaz de instalación al usuario (interfaz que no es igual que la estándar).

Descarga de un archivo de AIR en el equipo del usuario

Para poder utilizar la clase `Updater`, el usuario o la aplicación deben guardar primero localmente un archivo de AIR en el equipo del usuario.

Nota: AIR 1.5 incluye un marco de actualización que ayuda a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR. El uso de este marco puede resultar mucho más sencillo que la utilización del método `update()` de la clase `Update` directamente. Para obtener más información, consulte “[Utilización del marco de actualización](#)” en la página 277.

El siguiente código lee un archivo de AIR desde una dirección URL (http://example.com/air/updates/Sample_App_v2.air) y lo guarda en el directorio de almacenamiento de la aplicación:

Ejemplo de ActionScript:

```
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.");
}
```

Ejemplo de JavaScript:

```
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.");
}
```

Para obtener más información, consulte:

- [Flujo de trabajo de lectura y escritura de archivos](#) (para desarrolladores de ActionScript)
- [Flujo de trabajo de lectura y escritura de archivos](#) (para desarrolladores de HTML)

Comprobar si una aplicación se está ejecutando por primera vez

Una vez actualizada la aplicación, puede ofrecer al usuario un mensaje de bienvenida o de primeros pasos. Al iniciarse, la aplicación comprueba si se está ejecutando por primera vez para saber si debe mostrar o no el mensaje.

***Nota:** AIR 1.5 incluye un marco de actualización que ayuda a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR. Este marco proporciona métodos sencillos para comprobar si una versión de una aplicación se está ejecutando por primera vez. Para obtener más información, consulte [“Utilización del marco de actualización”](#) en la página 277.*

Una forma de hacerlo es guardar un archivo en el directorio de almacenamiento de la aplicación al inicializarla. Cada vez que se inicie la aplicación, comprobará si existe este archivo. Si el archivo no existe, significa que la aplicación se está ejecutando por primera vez para el usuario. Si el archivo existe, significa que el usuario ya ha ejecutado la aplicación al menos una vez. Si el archivo existe y su número de versión es anterior a la versión actual, significa que el usuario está ejecutando la aplicación por primera vez.

Este concepto se demuestra en el siguiente ejemplo de Flex:

```
<?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";
        }
      }
    ]]>
  </mx:Script>
</mx:WindowedApplication>
```



```
        } 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>
```

Este concepto se muestra en el siguiente ejemplo de JavaScript:

```
<html>
  <head>
    <script src="AIRAliases.js" />
    <script>
      var file;
      var currentVersion = "1.2";
      function system extension() {
        file = air.File.appStorageDirectory.resolvePath("Preferences/version.txt");
        air.trace(file.nativePath);
        if(file.exists) {
          checkVersion();
        } else {
          firstRun();
        }
      }
      function checkVersion() {
        var stream = new air.FileStream();
        stream.open(file, air.FileMode.READ);
        var reversion = stream.readUTFBytes(stream.bytesAvailable);
        stream.close();
        if (reversion != currentVersion) {
          window.document.getElementById("log").innerHTML
            = "You have updated to version " + currentVersion + ".\n";
        } else {
          saveFile();
        }
        window.document.getElementById("log").innerHTML
          += "Welcome to the application.";
```

```
    }  
    function firstRun() {  
        window.document.getElementById("log").innerHTML  
            = "Thank you for installing the application. \n"  
            + "This is the first time you have run it.";  
        saveFile();  
    }  
    function saveFile() {  
        var stream = new air.FileStream();  
        stream.open(file, air.FileMode.WRITE);  
        stream.writeUTFBytes(currentVersion);  
        stream.close();  
    }  
    </script>  
</head>  
<body onLoad="system extension()">  
    <textarea ID="log" rows="100%" cols="100%" />  
</body>  
</html>
```

Si la aplicación guarda datos localmente (por ejemplo, en el directorio de almacenamiento de la aplicación), puede buscar datos guardados previamente (de versiones anteriores) durante la primera ejecución de la aplicación.

Utilización del marco de actualización

La administración de actualizaciones en las aplicaciones puede resultar una tarea tediosa. Las aplicaciones de *AIR del marco de actualización para Adobe* proporcionan las APIs que permiten que los desarrolladores proporcionen capacidades sólidas de actualización en las aplicaciones de AIR. El marco de actualización de AIR realiza las siguientes tareas para los desarrolladores:

- Buscar periódicamente actualizaciones en función de un intervalo o cuando el usuario lo solicita.
- Descargar archivos de AIR (actualizaciones) desde un origen web.
- Alertar al usuario la primera vez que ejecuta la versión recién instalada.
- Confirmar que el usuario desea comprobar si hay actualizaciones.
- Mostrar al usuario información sobre la versión de la nueva actualización.
- Mostrar al usuario el progreso de la descarga y la información sobre los posibles errores.

El marco de actualización de AIR proporciona una interfaz de usuario de ejemplo para su aplicación. Proporciona al usuario información básica y opciones de configuración para las actualizaciones de la aplicación. La aplicación también puede definir una interfaz de usuario para su uso con el marco de actualización.

El marco de actualización de AIR permite almacenar información sobre la versión de actualización de una aplicación de AIR en sencillos archivos de configuración XML. En la mayoría de aplicaciones, al definir los archivos de configuración e incluir algo de código básico, los resultados de funcionalidad de la actualización son buenos para el usuario.

Aun sin el uso del marco de actualización, Adobe AIR incluye una clase Updater que las aplicaciones de AIR pueden emplear para actualizar a nuevas versiones. La clase Updater permite que una aplicación se actualice a una versión incluida en un archivo de AIR en el equipo del usuario. No obstante, la administración de la actualización puede implicar un proceso más complejo que el basar simplemente la actualización de la aplicación en un archivo de AIR almacenado localmente.

Archivos en el marco de actualización de AIR

El marco de actualización de AIR se incluye en el directorio `frameworks/libs/air` del SDK de AIR 2. Incluye los siguientes archivos:

- `applicationupdater.swc`: define la funcionalidad básica de la biblioteca de actualización, para su uso en ActionScript. Esta versión no contiene interfaz de usuario.
- `applicationupdater.swf`: define la funcionalidad básica de la biblioteca de actualización, para su uso en JavaScript. Esta versión no contiene interfaz de usuario.
- `applicationupdater_ui.swc`: define la funcionalidad básica de la biblioteca de actualización de la versión Flex 4, incluyendo una interfaz de usuario que la aplicación puede utilizar para mostrar las opciones de actualización
- `applicationupdater_ui.swf`: define la funcionalidad básica de la biblioteca de actualización de la versión de JavaScript, incluyendo una interfaz de usuario que la aplicación puede utilizar para mostrar las opciones de actualización

Para obtener más información, consulte las siguientes secciones:

- [“Configuración del entorno de desarrollo de Flex”](#) en la página 278
- [“Inclusión de archivos del marco en una aplicación de AIR basada en HTML”](#) en la página 279
- [“Ejemplo básico: Uso de la versión ApplicationUpdaterUI”](#) en la página 279

Configuración del entorno de desarrollo de Flex

Los archivos SWC del directorio `frameworks/libs/air` del SDK de AIR 2 definen clases que se pueden utilizar en el desarrollo de Flex y Flash.

Para utilizar el marco de actualización al compilar con el SDK de Flex, incluya el archivo `ApplicationUpdater.swc` o `ApplicationUpdater_UI.swc` en la llamada al compilador `amxmlc`. En el siguiente ejemplo, el compilador carga el archivo `ApplicationUpdater.swc` en el subdirectorio `lib` del directorio Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater.swc -- myApp.mxml
```

En el siguiente ejemplo, el compilador carga el archivo `ApplicationUpdater_UI.swc` en el subdirectorio `lib` del directorio Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc -- myApp.mxml
```

Al realizar el desarrollo con el uso de Flash Builder, añada el archivo SWC en la ficha de ruta de biblioteca de la configuración de la ruta de compilación de Flex en el cuadro de diálogo de propiedades.

Asegúrese de copiar los archivos SWC en el directorio al que se hará referencia en el compilador `amxmlc` (usando el SDK de Flex) o Flash Builder.

Inclusión de archivos del marco en una aplicación de AIR basada en HTML

El directorio `frameworks/html` del marco de actualización incluye los siguientes archivos SWF:

- `ApplicationUpdater.swf`: define la funcionalidad básica de la biblioteca de actualización, sin ninguna interfaz de usuario.
- `ApplicationUpdater_UI.swf`: define la funcionalidad básica de la biblioteca de actualización, incluyendo una interfaz de usuario que la aplicación puede utilizar para mostrar las opciones de actualización.

El código JavaScript de las aplicaciones de AIR puede utilizar clases definidas en archivos SWF.

Para utilizar el marco de actualización, incluya el archivo `ApplicationUpdater.swf` o `ApplicationUpdater_UI.swf` en el directorio de la aplicación (o un subdirectorio) A continuación, en el archivo HTML que utilizará el marco (en código JavaScript), incluya una etiqueta `script` que cargue el archivo:

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

O bien, utilice esta etiqueta `script` para cargar el archivo `ApplicationUpdater_UI.swf`:

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

La API definida en estos dos archivos se describe en el resto del documento.

Ejemplo básico: Uso de la versión `ApplicationUpdaterUI`

La versión `ApplicationUpdaterUI` del marco de actualización proporciona una interfaz básica que se puede emplear fácilmente en la aplicación. A continuación se incluye un ejemplo básico.

En primer lugar, cree una aplicación de AIR que llame al marco de actualización:

- 1 Si su aplicación es una aplicación de AIR basada en HTML, cargue el archivo `applicationupdaterui.swf`:

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

- 2 En la lógica del programa de su aplicación de AIR, cree una instancia de un objeto `ApplicationUpdaterUI`.

En `ActionScript`, utilice el siguiente código:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

En `JavaScript`, utilice el siguiente código:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Es posible que desee añadir este código en una función de inicialización que se ejecute una vez cargada la aplicación.

- 3 Cree un archivo de texto denominado `updateConfig.xml` y añádale lo siguiente:

```
<?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>
```

Edite el elemento `URL` del archivo `updateConfig.xml` para que coincida con la ubicación final del archivo descriptor de actualización en su servidor web (consulte el siguiente procedimiento).

`delay` es el número de días que la aplicación espera entre las búsquedas de actualizaciones.

- 4 Añada el archivo `updateConfig.xml` al directorio del proyecto de su aplicación de AIR.
- 5 Haga que el objeto `updater` haga referencia al archivo `updateConfig.xml` y llame al método `initialize()` del objeto.

En ActionScript, utilice el siguiente código:

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");  
appUpdater.initialize();
```

En JavaScript, utilice el siguiente código:

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");  
appUpdater.initialize();
```

- 6 Cree una segunda versión de la aplicación de AIR que tenga una versión distinta a la primera aplicación. (La versión se especifica en el archivo descriptor de la aplicación, en el elemento `version`.)

A continuación, añada la versión de actualización de la aplicación de AIR al servidor web:

- 1 Sitúe la versión de actualización del archivo de AIR en el servidor web.
- 2 Cree un archivo de texto denominado `updateDescriptor.2.5.xml` y agréguele el siguiente contenido:

```
<?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>
```

Edite los elementos `versionNumber`, URL y `description` del archivo `updateDescriptor.xml` para que coincida con el archivo de AIR de actualización. Este formato del descriptor de actualización se utiliza en las aplicaciones usando el marco de actualización incluido en el SDK de AIR 2.5 (y posterior).

- 3 Cree un archivo de texto denominado `updateDescriptor.1.0.xml` y agréguele el siguiente contenido:

```
<?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>
```

Edite los elementos `version`, URL y `description` del archivo `updateDescriptor.xml` para que coincida con el archivo de AIR de actualización. El formato descriptor de actualización se utiliza en las aplicaciones con el uso del marco de actualización incluido en el SDK de AIR 2 (y posterior).

Nota: la creación de este segundo archivo descriptor de la aplicación solo es necesario cuando se admiten actualizadas en las aplicaciones creadas antes de AIR 2.5.

- 4 Añada el archivo `updateDescriptor.2.5.xml` y `updateDescriptor.1.0.xml` al mismo directorio del servidor web que contenga el archivo de AIR de actualización.

Este es un ejemplo básico, pero proporciona la funcionalidad de actualización suficiente para diversas aplicaciones. En el resto del documento se describe cómo emplear el marco de actualización para que mejor se adapte a sus necesidades.

Para obtener otro ejemplo del uso de marco de actualización, consulte la siguiente aplicación de ejemplo en el centro de desarrollo de Adobe AIR:

- [Update Framework in a Flash-based Application](http://www.adobe.com/go/learn_air_qs_update_framework_flash_es) (Marco de actualización en una aplicación basada en Flash; en inglés)(http://www.adobe.com/go/learn_air_qs_update_framework_flash_es)

Actualización a AIR 2.5

Debido a que las reglas para asignar números de versión a las aplicaciones cambiaron en AIR 2.5, el marco de actualización de AIR 2 no puede analizar la información de versión en un descriptor de la aplicación de AIR 2.5. Esta incompatibilidad significa que se debe actualizar la aplicación para que utilice el nuevo marco de actualización ANTES de que la aplicación se actualice para que utilice el SDK de AIR 2.5. De este modo, la actualización de la aplicación a AIR 2.5 o posterior desde cualquier versión de AIR anterior a 2.5, requiere DOS actualizaciones. La primera debe emplear el espacio de nombres de AIR 2 e incluir la biblioteca del marco de actualización de AIR 2.5 (aún se puede crear el paquete de la aplicación utilizando el SDK de AIR 2.5). La segunda actualización debe utilizar el espacio de nombres de AIR 2.5 e incluir las nuevas funciones de la aplicación.

