Construindo aplicativos no ADOBE® AIR®

Avisos legais

Última atualização em 31/5/2017

Conteúdo

Capítulo 1: Sobre o Adobe AIR

O Adobe® AIR® é um de runtime de várias telas e sistema multioperacional que permite potencializar suas habilidades de desenvolvimento para a Web a fim de criar e implantar aplicativos avançados da Internet (RIAs - rich Internet applications) para computadores desktop e dispositivos móveis. Aplicativos para desktop, televisão e AIR móveis podem ser desenvolvidos com ActionScript 3.0 usando Adobe® Flex e Adobe® Flash® (com base SWF). Os aplicativos de desktop AIR também podem ser desenvolvidos com HTML, JavaScript® e Ajax (com base HTML).

Você pode encontrar mais informações sobre introdução e uso do Adobe AIR na Conexão de desenvolvedores do Adobe AIR em (http://www.adobe.com/devnet/air/).

O AIR permite que você trabalhe em ambientes domésticos para potencializar as ferramentas e abordagens que julgar mais confortáveis. Ao suportar Flash, Flex, HTML, JavaScript e Ajax, você pode criar a melhor experiência possível que atenda às suas necessidades.

Por exemplo, os aplicativos podem ser desenvolvidos usando uma das seguintes tecnologias ou uma combinação delas:

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

Usuários interagem com aplicativos AIR da mesma forma que interagem com aplicativos nativos. O runtime é instalado uma vez no dispositivo ou computador do usuário e, em seguida, os aplicativos AIR são instalados e executados exatamente como qualquer outro aplicativo da área de trabalho. (No iOS o runtime de um AIR separado não é instalado; cada aplicativo do AIR iOS é um aplicativo independente).

O runtime oferece uma plataforma cruzada de sistema operacional e estrutura para implantação de aplicativos e, portanto, elimina os testes entre navegadores, assegurando funcionalidade e interações consistentes entre áreas de trabalho. Em vez de desenvolver para um sistema operacional específico, você direciona o runtime, o que oferece seguintes benefícios:

- Aplicativos desenvolvidos para execução do AIR através de vários sistemas operacionais sem nenhum trabalho adicional feito por você. O runtime assegura apresentações e interações previsíveis e consistentes entre todos os sistemas operacionais com suporte do AIR.
- Os aplicativos podem ser criados mais rapidamente, permitindo que você potencialize tecnologias da Web e padrões de projeto existentes. Você pode estender aplicativos com base na Web para a área de trabalho sem aprender tecnologias tradicionais para desenvolvimento de área de trabalho nem a complexidade do código nativo.
- O desenvolvimento do aplicativo é mais fácil do que o uso de linguagens de nível inferior, como C e C++. Você não precisa gerenciar APIs complexas de nível inferior, específicas de cada sistema operacional.

Ao desenvolver aplicativos para o AIR, você pode potencializar um enorme conjunto de estruturas e APIs:

- APIs específicas do AIR fornecidas pelo runtime e pela estrutura AIR
- APIs do ActionScript usadas em arquivos SWF e na estrutura Flex (bem como outras bibliotecas e estruturas baseadas no ActionScript)
- HTML, CSS e JavaScript
- Maioria das estruturas Ajax
- As extensões nativas do Adobe AIR fornecem APIs do ActionScript que permitem o acesso à funcionalidade específica da plataforma programada no código nativo. As extensões nativas também poderão permitir o acesso ao código nativo legado e ao código nativo que possibilita um melhor desempenho.

O AIR altera significativamente o modo como os aplicativos podem ser criados, implantados e experimentados. Você obtém mais controle criativo e pode estender os aplicativos baseados em Flash, Flex, HTML e Ajax para a área de trabalho, dispositivos móveis e televisões.

Para obter mais informações sobre o que está incluído em cada nova versão do AIR, consulte as Notas de versão do Adobe AIR (http://www.adobe.com/go/learn_air_relnotes_br).

Capítulo 2: Instalação do Adobe AIR

O runtime do Adobe® AIR® permite executar aplicativos AIR. É possível instalar o runtime das seguintes formas:

- Instalando o runtime separadamente (sem instalar também um aplicativo do AIR)
- Ao instalar um aplicativo do AIR por meio de um "badge" de instalação de uma página web