También se puede hacer que la actualización intermedia no realice nada excepto actualizar la aplicación de AIR 2.5 utilizando la clase Updater de AIR directamente.

En el siguiente ejemplo se muestra cómo actualizar una aplicación de la versión 1.0 a 2.0. La versión 1.0 utiliza el antiguo espacio de nombres 2.0. La versión 2.0 usa el espacio de nombres 2.5 y dispone de las nuevas funciones implementadas utilizando las API de AIR 2.5.

1 Cree una versión intermedia de la aplicación, versión 1.0.1, basada en la versión 1.0 de la aplicación.

a Utilice el marco del actualizador de aplicaciones de AIR 2.5 mientras se crea la aplicación.

***Nota:** emplee `applicationupdater.swc 0 applicationupdater_ui.swc` para las aplicaciones de AIR basadas en tecnología Flash y `applicationupdater.swf 0 applicationupdater_ui.swf` para las aplicaciones de AIR basadas en HTML.*

b Cree un archivo descriptor de actualización para la versión 1.0.1 utilizando el espacio de nombres y la versión tal y como se muestra a continuación:

```
<?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 Cree la versión 2.0 de la aplicación que utiliza el espacio de nombres 2.5 y de las APIs de AIR 2.5.

3 Cree un descriptor de actualización para actualizar la aplicación de la versión 1.0.1 a la versión 2.0.

```
<?xml version="1.0" encoding="utf-8"?>
  <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
    <version>2.0</version>
    <url>http://example.com/updates/sample_2.0.air</url>
    <description>This is the intermediate version.</description>
  </update>
```

Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web

Cuando se utiliza el marco de actualización de AIR, se define la información básica sobre la actualización disponible en un archivo descriptor de actualización, almacenado en el servidor web. Un archivo descriptor de actualización es un sencillo archivo XML. El marco de actualización incluido en la aplicación comprueba este archivo para ver si se ha cargado una nueva versión.

Formato del archivo descriptor de actualización modificado para AIR 2.5. El nuevo formato utiliza un espacio de nombres distinto. El espacio de nombres original es "http://ns.adobe.com/air/framework/update/description/1.0". El espacio de nombres de AIR 2.5 es "http://ns.adobe.com/air/framework/update/description/2.5".

Las aplicaciones de AIR creadas antes de AIR 2.5 solo pueden leer el descriptor de actualización 1.0 de la versión. Las aplicaciones de AIR creadas utilizando el marco actualizador incluido en AIR 2.5 o posterior solo pueden leer el descriptor de actualización de la versión 2.5. Debido a esta incompatibilidad de versiones, en ocasiones es necesario crear dos archivos del descriptor de actualización. La lógica de actualización en las versiones de AIR 2.5 de la aplicación debe descargar un descriptor de actualización que utilice el nuevo formato. Las versiones anteriores de la aplicación de AIR deben continuar utilizando el formato original. Ambos archivos deben modificarse para cada actualización que se publique (hasta que deje de admitir versiones creadas antes de AIR 2.5).

El archivo descriptor de actualización contiene los siguientes datos:

- `versionNumber`: la nueva versión de la aplicación de AIR. Utilice el elemento `versionNumber` en los descriptors de actualización para actualizar las aplicaciones de AIR 2.5. El valor debe ser la misma cadena que se utiliza en el elemento `versionNumber` del nuevo archivo descriptor de la aplicación de AIR. El número de versión del archivo descriptor no coincide con el número de versión del archivo de actualización de AIR, el marco de actualización emitirá una excepción.
- `version`: nueva versión de la aplicación de AIR. Utilice el elemento `version` en los descriptors de actualización utilizados para actualizar las aplicaciones creadas antes de AIR 2.5. El valor debe ser la misma cadena empleada en el elemento de la versión `version` del nuevo archivo descriptor de la aplicación de AIR. Si la versión del archivo descriptor no coincide con la versión del archivo de actualización de AIR, el marco de actualización emitirá una excepción.
- `versionLabel`: cadena de versión que puede leer el usuario para mostrar a los usuarios. `versionLabel` es opcional, pero solo se puede especificar en los archivos del descriptor de actualización de la versión 2.5. Utilícelo si se emplea un elemento `versionLabel` en el descriptor de la aplicación y establézcalo en el mismo valor.
- `url`: ubicación del archivo de actualización de AIR. Se trata del archivo que contiene la versión de actualización de la aplicación de AIR.
- `description`: información sobre la nueva versión. Esta información se puede mostrar al usuario durante el proceso de actualización.

Los elementos `version` y `url` son obligatorios. El elemento `description` es opcional.

A continuación se incluye un archivo descriptor de actualización de ejemplo de la versión 2.5:

```
<?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>
```

A continuación se incluye un archivo descriptor de actualización de ejemplo de la versión 1.0:

```
<?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>
```

Si desea definir la etiqueta `description` mediante varios idiomas, utilice varios elementos `text` que definan un atributo `lang`:

```
<?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>
```

Sitúe el archivo descriptor de actualización, junto con el archivo de actualización de AIR, en el servidor web.

El directorio de plantillas incluido con el descriptor de actualización contiene archivos descriptores de actualización de ejemplo. Estos incluyen versiones en un solo idioma y varios idiomas.

Creación de una instancia del objeto updater

Tras cargar el marco de actualización de AIR en su código (consulte “[Configuración del entorno de desarrollo de Flex](#)” en la página 278 e “[Inclusión de archivos del marco en una aplicación de AIR basada en HTML](#)” en la página 279), es necesario crear una instancia del objeto updater, tal y como se muestra en el siguiente ejemplo:

Ejemplo de ActionScript:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

Ejemplo de JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

El código anterior utiliza la clase `ApplicationUpdater` (que no proporciona ninguna interfaz de usuario). Si desea usar la clase `ApplicationUpdaterUI` (que proporciona una interfaz de usuario), utilice lo siguiente:

Ejemplo de ActionScript:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Ejemplo de JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

En los ejemplos de código restantes de este documento se presupone que se ha creado una instancia de un objeto updater denominada `appUpdater`.

Definición de la configuración de actualización

Tanto `ApplicationUpdater` como `ApplicationUpdaterUI` se pueden configurar mediante un archivo de configuración incluido con la aplicación, o bien, a través de ActionScript o JavaScript en la aplicación.

Definición de la configuración de actualización en un archivo de configuración XML

El archivo de configuración de actualización es un archivo XML. Puede incluir los siguientes elementos:

- `updateURL`: una cadena. Representa la ubicación del descriptor de actualización en el servidor remoto. Se permite cualquier ubicación URLRequest válida. Se debe definir la propiedad `updateURL`, a través del archivo de configuración o mediante un script (consulte “[Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web](#)” en la página 281). Esta propiedad se debe definir antes de utilizar el elemento `updater` (antes de llamar al método `initialize()` del objeto `updater`, descrito en “[Inicialización del marco de actualización](#)” en la página 286).

- `delay`: un número. Representa un intervalo de tiempo determinado en días (se permiten valores similares a 0, 25) para buscar actualizaciones. El valor de 0 (valor predeterminado) especifica que updater no realiza ninguna comprobación periódica automática.

El archivo de configuración para `ApplicationUpdaterUI` puede incluir el siguiente elemento además de los elementos `updateURL` y `delay`:

- `defaultUI`: una lista de elementos `dialog`. Cada elemento `dialog` dispone de un atributo `name` que se corresponde con el cuadro de diálogo en la interfaz de usuario. Todos los elementos `dialog` cuentan con un atributo `visible` que define si el cuadro de diálogo es visible. El valor predeterminado es `true`. Entre los posibles valores para el atributo `name` se encuentran los siguientes:
 - `"checkForUpdate"`: corresponde a los cuadros de diálogo de búsqueda de actualizaciones, sin actualizaciones y error de actualización.
 - `"downloadUpdate"`: corresponde al cuadro de diálogo de descarga de actualizaciones.
 - `"downloadProgress"`: corresponde a los cuadros de diálogo de progreso y error de descarga.
 - `"installUpdate"`: corresponde al cuadro de diálogo de instalación de la actualización.
 - `"fileUpdate"`: corresponde a los cuadros de diálogo de actualización de archivo, no actualización de archivo y error de archivo.
- `"unexpectedError"`: corresponde al cuadro de diálogo de error inesperado.

Cuando se establece en `false`, el cuadro de diálogo correspondiente no aparece como parte del proceso de actualización.

A continuación se incluye un ejemplo del archivo de configuración para el marco `ApplicationUpdater`:

```
<?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>
```

Se muestra un ejemplo del archivo de configuración para el marco `ApplicationUpdaterUI`, que incluye una definición para el elemento `defaultUI`:

```
<?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>
```

Seleccione la propiedad `configurationFile` en la ubicación de ese archivo:

Ejemplo de ActionScript:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

Ejemplo de JavaScript:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

El directorio de plantillas del marco de actualización incluye un archivo de configuración de ejemplo, `config-template.xml`.

Definición de la configuración de actualización con código ActionScript o JavaScript

Estos parámetros de configuración también se pueden establecer utilizando código en la aplicación, tal y como se muestra a continuación:

```
appUpdater.updateURL = " http://example.com/updates/update.xml ";  
appUpdater.delay = 1;
```

Las propiedades del objeto `updater` son `updateURL` y `delay`. Estas propiedades definen la misma configuración que los elementos `updateURL` y `delay` en el archivo de configuración: la dirección URL del archivo descriptor de actualización y el intervalo para buscar actualizaciones. Si se especifica un archivo de configuración y la configuración en código, todas las propiedades establecidas utilizando código tienen prioridad sobre la configuración correspondiente en el archivo de configuración.

Se debe definir la propiedad `updateURL`, a través del archivo de configuración o mediante script (consulte [“Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web”](#) en la página 281) antes de utilizar `updater` (antes de llamar al método `initialize()` del objeto `updater`, descrito en [“Inicialización del marco de actualización”](#) en la página 286).

El marco `ApplicationUpdaterUI` define estas propiedades adicionales del objeto `updater`:

- `isCheckForUpdateVisible`: corresponde a los cuadros de diálogo de búsqueda de actualizaciones, sin actualizaciones y error de actualización.
- `isDownloadUpdateVisible`: corresponde al cuadro de diálogo de descarga de actualizaciones.
- `isDownloadProgressVisible`: corresponde a los cuadros de diálogo de progreso y error de descarga.
- `isInstallUpdateVisible`: corresponde al cuadro de diálogo de instalación de la actualización.
- `isFileUpdateVisible`: corresponde a los cuadros de diálogo de actualización de archivo, no actualización de archivo y error de archivo.
- `isUnexpectedErrorVisible`: corresponde al cuadro de diálogo de error inesperado.

Todas las propiedades hacen referencia a uno o varios cuadros de diálogo en la interfaz de usuario `ApplicationUpdaterUI`. Cada propiedad es un valor booleano, con un valor predeterminado de `true`. Cuando se establece en `false`, los cuadros de diálogo correspondientes no aparecen como parte del proceso de actualización.

Estas propiedades de cuadro de diálogo anulan la configuración del archivo de configuración de actualización.

Proceso de actualización

El marco de actualización de AIR completa el proceso de actualización en los siguientes pasos:

- 1 Con la inicialización de `updater` se verifica si se ha realizado una comprobación de actualización en el intervalo de días definido (consulte [“Definición de la configuración de actualización”](#) en la página 283). Si hay programada una comprobación de actualización, el proceso de actualización continúa.
- 2 `Updater` descarga e interpreta el archivo descriptor de actualización.
- 3 También se descarga el archivo de actualización de AIR.
- 4 `Updater` instala la versión actualizada de la aplicación.

El objeto `updater` distribuye eventos a la finalización de cada uno de estos pasos. En la versión de `ApplicationUpdater`, es posible cancelar los eventos que indiquen la correcta finalización de un paso en el proceso. Si cancela uno de estos eventos, se cancelará el siguiente paso del proceso. En la versión de `ApplicationUpdaterUI`, `updater` presenta un cuadro de diálogo que permite al usuario cancelar o continuar en cada paso del proceso.

Si cancela el evento, se pueden llamar a métodos del objeto `updater` para reanudar el proceso.

Conforme progresa la versión `ApplicationUpdater` de `updater` en el proceso de actualización, se registra su estado actual, en una propiedad `currentState`. Esta propiedad se establece en una cadena los siguientes posibles valores:

- "UNINITIALIZED": `updater` no se ha inicializado.
- "INITIALIZING": `updater` se está inicializando.
- "READY": `updater` se ha inicializado.
- "BEFORE_CHECKING": `updater` no ha comprobado aún el archivo descriptor de actualización.
- "CHECKING": `updater` está buscando un archivo descriptor de actualización.
- "AVAILABLE": el archivo descriptor de `updater` está disponible.
- "DOWNLOADING": `updater` está descargando el archivo de AIR.
- "DOWNLOADED": `updater` ha descargado el archivo de AIR.
- "INSTALLING": `updater` está instalando el archivo de AIR.
- "PENDING_INSTALLING": `updater` se ha inicializado y hay actualizaciones pendientes.

Algunos métodos del objeto `updater` solo se ejecutan si `updater` se encuentra en un determinado estado.

Inicialización del marco de actualización

Una vez definidas las propiedades de configuración, (consulte “[Ejemplo básico: Uso de la versión ApplicationUpdaterUI](#)” en la página 279), llame al método `initialize()` para inicializar la actualización:

```
appUpdater.initialize();
```

Este método realiza lo siguiente:

- Inicializa el marco de actualización, realizando una instalación silenciosa de forma sincrónica de todas las actualizaciones pendientes. Es necesario llamar a este método durante el inicio de la aplicación, ya que es posible que reinicie la aplicación cuando se llame.
- Compruebe si hay alguna actualización pendiente y la instala.
- Si se produce un error durante el proceso de actualización, borra el archivo de actualización y la información de la versión del área de almacenamiento de la aplicación.
- Si el intervalo de días de comprobación de actualización ha caducado, inicia el proceso de actualización. De lo contrario, reinicia el temporizador.

La llamada a este método puede provocar que el objeto `updater` distribuya los siguientes eventos:

- `UpdateEvent.INITIALIZED`: se distribuye cuando se completa la inicialización.
- `ErrorEvent.ERROR`: se distribuye cuando se produce un error durante la inicialización.

Al distribuir el evento `UpdateEvent.INITIALIZED`, el proceso de actualización se completa.

Cuando se llama al método `initialize()`, `updater` inicia el proceso de actualización y completa todos los pasos, en función de la configuración del intervalo de demora del temporizador. No obstante, también puede iniciar el proceso de actualización en cualquier momento llamando al método `checkNow()` del objeto `updater`:

```
appUpdater.checkNow();
```

Este método no realiza ninguna operación si el proceso de actualización ya se está ejecutando. De lo contrario, inicia el proceso de actualización.

El objeto `Updater` puede distribuir el siguiente evento como resultado de la llamada al método `checkNow()`:

- `UpdateEvent.CHECK_FOR_UPDATE` justo antes de que intente descargar el archivo descriptor de actualización.

Si se cancela el evento `checkForUpdate`, se puede llamar al método `checkForUpdate()` del objeto `Updater`. (Consulte la siguiente sección.) Si no cancela el evento, el proceso de actualización continúa comprobando el archivo descriptor de actualización.

Administración del proceso de actualización en la versión de `ApplicationUpdaterUI`

En la versión de `ApplicationUpdaterUI`, el usuario puede cancelar el proceso mediante los botones Cancelar de los cuadros de diálogo de la interfaz de usuario. Asimismo, es posible cancelar mediante programación el proceso de actualización llamando al método `cancelUpdate()` del objeto `ApplicationUpdaterUI`.

Se pueden establecer las propiedades del objeto `ApplicationUpdaterUI` o definir elementos en el archivo de configuración de actualización para especificar qué confirmaciones de cuadro de diálogo muestra `Updater`. Para obtener más información, consulte [“Definición de la configuración de actualización”](#) en la página 283.

Administración del proceso de actualización en la versión de `ApplicationUpdater`

Puede llamar al método `preventDefault()` de los objetos de evento distribuidos mediante el objeto `ApplicationUpdater` con el fin de cancelar pasos del proceso de actualización (consulte [“Proceso de actualización”](#) en la página 285). La cancelación del comportamiento predeterminado ofrece a la aplicación una oportunidad para mostrar un mensaje al usuario donde se le pregunta si se desea continuar.

En las siguientes secciones se describe cómo continuar con el proceso de actualización cuando se ha cancelado un paso del proceso.

Descarga e interpretación del archivo descriptor de actualización

El objeto `ApplicationUpdater` distribuye el evento `checkForUpdate` antes de que comience el proceso de actualización y justo antes de que `Updater` intente descargar el archivo descriptor de actualización. Si cancela el comportamiento predeterminado del evento `checkForUpdate`, `Updater` no descargará el archivo descriptor de actualización. Puede llamar al método `checkForUpdate()` para reanudar el proceso de actualización:

```
appUpdater.checkForUpdate();
```

Al llamar al método `checkForUpdate()`, `Updater` interpreta y descarga de forma asincrónica el archivo descriptor de actualización. Como resultado de la llamada al método `checkForUpdate()`, el objeto `Updater` puede distribuir los siguientes eventos:

- `StatusUpdateEvent.UPDATE_STATUS`: `Updater` ha descargado e interpretado el archivo descriptor de actualización correctamente. Este evento cuenta con las siguientes propiedades:
 - `available`: valor booleano. Se establece en `true` si existe una versión distinta disponible a la de la aplicación actual; de lo contrario, se establece en `false` (la versión es la misma).
 - `version`: cadena. La versión del archivo descriptor de la aplicación del archivo de actualización.
 - `details`: conjunto. Si no existen versiones localizadas de la descripción, este conjunto devuelve una cadena vacía ("") como primer elemento y la descripción como segundo elemento.

Si existen varias versiones de la descripción (en el archivo descriptor de actualización), el conjunto contiene varios subconjuntos. Cada conjunto dispone de dos elementos: el primero es un código de idioma (como, por ejemplo, "en") y el segundo es la descripción correspondiente (una cadena) para ese idioma. Consulte [“Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web”](#) en la página 281.

- `StatusUpdateErrorEvent.UPDATE_ERROR`: hubo un error y `updater` no pudo descargar ni interpretar el archivo descriptor de actualización.

Descarga del archivo de actualización de AIR

El objeto `ApplicationUpdater` distribuye el evento `updateStatus` una vez que `updater` descarga e interpreta correctamente el archivo descriptor de actualización. El comportamiento predeterminado es comenzar a descargar el archivo de actualización, si está disponible. Si cancela este comportamiento, puede llamar al método `downloadUpdate()` para que se reanude el proceso de actualización:

```
appUpdater.downloadUpdate();
```

Al llamar a este método, `updater` descarga de forma asincrónica la versión de actualización del archivo de AIR.

El método `downloadUpdate()` puede distribuir los siguientes eventos:

- `UpdateEvent.DOWNLOAD_START`: se ha establecido la conexión con el servidor. Al utilizar la biblioteca `ApplicationUpdaterUI`, este evento muestra un cuadro de diálogo con una barra de progreso para realizar un seguimiento del curso de la descarga.
- `ProgressEvent.PROGRESS`: se distribuye periódicamente conforme progresa la descarga del archivo.
- `DownloadErrorEvent.DOWNLOAD_ERROR`: se distribuye si se produce un error al conectar o descargar el archivo de actualización. También se distribuye para estados HTTP no válidos; por ejemplo, "404 - File not found" (No se encontró el archivo). Este evento cuenta con una propiedad `errorID`, un entero que define información de error adicional. Una propiedad adicional `subErrorID` puede incluir más información de error.
- `UpdateEvent.DOWNLOAD_COMPLETE`: `updater` ha descargado e interpretado correctamente el archivo descriptor de actualización. Si no cancela este evento, la versión de `ApplicationUpdater` procede a instalar la versión de actualización. En la versión de `ApplicationUpdaterUI`, se muestra un cuadro de diálogo al usuario en el que puede optar por continuar con el proceso.

Actualización de la aplicación

El objeto `ApplicationUpdater` distribuye el evento `downloadComplete` cuando finaliza la descarga del archivo de actualización. Si cancela el comportamiento predeterminado, puede llamar al método `installUpdate()` para que se reanude el proceso de actualización:

```
appUpdater.installUpdate(file);
```

Al llamar a este método, `updater` instala una versión de actualización del archivo de AIR. El método incluye un parámetro, `file`, que es un objeto `File` que hace referencia al archivo de AIR para usar como actualización.

El objeto `ApplicationUpdater` puede distribuir el evento `beforeInstall` como resultado de la llamada al método `installUpdate()`:

- `UpdateEvent.BEFORE_INSTALL`: se distribuye justo antes de la instalación de la actualización. En ocasiones, resulta útil evitar la instalación de la actualización en este momento, de modo que el usuario pueda completar el trabajo actual antes de que continúe la actualización. Con la llamada al método `preventDefault()` del objeto `Event` se pospone la instalación hasta el siguiente reinicio y no puede comenzar ningún proceso de actualización adicional. (Se incluyen actualizaciones que podrían aparecer con la llamada al método `checkNow()` o debido a las comprobaciones periódicas.)

Instalación desde un archivo arbitrario de AIR

Puede llamar al método `installFromAIRFile()` para instalar la versión de actualización desde un archivo de AIR en el equipo del usuario:

```
appUpdater.installFromAIRFile();
```

Este método hace que `updater` instale una versión de actualización de la aplicación desde el archivo de AIR.

El método `installFromAIRFile()` puede distribuir los siguientes eventos:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS`: se distribuye una vez que `ApplicationUpdater` ha validado correctamente el archivo enviado utilizando el método `installFromAIRFile()`. Este evento tiene las propiedades siguientes:
 - `available`: se establece en `true` si existe una versión distinta disponible a la de la aplicación actual; de lo contrario, se establece en `false` (las versiones son las mismas).
 - `version`: cadena que representa la nueva versión disponible.
 - `path`: representa la ruta nativa del archivo de actualización.

Puede cancelar este evento si la propiedad `available` del objeto `StatusFileUpdateEvent` se define como `true`. La cancelación del evento implica que no continúe la actualización. Llame al método `installUpdate()` para continuar con la actualización cancelada.

- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR`: hubo un error y `updater` no pudo instalar la aplicación de AIR.

Cancelación del proceso de actualización

Puede llamar al método `cancelUpdate()` para cancelar el proceso de actualización:

```
appUpdater.cancelUpdate();
```

Este método cancela todas las descargas pendientes, elimina todos los archivos descargados incompletos y reinicia el temporizador de comprobación periódica.

El método no realiza ninguna operación si el objeto `updater` se está inicializando.

Localización de la interfaz `ApplicationUpdaterUI`

La clase `ApplicationUpdaterUI` proporciona una interfaz de usuario predeterminada para el proceso de actualización. Esto incluye cuadros de diálogo que permiten al usuario iniciar y cancelar el proceso y llevar a cabo otras operaciones relacionadas.

El elemento `description` del archivo descriptor de actualización permite definir la descripción de la aplicación en varios idiomas. Emplee varios elementos `text` que definan atributos `lang`, `tal` y como se muestra a continuación:

```
<?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>
```

El marco de actualización utiliza la descripción que mejor se adapta a la cadena de localización del usuario final. Para obtener más información, consulte Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web.

Los desarrolladores de Flex puede añadir directamente un nuevo idioma al paquete "ApplicationUpdaterDialogs".

Los desarrolladores de JavaScript pueden llamar al método `addResources()` del objeto `updater`. Este método agrega de forma dinámica un nuevo paquete de recursos para un idioma. El paquete de recursos define cadenas localizadas para un idioma. Estas cadenas se utilizan en distintos campos de texto de cuadro de diálogo.

Los desarrolladores de JavaScript pueden emplear la propiedad `localeChain` de la clase `ApplicationUpdaterUI` para definir la cadena de configuraciones regionales empleada en la interfaz de usuario. Generalmente solo los desarrolladores de JavaScript (HTML) utilizan esta propiedad. Los desarrolladores de Flex pueden usar `ResourceManager` para administrar la cadena de configuraciones regionales.

Por ejemplo, el siguiente código de JavaScript define paquetes de recursos para rumano y húngaro:

```
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;
```

Para obtener más información, consulte la descripción del método `addResources()` de la clase `ApplicationUpdaterUI` en la referencia del lenguaje.

Capítulo 18: Visualización de código fuente

Del mismo modo que un usuario puede ver el código fuente de una página HTML en un navegador web, los usuarios pueden ver el código fuente de una aplicación de AIR basada en HTML. El SDK de Adobe® AIR® incluye un archivo AIRSourceViewer.js JavaScript que se puede utilizar en la aplicación para mostrar fácilmente el código fuente a los usuarios finales.

Carga, configuración y apertura del visor de código fuente

El código del visor de código fuente se incluye en un archivo JavaScript, AIRSourceViewer.js, que se incluye en el directorio frameworks del SDK de AIR. Para utilizar el visor de código fuente en su aplicación, copie AIRSourceViewer.js en el directorio project de la aplicación y cargue el archivo mediante una etiqueta de script en el archivo HTML principal en su aplicación:

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

El archivo AIRSourceViewer.js define una clase, SourceViewer, a la que se puede acceder desde el código JavaScript llamando a `air.SourceViewer`.

La clase SourceViewer define tres métodos: `getDefault()`, `setup()` y `viewSource()`.

Método	Descripción
<code>getDefault()</code>	Método estático. Devuelve una instancia de SourceViewer, que se puede utilizar para llamar a los otros métodos.
<code>setup()</code>	Aplica las opciones de configuración al visor de código fuente. Para obtener más información, consulte "Configuración del visor de código fuente" en la página 291
<code>viewSource()</code>	Abre una nueva ventana en la que el usuario puede examinar y abrir archivos de origen de la aplicación host.

Nota: el código que utiliza el visor de código fuente debe estar en el entorno limitado de seguridad de la aplicación (en un archivo del directorio de la aplicación).

Por ejemplo, el siguiente código JavaScript crea una instancia de un objeto Source Viewer y abre la ventana de Source Viewer donde se incluyen todos los archivos de origen:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Configuración del visor de código fuente

El método `config()` aplica la configuración dada en el visor de código fuente. Este método adopta un parámetro: `configObject`. El objeto `configObject` dispone de propiedades que definen las opciones de configuración del visor de código fuente. Entre las propiedades se incluyen `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` y `typesToAdd`.

default

Cadena que especifica la ruta relativa al archivo inicial que se mostrará en el visor de código fuente.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente con el archivo index.html como archivo inicial mostrado:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

Conjunto de cadenas que especifican los archivos o directorios que se excluirán de la lista del visor de código fuente. Las rutas son relativas al directorio de la aplicación. Los caracteres comodines no se admiten.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente donde se incluyen todos los archivos de origen, excepto AIRSourceViewer.js y los archivos de los subdirectorios Images y Sounds:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

Conjunto que incluye dos números, que especifican las coordenadas iniciales x e y de la ventana del visor de código fuente.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente con las coordenadas de la pantalla [40, 60] (X = 40, Y = 60):

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

Valor booleano que especifica si el visor de código fuente debe ser una ventana modal (true) o no modal (false). De forma predeterminada, la ventana del visor de código fuente es modal.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente de modo que el usuario pueda interactuar tanto con la ventana del visor como con cualquier ventana de la aplicación:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

Conjunto de cadenas que especifica los tipos de archivo que se incluirán en la lista del visor de código fuente, además de los tipos predeterminados incluidos.

De forma predeterminada, la lista Visor de origen incluye los siguientes tipos de archivo:

- Archivos de texto: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG.
- Archivo de imagen: JPG, JPEG, PNG, GIF.

Si no se especifica ningún valor, se incluyen todos los tipos predeterminados (excepto los especificados en la propiedad `typesToExclude`).

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente con los archivos VCF y VCARD:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToAdd = ["text.vcf", "text.vcard"];
viewer.viewSource(configObj);
```

Para cada tipo de archivo que se incluya, se debe especificar "text" (para los tipos de archivo de texto) o "image" (para los tipos de archivo de imagen).

typesToExclude

Conjunto de cadenas que especifican los tipos de archivo que se excluirán del visor de código fuente.

De forma predeterminada, la lista Visor de origen incluye los siguientes tipos de archivo:

- Archivos de texto: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG.
- Archivo de imagen: JPG, JPEG, PNG, GIF.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente sin incluir los archivos GIF y XML:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

Para cada tipo de archivo que se incluya, se debe especificar "text" (para los tipos de archivo de texto) o "image" (para los tipos de archivo de imagen).

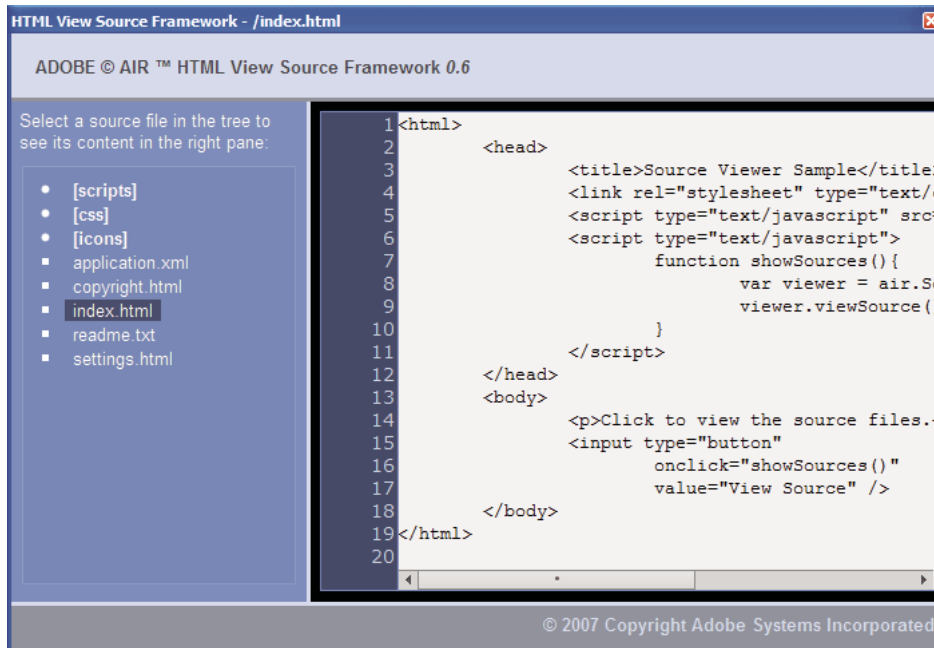
Apertura del visor de código fuente

Se debe incluir un elemento de interfaz de usuario como, por ejemplo, un vínculo, botón o comando de menú, que llame al código del visor de código fuente cuando el usuario lo seleccione. Por ejemplo, la siguiente aplicación sencilla abre el visor de código fuente cuando el usuario hace clic en un vínculo:

```
<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>
```

Interfaz de usuario del visor de código fuente

Si la aplicación llama al método `viewSource()` de un objeto `SourceViewer`, la aplicación de AIR abre una ventana del visor de código fuente. La ventana incluye una lista de directorios y archivos de origen (en la parte izquierda) y un área de visualización que muestra el código fuente del archivo seleccionado (en la parte derecha):



Los directorios se incluyen entre corchetes. El usuario puede hacer clic en una llave para expandir o contraer la lista de un directorio.

El visor de código fuente puede mostrar el código de los archivos de texto con extensiones reconocidas (por ejemplo, HTML, HTML, JS, TXT, XML, etc.) o de archivos de imagen con extensiones de imagen reconocidas (JPG, JPEG, PNG y GIF). Si el usuario selecciona un archivo que no dispone de una extensión reconocida, aparece el mensaje de error “Cannot retrieve text content from this filetype” (No se puede recuperar el contenido de texto de este tipo de archivo).

Todos los archivos de origen excluidos mediante el método `setup()` no se incluyen en la lista (consulte “[Carga, configuración y apertura del visor de código fuente](#)” en la página 291).

Capítulo 19: Depuración con el introspector HTML de AIR

EL SDK de Adobe® AIR® incluye un archivo de JavaScript AIRIntrospector.js que se puede incluir en la aplicación para ayudar a depurar aplicaciones basadas en HTML.

Introspector de AIR

El introspector de aplicaciones HTML/JavaScript de Adobe AIR (denominado AIR HTML Introspector) ofrece funciones útiles para ayudar en el desarrollo y depuración de aplicaciones basadas en HTML:

- Incluye una herramienta introspectora que permite señalar un elemento de la interfaz de usuario en la aplicación y ver su marcado y propiedades DOM.
- Incluye una consola para enviar referencias de objetos para introspección y es posible ajustar valores de propiedad y ejecutar código JavaScript. Asimismo, se pueden serializar objetos en la consola, lo que supone limitaciones para editar los datos. También se puede copiar y guardar texto desde la consola.
- Incluye una vista de árbol para las funciones y propiedades DOM.
- Permite editar atributos y nodos de texto para elementos DOM.
- Realiza listados de vínculos, estilos CSS, imágenes y archivos de JavaScript cargados en la aplicación.
- Permite visualizar el código fuente HTML inicial y el código fuente de marcado para la interfaz de usuario.
- Permite acceder a archivos en el directorio de la aplicación. (Esta función solo está disponible en la consola AIR HTML Introspector abierta para el entorno limitado de la aplicación. No está disponible para las consolas abiertas para el contenido del entorno limitado que no pertenece a la aplicación.)
- Incluye un visor para objetos XMLHttpRequest y sus propiedades, entre las que se incluyen `responseText` y `responseXML` (si están disponibles).
- Puede realizar búsquedas por texto coincidente en los archivos y el código fuente.

Carga del código del introspector de AIR

El código del introspector de AIR se incluye en un archivo de JavaScript, AIRIntrospector.js, incluido en el directorio frameworks del SDK de AIR. Para utilizar el introspector de AIR en la aplicación, copie AIRIntrospector.js en el directorio project de la aplicación y cargue el archivo mediante una etiqueta de script en el archivo HTML principal de la aplicación:

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

Asimismo, incluya en archivo en todos los archivos HTML que se correspondan con diferentes ventanas nativas de la aplicación.

Importante: incluya el archivo AIRIntrospector.js únicamente al desarrollar y depurar la aplicación. Elimínelo de la aplicación de AIR empaquetada que se distribuya.

El archivo AIRIntrospector.js define una clase, Console, a la que se puede acceder desde el código JavaScript, llamando a `air.Introspector.Console`.

Nota: el código que utilice el introspector de AIR debe estar en el entorno limitado de seguridad de la aplicación (en un archivo del directorio de la aplicación).

Inspección de un objeto en la ficha Console (Consola)

La clase Console define cinco métodos: `log()`, `warn()`, `info()`, `error()` y `dump()`.

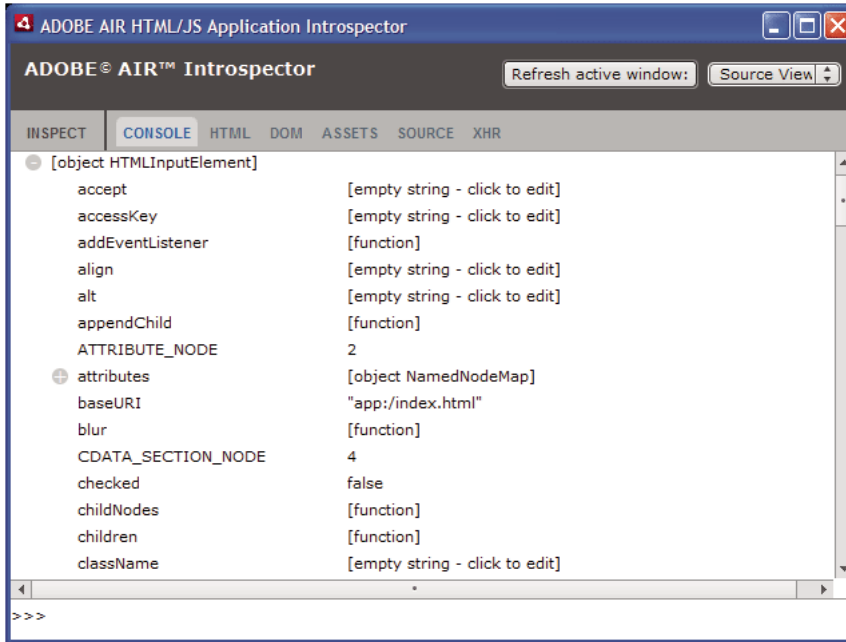
Los métodos `log()`, `warn()`, `info()`, and `error()` permiten enviar un objeto a la ficha Console (Consola). El método más básico es `log()`. El siguiente código envía un objeto simple, representado mediante la variable `test`, a la ficha Console (Consola):

```
var test = "hello";
air.Introspector.Console.log(test);
```

No obstante, resulta más útil enviar un objeto complejo a la ficha. Por ejemplo, la siguiente página HTML incluye un botón (`btn1`) que llama a una función que, a su vez, envía el propio objeto del botón a la ficha Console (Consola):

```
<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>
```

Cuando se hace clic en el botón, la ficha Console (Consola) muestra el objeto btn1 y se puede expandir la vista de árbol del objeto para inspeccionar sus propiedades:



Se puede editar una propiedad del objeto haciendo clic en la lista situada a la derecha del nombre de la propiedad y modificando el listado de texto.

Los métodos `info()`, `error()` y `warn()` son similares al método `log()`. Sin embargo, cuando se llama a estos métodos, la consola muestra un icono al principio de la línea:

Método	Icono
<code>info()</code>	
<code>error()</code>	
<code>warn()</code>	

Los métodos `log()`, `warn()`, `info()` y `error()` envían una referencia solo a un objeto real, por lo que las propiedades disponibles son las que se muestran en el momento de la visualización. Si desea serializar el objeto real, utilice el método `dump()`. El método cuenta con dos parámetros:

Parámetro	Descripción
<code>dumpObject</code>	Objeto que se va a serializar.
<code>levels</code>	Número máximo de niveles que se examinarán en el árbol del objeto (además del nivel de raíz). El valor predeterminado es 1 (lo que significa que se muestra un nivel superior al nivel de raíz del árbol). Este parámetro es opcional.

Al llamar al método `dump()` se serializa un objeto antes de enviarlo a la ficha Console (Consola), por lo que no se pueden editar las propiedades de los objetos. Por ejemplo, considérese el fragmento de código siguiente:

```
var testObject = new Object();  
testObject.foo = "foo";  
testObject.bar = 234;  
air.Introspector.Console.dump(testObject);
```

Cuando se ejecuta este código, la consola muestra el objeto `testObject` y sus propiedades, pero no se pueden editar los valores de las propiedades en la consola.

Configuración del introspector de AIR

La consola se puede configurar definiendo las propiedades de la variable global `AIRIntrospectorConfig`. Por ejemplo, el siguiente código JavaScript configura el introspector de AIR para que se ajusten las columnas a 100 caracteres:

```
var AIRIntrospectorConfig = new Object();  
AIRIntrospectorConfig.wrapColumns = 100;
```

Asegúrese de establecer las propiedades de la variable `AIRIntrospectorConfig` antes de cargar el archivo `AIRIntrospector.js` (mediante una etiqueta `script`).

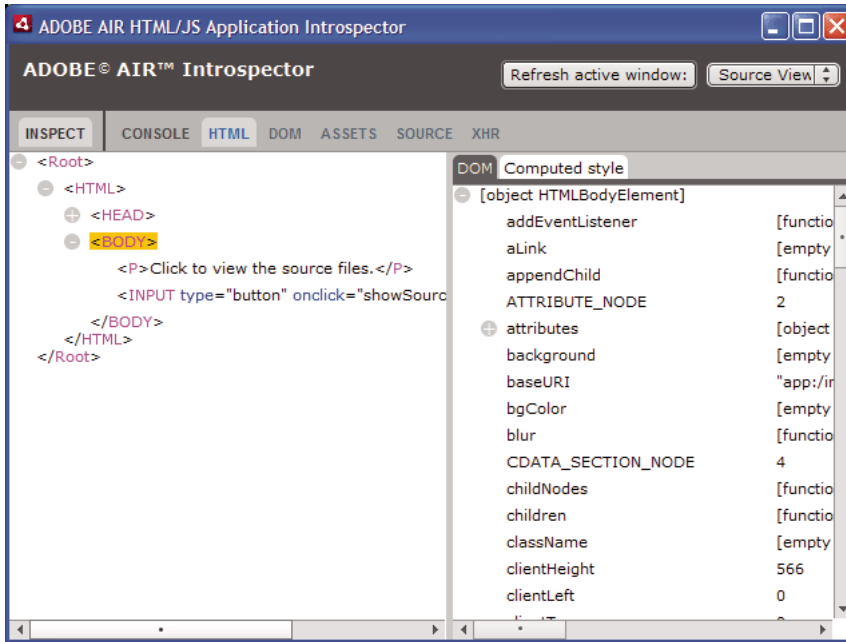
Existe ocho propiedades de la variable `AIRIntrospectorConfig`:

Propiedad	Valor predeterminado	Descripción
<code>closeIntrospectorOnExit</code>	<code>true</code>	Establezca la ventana del inspector para que se cierre cuando se cierren todas las demás ventanas de la aplicación.
<code>debuggerKey</code>	123 (tecla F12)	Código de tecla del método abreviado de teclado para mostrar y ocultar la ventana del introspector de AIR.
<code>debugRuntimeObjects</code>	<code>true</code>	Establece el introspector para que amplíe los objetos del motor de ejecución definidos en JavaScript.
<code>flashTabLabels</code>	<code>true</code>	Establece las fichas Console (Consola) y XMLHttpRequest en flash, indicando el momento en que se producen cambios (por ejemplo, cuando el texto se registre en estas fichas).
<code>introspectorKey</code>	122 (tecla F11)	Código de tecla del método abreviado de teclado para abrir el panel Inspect (Inspeccionar).
<code>showTimestamp</code>	<code>true</code>	Establece la ficha Console (Consola) para que las marcas de hora aparezcan al principio de todas las líneas.
<code>showSender</code>	<code>true</code>	Establece la ficha Console (Consola) para que muestre información sobre el objeto que envía el mensaje al principio de todas las líneas.
<code>wrapColumns</code>	2000	Número de columnas al que se ajustan los archivos de origen.

Interfaz del introspector de AIR

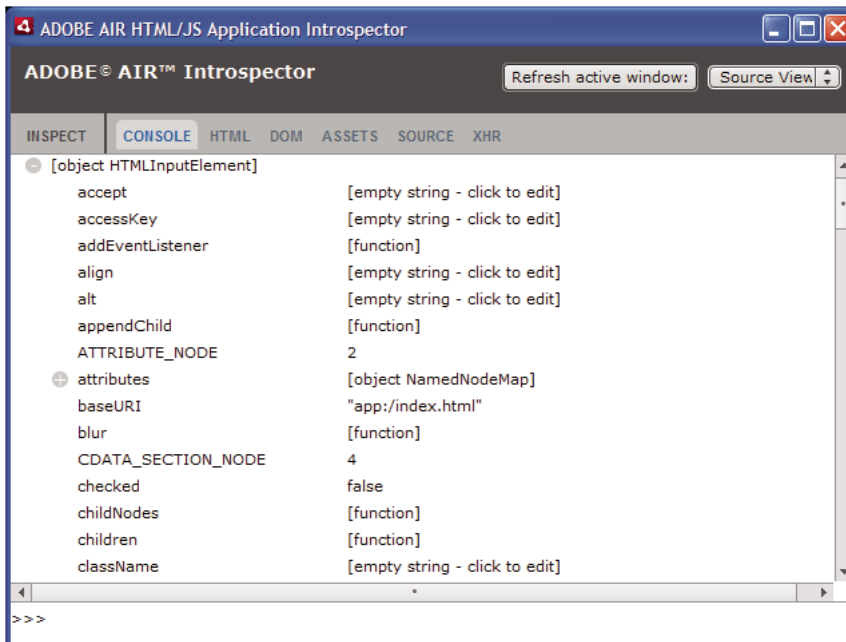
Para abrir la ventana del introspector de AIR al depurar la aplicación, pulse la tecla F12 o llame a uno de los métodos de la clase `Console` [consulte [“Inspección de un objeto en la ficha Console \(Consola\)”](#) en la página 296]. Puede configurar la tecla directa para que sea una tecla distinta a F12; consulte [“Configuración del introspector de AIR”](#) en la página 298.

La ventana del introspector de AIR dispone de seis fichas: Console (Consola), HTML, DOM, Assets (Componentes), Source (Código fuente) y XHR, tal y como se muestra en la siguiente ilustración:



Ficha Console (Consola)

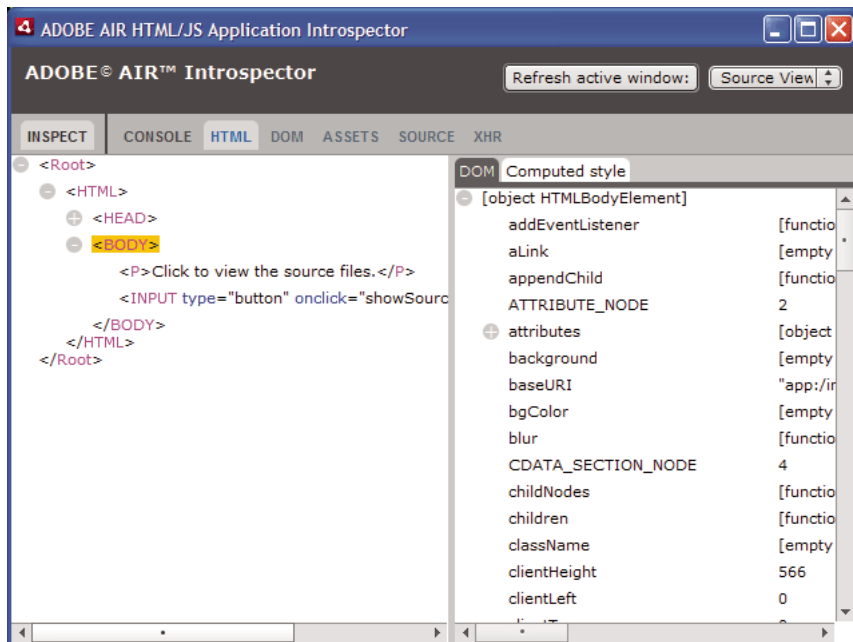
La ficha Console (Consola) muestra los valores de las propiedades transmitidas como parámetros a uno de los métodos de la clase `air.Introspector.Console`. Para obtener más información, consulte [“Inspección de un objeto en la ficha Console \(Consola\)”](#) en la página 296.



- Para borrar la consola, haga clic con el botón derecho del ratón en el texto y seleccione Clear Console (Borrar consola).
- Para guardar texto en la ficha Console (Consola) en un archivo, haga clic con el botón derecho del ratón en la ficha de consola y seleccione Save Console To File (Guardar consola en archivo).
- Para guardar texto en la ficha Console (Consola) en el portapapeles, haga clic con el botón derecho del ratón en la ficha de consola y seleccione Save Console To Clipboard (Guardar consola en portapapeles). Para copiar solo texto seleccionado en el portapapeles, haga clic con el botón derecho del ratón en el texto y seleccione Copy (Copiar).
- Para guardar texto de la clase Console en un archivo, haga clic con el botón derecho del ratón en la ficha de consola y seleccione Save Console To File (Guardar consola en archivo).
- Para buscar texto coincidente mostrado en la ficha, pulse Ctrl+F en Windows o Comando+F en Mac OS. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha HTML

La ficha HTML permite ver todo el DOM de HTML en una estructura de árbol. Haga clic en un elemento para ver sus propiedades en la parte derecha de la ficha. Haga clic en los iconos + y - para expandir y contraer un nodo en el árbol.



Se puede editar cualquier atributo o elemento de texto en la ficha HTML y el valor editado se refleja en la aplicación.

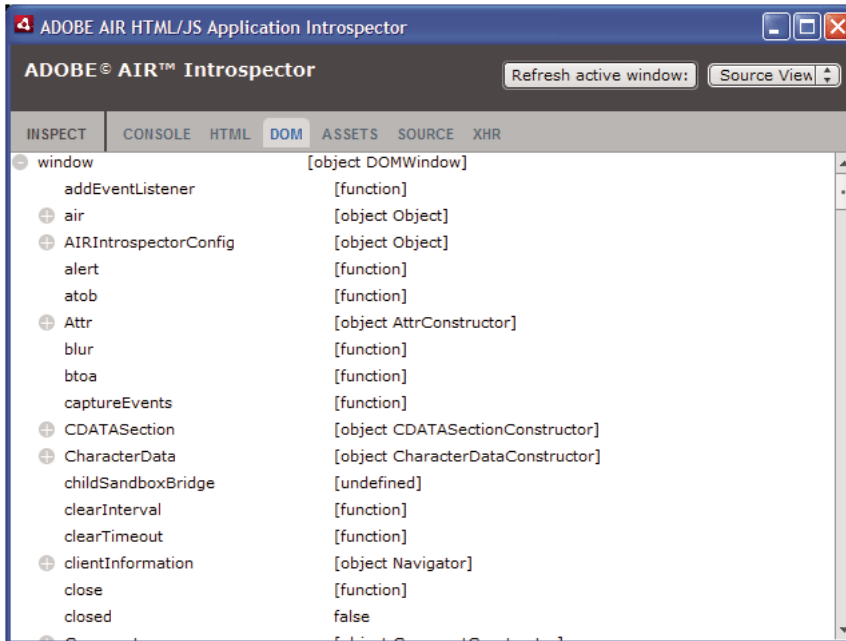
Haga clic en el botón Inspect (Inspeccionar) (situado a la izquierda de la lista de fichas de la ventana del introspector de AIR). Se puede hacer clic en cualquier elemento de la página HTML de la ventana principal y el objeto DOM asociado se muestra en la ficha HTML. Si la ventana principal está seleccionada, también se puede pulsar el método abreviado de teclado para activar y desactivar el botón Inspect (Inspeccionar). El método abreviado de teclado es F11 de forma predeterminada. Es posible configurar el método abreviado de teclado para sea una tecla distinta a F11; consulte "[Configuración del introspector de AIR](#)" en la página 298.

Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha HTML.

Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha DOM

La ficha DOM muestra los objetos de ventana en una estructura de árbol. Se puede editar cualquier propiedad string y numeric y el valor editado se refleja en la aplicación.

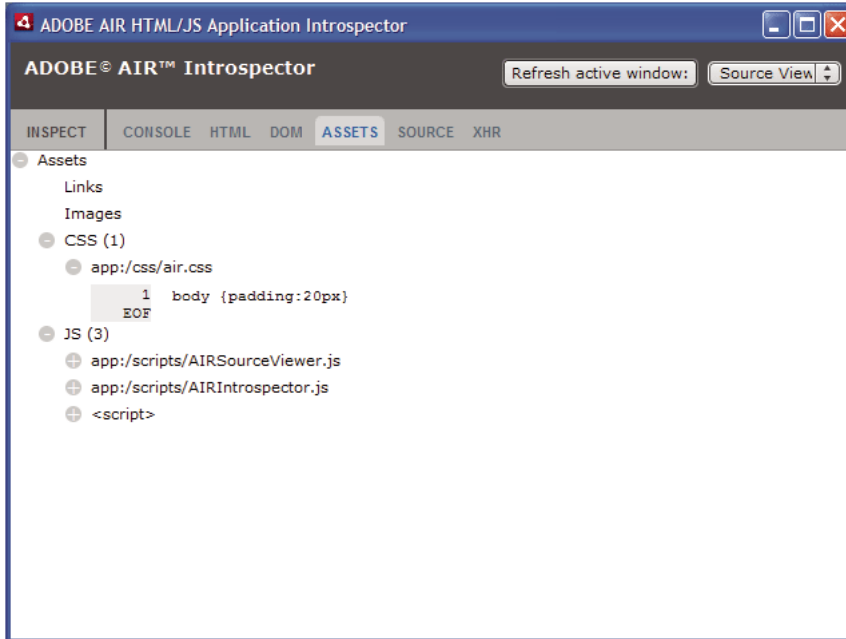


Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha DOM.

Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha Assets (Componentes)

La ficha Assets (Componentes) permite comprobar vínculos, imágenes, CSS y archivos de JavaScript cargados en la ventana nativa. Al expandir uno de estos nodos se muestra el contenido del archivo o la imagen real utilizada.



Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha Assets (Componentes).

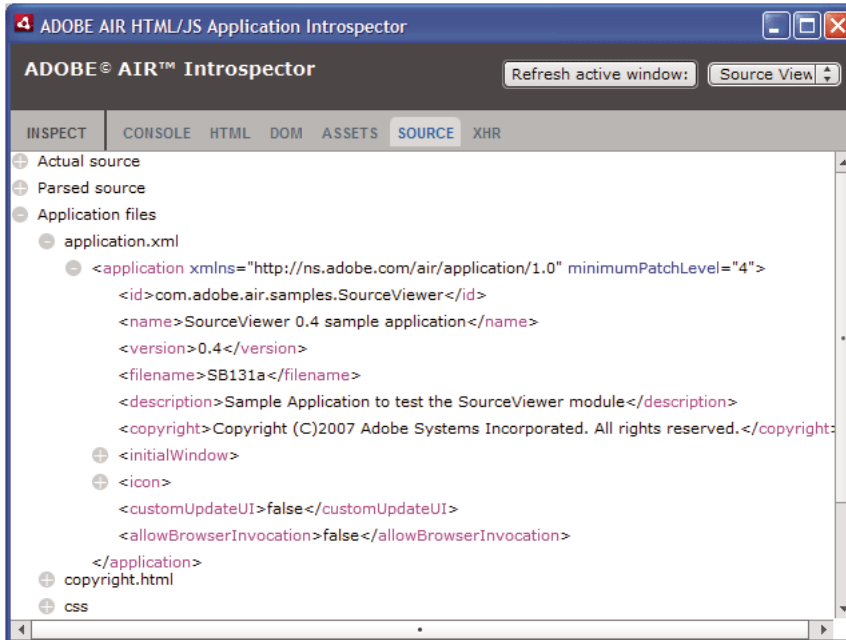
Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha Source (Código fuente)

La ficha Source (Código fuente) incluye tres secciones:

- Código fuente real: muestra el código fuente HTML de la página cargado como contenido raíz cuando se inició la aplicación.
- Código fuente analizado: muestra el marcado actual que conforma la IU de la aplicación, que puede ser distinto al código fuente real, ya que la aplicación genera código de marcado sobre la marcha utilizando técnicas de Ajax.

- Archivos de la aplicación: incluye los archivos en el directorio de la aplicación. Esta lista solo está disponible para el introspector de AIR cuando se inicia desde el contenido del entorno limitado de seguridad de la aplicación. En esta sección se puede ver el contenido de archivos de texto o ver imágenes.

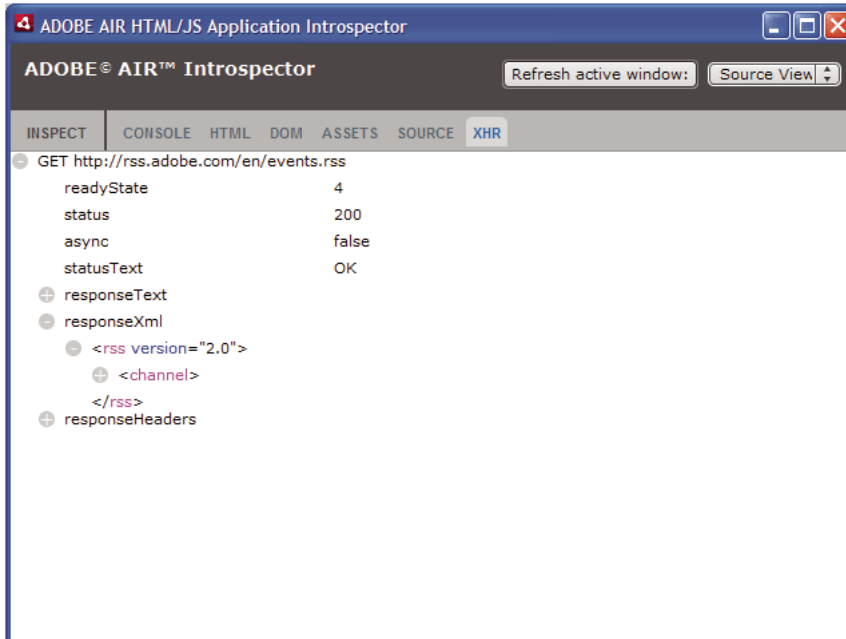


Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha Source (Código fuente).

Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha XHR

La ficha XHR intercepta toda la comunicación XMLHttpRequest en la aplicación y registra la información. Esto permite ver las propiedades XMLHttpRequest, entre las que se incluyen `responseText` y `responseXML` (cuando están disponibles) en una vista de árbol.



Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Utilización del introspector de AIR con contenido en un entorno limitado ajeno a la aplicación

Se puede cargar contenido de un directorio de la aplicación en un `iframe` o `frame` que se asigne a un entorno limitado ajeno a la aplicación. (Consulte [Seguridad HTML en Adobe AIR](#) para desarrolladores de ActionScript o [Seguridad HTML en Adobe AIR](#) para desarrolladores de HTML). El introspector de AIR se puede utilizar con este contenido, pero se deben tener en cuenta las siguientes reglas:

- El archivo `AIRIntrospector.js` se debe incluir en el contenido del entorno limitado de la aplicación y del entorno limitado ajeno a la aplicación (`iframe`).
- No sobrescriba la propiedad `parentSandboxBridge`; el código del introspector de AIR usa esta propiedad. Añada propiedades según sea necesario. Por lo tanto, en lugar de escribir lo siguiente:

```
parentSandboxBridge = mytrace: function(str) {runtime.trace(str)} ;
```

Utilice la siguiente sintaxis:

```
parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
```

- Desde el contenido del entorno limitado ajeno a la aplicación no es posible abrir el introspector de AIR pulsando la tecla F12 ni llamando a uno de los métodos de la clase `air.Introspector.Console`. La ventana del introspector solo se puede abrir haciendo clic en el botón Open Introspector (Abrir introspector). El botón se añade de forma predeterminada a la esquina superior derecha de `iframe` o `frame`. (Debido a las restricciones de seguridad aplicadas al contenido del entorno limitado ajeno a la aplicación, una ventana nueva solo se puede abrir como resultado de un gesto del usuario como, por ejemplo, hacer clic en un botón.)
- Es posible abrir ventanas del introspector de AIR independientes para el entorno limitado de la aplicación y el entorno limitado ajeno a la aplicación. Los dos entornos se pueden diferenciar utilizando el título que aparece en las ventanas del introspector de AIR.
- La ficha Source (Código fuente) no muestra archivos de la aplicación cuando el introspector de AIR se ejecuta desde un entorno limitado ajeno a la aplicación.
- El introspector de AIR solo puede consultar código en el entorno limitado desde el que se ha abierto.

Capítulo 20: Localización de aplicaciones de AIR

Adobe AIR 1.1 y posterior

Adobe® AIR® es compatible con el uso de diversos idiomas.

Para obtener información general sobre la localización de contenido en ActionScript 3.0 y la arquitectura de Flex, consulte "Localización de aplicaciones" en la guía del desarrollador de Adobe ActionScript 3.0.

Idiomas admitidos en AIR

La compatibilidad con la localización para las aplicaciones de AIR en los siguientes idiomas se introdujo en AIR 1.1:

- Chino simplificado
- Chino tradicional
- Francés
- Alemán
- Italiano
- Japonés
- Coreano
- Portugués brasileño
- Ruso
- Español

En AIR 1.5, se añadieron los siguientes idiomas:

- Checo
- Holandés
- Polaco
- Sueco
- Turco

Más temas de ayuda

[Building multilingual Flex applications on Adobe AIR \(Creación de aplicaciones de Flex multilingües en Adobe AIR\)](#)

[Building a multilingual HTML-based application \(Creación de una aplicación multilingüe basada en HTML\)](#)

Localización del nombre y la descripción en el instalador de aplicaciones de AIR

Adobe AIR 1.1 y posterior

Se pueden especificar varios idiomas para los elementos `name` y `description` en el archivo descriptor de la aplicación. En el ejemplo siguiente se especifica el nombre de la aplicación en tres idiomas (inglés, francés y alemán):

```
<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>
```

El atributo `xml:lang` de cada elemento de texto especifica un código de idioma, tal y como se define en RFC4646 (<http://www.ietf.org/rfc/rfc4646.txt>).

El elemento "name" solo define el nombre de la aplicación que presenta el instalador de aplicaciones de AIR. El instalador de aplicaciones de AIR utiliza el valor localizado que mejor se corresponde con el idioma de la interfaz de usuario definido en la configuración del sistema operativo.

Se pueden especificar varias versiones de idiomas del elemento `description` en el archivo descriptor de la aplicación. Este elemento define el texto descriptivo que presenta el instalador de aplicaciones de AIR.

Estas opciones solo se aplican a los idiomas que se ofrecen en el instalador de aplicaciones de AIR. No definen las configuraciones regionales disponibles para la aplicación instalada y en funcionamiento. Las aplicaciones de AIR pueden ofrecer interfaces de usuario compatibles con varios idiomas, incluidos los del instalador de aplicaciones de AIR y otros adicionales.

Para obtener más información, consulte “[Elementos del descriptor de la aplicación de AIR](#)” en la página 218.

Más temas de ayuda

[Building multilingual Flex applications on Adobe AIR](#) (Creación de aplicaciones de Flex multilingües en Adobe AIR)

[Building a multilingual HTML-based application](#) (Creación de una aplicación multilingüe basada en HTML)

Localización de contenido HTML con la arquitectura de localización de HTML de AIR

Adobe AIR 1.1 y posterior

El SDK de AIR 1.1 incluye una arquitectura de localización de HTML. El archivo JavaScript AIRLocalizer.js define la arquitectura. El archivo AIRLocalizer.js se encuentra en el directorio `frameworks` del SDK de AIR. Este archivo incluye la clase `air.Localizer`, que ofrece funciones de utilidad para la creación de aplicaciones compatibles con varias versiones localizadas.

Carga del código de la arquitectura de localización de HTML de AIR

Para utilizar la arquitectura de localización, copie el archivo AIRLocalizer.js en su proyecto. Inclúyalo en el archivo HTML principal de la aplicación con una etiqueta de `script`:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

El código JavaScript que le sigue llamar al objeto `air.Localizer.localizer`:

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

El objeto `air.Localizer.localizer` es un objeto de instancia única que define métodos y propiedades para utilizar y gestionar los recursos localizados. La clase `Localizer` incluye los métodos siguientes:

Método	Descripción
<code>getFile()</code>	Obtiene el texto de un paquete de recursos especificado para una configuración regional especificada. Consulte “Obtención de recursos para una configuración regional específica” en la página 315.
<code>getLocaleChain()</code>	Devuelve los idiomas de la cadena de configuraciones regionales. Consulte “Definición de la cadena de configuraciones regionales” en la página 315.
<code>getResourceBundle()</code>	Devuelve las claves del paquete y los valores correspondientes como un objeto. Consulte “Obtención de recursos para una configuración regional específica” en la página 315.
<code>getString()</code>	Obtiene la cadena de caracteres definida para un recurso. Consulte “Obtención de recursos para una configuración regional específica” en la página 315.
<code>setBundlesDirectory()</code>	Configura la ubicación del directorio de paquetes (bundles). Consulte “Personalización de las opciones de AIR HTML Localizer” en la página 314.
<code>setLocalAttributePrefix()</code>	Define el prefijo para los atributos de localización que se utilizan en los elementos DOM de HTML. Consulte “Personalización de las opciones de AIR HTML Localizer” en la página 314.
<code>setLocaleChain()</code>	Define el orden de los idiomas en la cadena de configuraciones regionales. Consulte “Definición de la cadena de configuraciones regionales” en la página 315.
<code>sortLanguagesByPreference()</code>	Ordena las configuraciones regionales en la cadena de configuraciones regionales en función del orden en que se encuentran en la configuración del sistema operativo. Consulte “Definición de la cadena de configuraciones regionales” en la página 315.
<code>update()</code>	Actualiza el DOM de HTML (o un elemento DOM) con cadenas de caracteres localizadas procedentes de la actual cadena de configuraciones regionales. Para ver una discusión sobre las cadenas de configuraciones regionales, consulte “Gestión de cadenas de configuraciones regionales” en la página 311. Para obtener más información sobre el método <code>update()</code> , consulte “Actualización de elementos DOM para utilizar la configuración regional actual” en la página 312.

La clase `Localizer` incluye las siguientes propiedades estáticas:

Propiedad	Descripción
<code>localizer</code>	Devuelve una referencia al objeto <code>Localizer</code> de instancia única para la aplicación.
<code>ultimateFallbackLocale</code>	La configuración regional que se utiliza cuando la aplicación no admite ninguna de las preferencias del usuario. Consulte “Definición de la cadena de configuraciones regionales” en la página 315.

Especificación de los idiomas admitidos

Utilice el elemento `<supportedLanguages>` del archivo descriptor de la aplicación para identificar los idiomas admitidos por la aplicación. Este elemento solamente se utiliza en iOS, el motor de ejecución captador de Mac y las aplicaciones de Android; se omite en todos los demás tipos de aplicaciones.

Si no especifica el elemento `<supportedLanguages>`, el empaquetador lleva a cabo de forma predeterminada las siguientes acciones en función del tipo de aplicación:

- iOS: todos los idiomas admitidos por el motor de ejecución de AIR se enumeran en el App Store de iOS como idiomas admitidos de la aplicación.
- Motor de ejecución captador de Mac: la aplicación empaquetada con el paquete captador no contiene información de localización.
- Android: el paquete de la aplicación tiene los recursos para todos los idiomas admitidos por el motor de ejecución de AIR.

Para obtener más información, consulte [“supportedLanguages”](#) en la página 249.

Definición de paquetes de recursos

La arquitectura de localización de HTML lee las versiones localizadas de cadenas de caracteres en los archivos de *localización*. Un archivo de localización es una colección de valores basados en clave y serializados en un archivo de texto. A veces se le conoce por la palabra *paquete*.

Cree un subdirectorio del directorio del proyecto de la aplicación llamado “configregional”. (También puede utilizar otro nombre; consulte [“Personalización de las opciones de AIR HTML Localizer”](#) en la página 314.) Este directorio incluirá los archivos de localización. Este directorio se conoce como el *directorio de paquetes*.

Para cada configuración regional que admita la aplicación, cree un subdirectorio del directorio de paquetes. Dé a cada subdirectorio un nombre que corresponda al código de la configuración regional. Por ejemplo: llame al directorio francés “fr” y al directorio inglés “en”. Para definir una configuración regional con códigos de idioma y de país se puede utilizar guion bajo (_). Por ejemplo, el directorio de inglés estadounidense se llamaría “en_us”. (También se puede utilizar un guion normal en lugar de un guion bajo: “en-us”. La arquitectura de localización de HTML admite ambas formas).

No hay límite de la cantidad de archivos de recursos que se pueden añadir a un subdirectorio de configuraciones regionales. Se suele crear un archivo de localización para cada idioma (y colocar el archivo en el directorio de ese idioma). La arquitectura de localización de HTML incluye un método `getFile()` que permite leer el contenido de un archivo (consulte [“Obtención de recursos para una configuración regional específica”](#) en la página 315).

Los archivos que tienen la extensión de archivo `.properties` se denominan “archivos de propiedades de localización”. Se pueden utilizar para definir pares clave-valor para una configuración regional. Un archivo de propiedades define un valor de cadena en cada línea. En el siguiente ejemplo se define el valor de cadena “Hello in English.” para una clave denominada `greeting`:

```
greeting=Hello in English.
```

Un archivo de propiedades que contiene el texto siguiente define seis pares clave-valor:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Este ejemplo muestra una versión en inglés del archivo de propiedades, que se guarda en el directorio `en`.

La versión en francés de este archivo de propiedades se coloca en el directorio `fr`:

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Se pueden definir varios archivos de recursos para distintos tipos de información. Por ejemplo: un archivo `legal.properties` podría contener un texto jurídico estándar (como un aviso de copyright). Quizá se desee utilizar ese recurso en varias aplicaciones. Asimismo, se pueden definir archivos separados que definen el contenido localizado para distintas partes de la interfaz de usuario.

Utilice para estos archivos la codificación UTF-8 para mayor compatibilidad con distintos idiomas.

Gestión de cadenas de configuraciones regionales

Cuando la aplicación carga el archivo `AIRLocalizer.js`, la misma examina las configuraciones regionales que tiene definidas. Estas configuraciones regionales corresponden a los subdirectorios del directorio de paquetes (consulte [“Definición de paquetes de recursos”](#) en la página 310). Esta lista de configuraciones regionales disponibles se denomina la *cadena de configuraciones regionales*. El archivo `AIRLocalizer.js` ordena automáticamente la cadena de configuraciones regionales en función del orden definido en la configuración del sistema operativo. (La propiedad `Capabilities.languages` enumera los idiomas de la interfaz de usuario del sistema operativo en orden de preferencia).

De este modo, si una aplicación define recursos para las configuraciones regionales "es", "es_ES" y "es_UY", la arquitectura AIR HTML Localizer ordena en consecuencia la cadena de configuraciones regionales. Cuando se inicia una aplicación en un sistema que da aviso de "es" como configuración regional primaria, la cadena de configuraciones regionales se ordena en la secuencia ["es", "es_ES" y "es_UY"]. En este caso la aplicación busca recursos primero en el paquete "es" y después en el paquete "es_ES".

Sin embargo, si el sistema da aviso de "es_ES" como configuración regional primaria, la clasificación es ["es_ES", "es", "es_UY"]. En este caso la aplicación busca recursos primero en el paquete "es_ES" y después en el paquete "es".

La aplicación define automáticamente la primera configuración regional de la cadena como la configuración regional predeterminada a utilizarse. Puede pedir al usuario que seleccione una configuración regional la primera vez que ejecuta la aplicación. Puede optar por guardar la selección en un archivo de preferencias y en adelante utilizar esa configuración regional cada vez que se inicie la aplicación.

La aplicación puede utilizar cadenas de caracteres de recurso en cualquier configuración regional de la cadena de configuraciones regionales. Si una configuración regional específica no define una cadena de caracteres de recurso, la aplicación utiliza la siguiente cadena de caracteres de recurso que coincida para otras configuraciones regionales definidas en la cadena de dichas configuraciones.

Se puede personalizar la cadena de configuraciones regionales llamando al método `setLocaleChain()` del objeto `Localizer`. Consulte [“Definición de la cadena de configuraciones regionales”](#) en la página 315.

Actualización de los elementos DOM con contenido localizado

Un elemento de la aplicación puede hacer referencia a un valor de clave de un archivo de propiedades de localización. En el ejemplo siguiente, el elemento `title` especifica un atributo `local_innerHTML`. La arquitectura de localización utiliza este atributo para buscar un valor localizado. De forma predeterminada, la arquitectura busca nombres de atributo que empiezan con "local_". La arquitectura actualiza los atributos cuyos nombres coinciden con el texto que sigue a "local_". En este caso, la arquitectura define el atributo `innerHTML` del elemento `title`. El atributo `innerHTML` utiliza el valor definido para la clave `mainWindowTitle` en el archivo de propiedades predeterminadas (`default.properties`):

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Si la configuración regional actual no define ningún valor que coincida, la arquitectura de localización busca en el resto de la cadena de configuraciones regionales. Utiliza la siguiente configuración regional de la cadena que tenga definido un valor.

En el siguiente ejemplo el texto (el atributo `innerHTML`) del elemento `p` utiliza el valor de la clave `greeting` definido en el archivo de propiedades predeterminadas:

```
<p local_innerHTML="default.greeting" />
```

En el siguiente ejemplo el atributo del valor (y el texto mostrado) del elemento `input` utiliza el valor de la clave `btnBlue` definido en el archivo de propiedades predeterminadas:

```
<input type="button" local_value="default.btnBlue" />
```

Para actualizar el DOM de HTML para que utilice las cadenas de caracteres definidas en la cadena de configuraciones regionales actual, llame al método `update()` del objeto `Localizer`. Al llamar al método `update()` el objeto `Localizer` analiza el DOM y aplica manipulaciones donde encuentre atributos de localización ("local_..."):

```
air.Localizer.localizer.update();
```

Se pueden definir valores tanto para un atributo ("innerHTML", por ejemplo) como para su correspondiente atributo de localización ("local_innerHTML", por ejemplo). En este caso, la arquitectura de localización solo sobrescribe el valor del atributo si encuentra un valor coincidente en la cadena de localización. Por ejemplo, el siguiente elemento define ambos atributos, `value` y `local_value`:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

También puede actualizarse un solo elemento DOM en particular. Consulte el apartado siguiente, [“Actualización de elementos DOM para utilizar la configuración regional actual”](#) en la página 312.

De forma predeterminada, AIR HTML Localizer utiliza "local_" como prefijo para los atributos que definen las opciones de localización para un elemento. Por ejemplo: de forma predeterminada, un atributo `local_innerHTML` define el nombre del paquete y recurso que se utiliza para el valor `innerHTML` de un elemento. También de forma predeterminada, un atributo `local_value` define el nombre del paquete y recurso que se utiliza para el atributo `value` de un elemento. Se puede configurar AIR HTML Localizer para que utilice otro prefijo de atributo en lugar de "local_". Consulte [“Personalización de las opciones de AIR HTML Localizer”](#) en la página 314.

Actualización de elementos DOM para utilizar la configuración regional actual

Cuando el objeto `Localizer` actualiza del DOM de HTML, hace que los elementos marcados utilicen valores de atributo basados en las cadenas de caracteres definidas en la actual cadena de configuraciones regionales. Para que el localizador de HTML actualice el DOM de HTML, llame al método `update()` del objeto `Localizer`:

```
air.Localizer.localizer.update();
```

Para actualizar un solo elemento DOM especificado, páselo como parámetro al método `update()`. El método `update()` tiene un solo parámetro, `parentNode`, que es optativo. Cuando está especificado, el parámetro `parentNode` define el elemento DOM que se debe localizar. Llamar al método `update()` y especificar un parámetro `parentNode` define valores localizados para todos los elementos secundarios que especifican atributos de localización.

Por ejemplo, tomemos el siguiente elemento `div`:

```
<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>
```

Para actualizar este elemento de modo que utilice cadenas de caracteres localizadas en la cadena de configuraciones regionales actual, use el código JavaScript siguiente:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Si no se encuentra un valor de clave en la cadena de configuraciones regionales, la arquitectura de localización define como valor de atributo el valor del atributo `"local_"`. Por ejemplo: supongamos que en el ejemplo anterior la arquitectura de localización no encuentra ningún valor para la clave `lblColors` (en ninguno de los archivos `default.properties` de la cadena de configuraciones regionales). En este caso, utiliza `"default.lblColors"` como el valor de `innerHTML`. El uso de este valor indica (al desarrollador) que faltan recursos.

El método `update()` distribuye un evento `resourceNotFound` cuando no encuentra un recurso en la cadena de configuraciones regionales. La constante `air.Localizer.RESOURCE_NOT_FOUND` define la cadena `"resourceNotFound"`. El evento tiene tres propiedades: `bundleName`, `resourceName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `resourceName` es el nombre del recurso no disponible. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `update()` distribuye un evento `bundleNotFound` cuando no encuentra el paquete especificado. La constante `air.Localizer.BUNDLE_NOT_FOUND` define la cadena `"bundleNotFound"`. El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `update()` funciona de modo asíncrono (y distribuye los eventos `resourceNotFound` y `bundleNotFound` de forma asíncrona). El siguiente código define detectores para los eventos `resourceNotFound` y `bundleNotFound`:

```
air.Localizer.localizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, bnfHandler);
air.Localizer.localizer.update();
function rnfHandler(event)
{
  alert(event.bundleName + ": " + event.resourceName + ":@" + event.locale);
}
function bnfHandler(event)
{
  alert(event.bundleName + ":@" + event.locale);
}
```

Personalización de las opciones de AIR HTML Localizer

El método `setBundlesDirectory()` del objeto `Localizer` permite personalizar la ruta del directorio de paquetes. El método `setLocalAttributePrefix()` del objeto `Localizer` permite personalizar la ruta del directorio de paquetes y el valor de atributo que utiliza el `Localizer`.

El directorio de paquetes predeterminado se define como el subdirectorio de configuraciones regionales del directorio de la aplicación. Para especificar otro directorio, llame al método `setBundlesDirectory()` del objeto `Localizer`. Este método utiliza un parámetro, `path`, que es la ruta al directorio de paquetes deseado, en forma de cadena de caracteres. El parámetro `path` puede tener cualquiera de los valores siguientes:

- Una cadena que define una ruta relativa al directorio de la aplicación, como "configregionales"
- Una cadena que define una URL válida que utiliza los esquemas de URL `app`, `app-storage` o `file`, por ejemplo "app://languages" (*no* utilice el esquema de URL `http`)
- Un objeto `File`

Para obtener información sobre direcciones URL y rutas de directorio, consulte:

- [Rutas a objetos File](#) (para desarrolladores de ActionScript)
- [Rutas a objetos File](#) (para desarrolladores de HTML)

En el siguiente ejemplo, el código define como directorio de paquetes un subdirectorio "languages" del directorio de almacenamiento de la aplicación (y no el directorio de la aplicación):

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

Pase una ruta válida como parámetro `path`. De lo contrario, el método emite una excepción `BundlePathNotFoundError`. Este error tiene a "BundlePathNotFoundError" como su propiedad `name` y su propiedad `message` especifica la ruta no válida.

De forma predeterminada, AIR HTML Localizer utiliza "local_" como prefijo para los atributos que definen las opciones de localización para un elemento. Por ejemplo, el atributo `local_innerHTML` define el nombre del paquete y recurso que se utiliza para el valor `innerHTML` del siguiente elemento `input`:

```
<p local_innerHTML="default.greeting" />
```

El método `setLocalAttributePrefix()` del objeto `Localizer` permite utilizar otro prefijo de atributo en lugar de "local_". Este método estático utiliza un parámetro, que es la cadena de caracteres que se desea utilizar como prefijo de atributo. En el siguiente ejemplo, el código define la arquitectura de localización para que utilice "loc_" como prefijo de atributo:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Se puede personalizar el prefijo de atributo que utiliza la arquitectura de localización. Puede ser conveniente personalizar el prefijo si el valor predeterminado ("local_") está en conflicto con el nombre de otro atributo que se utilice en el código. Cuando llame a este método, asegúrese de utilizar caracteres válidos para los atributos de HTML. (Por ejemplo, el valor no puede contener un carácter de espacio en blanco).

Para obtener más información sobre el uso de atributos de localización en elementos HTML, consulte "[Actualización de los elementos DOM con contenido localizado](#)" en la página 312.

Las opciones de directorio de paquetes y prefijo de atributo no persisten entre distintas sesiones de la aplicación. Si utiliza opciones personalizadas para el directorio de paquetes o el prefijo de atributo, asegúrese de configurarlas cada vez que inicie la aplicación.

Definición de la cadena de configuraciones regionales

Cuando se carga el código de AIRLocalizer.js, define automáticamente la cadena de configuraciones regionales predeterminada. Las configuraciones regionales disponibles en el directorio de paquetes y la configuración de idiomas del sistema operativo definen esta cadena de configuraciones regionales. (Para obtener más información, consulte [“Gestión de cadenas de configuraciones regionales”](#) en la página 311).

Se puede modificar la cadena de configuraciones regionales llamando al método estático `setLocaleChain()` del objeto `Localizer`. Por ejemplo, puede ser conveniente llamar a este método si el usuario indica una preferencia para un idioma concreto. El método `setLocaleChain()` utiliza un solo parámetro, `chain`, que es un conjunto de configuraciones regionales, por ejemplo `["fr_FR", "fr", "fr_CA"]`. El orden de las configuraciones locales en el conjunto define el orden en que la arquitectura busca recursos (en operaciones posteriores). Si no se encuentra un recurso para la primera configuración regional de la cadena, sigue buscando en los recursos de la otra configuración regional. Si falta el argumento `chain`, o si no es un conjunto o es un conjunto vacío, la función falla y emite una excepción `IllegalArgumentsError`.

El método estático `getLocaleChain()` del objeto `Localizer` devuelve un conjunto que enumera las configuraciones regionales de la cadena de configuraciones regionales actual.

El siguiente código lee la cadena de configuraciones regionales actual y añade dos configuraciones regionales francesas a la cabeza de la cadena:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

El método `setLocaleChain()` distribuye un evento "change" cuando actualiza la cadena de configuraciones regionales. La constante `air.Localizer.LOCALE_CHANGE` define la cadena "change". El evento tiene una propiedad, `localeChain`, que es un conjunto de códigos de configuración regional en la nueva cadena de configuraciones regionales. El siguiente código define un detector para este evento:

```
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);
}
```

La propiedad estática `air.Localizer.ultimateFallbackLocale` representa la configuración regional que se utiliza cuando la aplicación no admite ninguna de las preferencias del usuario. El valor predeterminado es "en". Se lo puede cambiar a otra configuración regional, como en el código siguiente:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Obtención de recursos para una configuración regional específica

El método `getString()` del objeto `Localizer` devuelve la cadena definida para un recurso en una configuración regional específica. No hace falta especificar un valor `locale` cuando se llama al método. En este caso el método busca en toda la cadena de configuraciones regionales y devuelve la cadena de caracteres de la primera configuración regional que proporciona el nombre del recurso especificado. El método utiliza los siguientes parámetros:

Parámetro	Descripción
bundleName	El paquete que contiene el recurso. Es el nombre del archivo de propiedades sin la extensión .properties. Por ejemplo: si este parámetro está definido en "alerts", el código del Localizer busca en archivos de localización que tengan el nombre alerts.properties.
resourceName	El nombre del recurso.
templateArgs	Opcional. Un conjunto de cadenas para sustituir las etiquetas numeradas en la cadena de sustitución. Tomemos como ejemplo una llamada a la función en que el parámetro templateArgs es ["Raúl", "4"] y la cadena del recurso coincidente es "Hello, {0}. You have {1} new messages.". En este caso, la función devuelve "Hello, Raúl. You have 4 new messages.". Para pasar por alto esta opción, pase un valor null.
locale	Opcional. El código de la configuración regional (por ejemplo: "en", "en_us" o "fr") que se debe utilizar. Si se facilita una configuración regional y no se encuentra ningún valor coincidente, el método no seguirá buscando valores en otras configuraciones regionales de la cadena. Si no se especifica ningún código de configuración regional, la función devuelve la cadena de caracteres que está en la primera configuración regional que proporciona un valor para el nombre del recurso especificado.

La arquitectura de localización puede actualizar los atributos marcados del DOM de HTML. Hay también otras formas de utilizar cadenas localizadas. Por ejemplo, se puede utilizar una cadena de caracteres en HTML generado de forma dinámica o como valor de parámetro en una llamada a una función. En el siguiente ejemplo, el código llama a la función `alert()` con la cadena de caracteres definida en el recurso `error114` del archivo de propiedades predeterminadas de la configuración regional `fr_FR`:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

El método `getString()` distribuye un evento `resourceNotFound` cuando no encuentra el recurso en el paquete especificado. La constante `air.Localizer.RESOURCE_NOT_FOUND` define la cadena `resourceNotFound`. El evento tiene tres propiedades: `bundleName`, `resourceName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `resourceName` es el nombre del recurso no disponible. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `getString()` distribuye un evento `bundleNotFound` cuando no encuentra el paquete especificado. La constante `air.Localizer.BUNDLE_NOT_FOUND` define la cadena `bundleNotFound`. El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `getString()` funciona de modo asíncrono (y distribuye los eventos `resourceNotFound` y `bundleNotFound` de forma asíncrona). El siguiente código define detectores para los eventos `resourceNotFound` y `bundleNotFound`:

```
air.Localizerlocalizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizerlocalizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, bnfHandler);
var str = air.Localizer.localizer.getString("default", "error114", null, "fr_FR");
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":@" + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":@" + event.locale);
}
```

El método `getResourceBundle()` del objeto `Localizer` devuelve un paquete especificado para una configuración regional determinada. El valor devuelto del método es un objeto con propiedades que coinciden con las claves del paquete. (Si la aplicación no puede encontrar el paquete especificado, el método devuelve `null`.)

El método adopta dos parámetros: `locale` y `bundleName`.

Parámetro	Descripción
<code>locale</code>	Configuración regional (p. ej. "fr").
<code>bundleName</code>	Nombre del paquete.

Por ejemplo, el siguiente código llama al método `document.write()` para cargar el paquete predeterminado para la configuración regional `fr`. A continuación llama al método `document.write()` para escribir valores de las claves `str1` y `str2` en ese paquete:

```
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/>");
```

El método `getResourceBundle()` distribuye un evento `bundleNotFound` cuando no encuentra el paquete especificado. La constante `air.Localizer.BUNDLE_NOT_FOUND` define la cadena "bundleNotFound". El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `getFile()` del objeto `Localizer` devuelve el contenido de un paquete, en forma de cadena, para una configuración regional determinada. El archivo del paquete se lee como archivo UTF-8. El método incluye los siguientes parámetros:

Parámetro	Descripción
<code>resourceFileName</code>	El nombre del archivo del recurso (por ejemplo, "about.html").
<code>templateArgs</code>	Opcional. Un conjunto de cadenas para sustituir las etiquetas numeradas en la cadena de sustitución. Tomemos como ejemplo una llamada a la función en que el parámetro <code>templateArgs</code> es ["Raúl", "4"] y el archivo del recurso coincidente contiene dos líneas: <pre><html> <body>Hello, {0}. You have {1} new messages.</body> </html></pre> En este caso, la función devuelve una cadena de dos líneas: <pre><html> <body>Hello, Raúl. You have 4 new messages. </body> </html></pre>
<code>locale</code>	El código de la configuración regional (por ejemplo: "es_ES") a utilizarse. Si se facilita una configuración regional y no se encuentra ningún archivo coincidente, el método no seguirá buscando en otras configuraciones regionales de la cadena. Si no se especifica <i>ningún</i> código de configuración regional, la función devuelve el texto de la primera configuración regional de la cadena que tenga un archivo que coincida con el nombre de archivo del recurso, <code>resourceFileName</code> .

En el siguiente ejemplo, el código llama al método `document.write()` utilizando el contenido del archivo `about.html` file de la configuración regional `fr`:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

El método `getFile()` distribuye un evento `fileNotFound` cuando no encuentra un recurso en la cadena de configuraciones regionales. La constante `air.Localizer.FILE_NOT_FOUND` define la cadena `"resourceNotFound"`. El método `getFile()` funciona de modo asíncrono (y distribuye el evento `fileNotFound` de forma asíncrona). El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `fileName` es el nombre del archivo que no se encuentra. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso. El siguiente código define un detector para este evento:

```
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);
}
```

Más temas de ayuda

[Building a multilingual HTML-based application \(Creación de una aplicación multilingüe basada en HTML\)](#)

Capítulo 21: Variables del entorno de ruta

El SDK de AIR contiene algunos programas que se pueden iniciar desde una línea de comandos o terminal. La ejecución de estos programas suele ser más adecuada cuando la ruta al directorio bien del SDK se incluye en la variable del entorno de la ruta.

La información presentada aquí analiza cómo establecer la ruta en Windows, Mac y Linux y debe servir como guía adecuada. Sin embargo, las configuraciones de los equipos varían en gran medida, por lo que el procedimiento no funciona para todos los sistemas. En estos casos, se debe poder encontrar la información necesaria la documentación del sistema operativo o Internet.

Configuración de la ruta en Linux y Mac OS utilizando el shell Bash

Cuando se escribe un comando en una ventana de terminal, el shell, un programa que lee lo que se escribe e intenta responder adecuadamente, debe localizar en primer lugar el programa del comando en el sistema de archivos. El shell busca comandos en una lista de directorios almacenados en una variable de entorno denominada \$PATH. Para ver que se incluye realmente en la ruta, indique:

```
echo $PATH
```

Esto devuelve una lista de directorios separados por puntos y comas que debe presentar el siguiente aspecto:

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

El objetivo es añadir la ruta al directorio bin del SDK de AIR para que el shell pueda encontrar las herramientas ADT y ADL. Asumiendo que el SDK de AIR se ha ubicado en /Users/fred/SDKs/AIR, el siguiente comando añadirá los directorios necesarios a la ruta:

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Nota: si la ruta contiene caracteres de espacio en blanco, sustitúyalos por una barra invertida tal y como se indica a continuación:

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

Se puede usar de nuevo el comando echo para asegurarse de que haya funcionado:

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Hasta ahora todo claro. Ahora se deben poder escribir los siguientes comandos y obtener una respuesta apropiada:

```
adt -version
```

Si se ha modificado la variable \$PATH correctamente, el comando debe indicar la nueva versión de ADT.

Sin embargo, aún existe un problema; la próxima vez que se active una nueva ventana de terminal, observará que las nuevas entradas en la ruta ya no se encuentran allí. El comando para establecer la ruta se debe ejecutar cada vez que se inicie un nuevo terminal.

Una solución común a este problema consiste en añadir el comando a uno de los scripts de inicio utilizado por el shell. En Mac OS, se puede crear el archivo, `.bash_profile`, en el directorio `~/username` y se ejecutará cada vez que se abra en una nueva ventana del terminal. En Ubuntu, el script de inicio se ejecuta cuando se inicia una nueva ventana de terminal (`.bashrc`). Otras distribuciones de Linux y programas shell disponen de convenciones similares.

Para añadir el comando al script de inicio del shell:

- 1 Cambie al directorio principal:

```
cd
```

- 2 Cree el perfil de configuración del shell (si es necesario) y redirija el texto que escriba al final del archivo con `"cat >>"`. Utilice el archivo adecuado para el shell y el sistema operativo. Por ejemplo, se puede utilizar `.bash_profile` en Mac OS y `.bashrc` en Ubuntu.

```
cat >> .bash_profile
```

- 3 Indique el texto para agregar al archivo:

```
export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
```

- 4 Termine la redirección de texto presionando `CTRL-SHIFT-D` en el teclado.

- 5 Muestre el archivo para asegurarse de que todo es correcto:

```
cat .bash_profile
```

- 6 Abra una nueva ventana de terminal para comprobar la ruta:

```
echo $PATH
```

Las adiciones de ruta se deben incluir.

Si posteriormente se crea una nueva versión de uno de los SDKs en un directorio distinto, asegúrese de actualizar el comando `path` en el archivo de configuración. De lo contrario, el shell continuará utilizando la antigua versión.

Configuración de la ruta en Windows

Cuando se abre una ventana de comando en Windows, esa ventana hereda las variables del entorno global definidas en las propiedades del sistema. Una de las variables importantes es la ruta, que es la lista de directorios que el programa de comandos busca cuando se escribe el nombre de un programa que se va a ejecutar. Para ver lo que se incluye actualmente en la ruta cuando se está utilizando una ventana de comandos, puede escribir lo siguiente:

```
set path
```

Se mostrará una lista de directorios separados por puntos y comas que presenta el siguiente aspecto:

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

El objetivo es añadir la ruta al directorio `bin` del SDK de AIR en la lista, de modo que el programa de comandos pueda buscar las herramientas `ADT` y `ADL`. Si se considera que el SDK de AIR se la ubicado en `C:\SDKs\AIR`, es posible añadir la entrada de ruta adecuada con el siguiente procedimiento:

- 1 Abra el cuadro de diálogo Propiedades del sistema en el Panel de control, o bien, haga clic con el botón derecho en el icono Mi PC y seleccione Propiedades en el menú.
- 2 En la ficha Avanzadas, haga clic en el botón Variables del entorno.
- 3 Seleccione la entrada Ruta en la sección de variables del sistema del cuadro de diálogo Variables del entorno.
- 4 Haga clic en Editar.

5 Desplácese al final del text en el campo del valor Variable.

6 Indique el siguiente texto al final del valor actual:

```
;C:\SDKs\AIR\bin
```

7 Haga clic en Aceptar en todos los cuadros de diálogo para guardar la ruta.

Si existe alguna ventana de comandos abierta, observe que sus entornos no están actualizados. Abra una nueva ventana de comandos y escriba el siguiente comando para garantizar que las rutas se hayan configurado correctamente:

```
adt -version
```

Si posteriormente se cambia la ubicación del SDK de AIR o se añade una nueva versión, recuerde actualizar la variable de ruta.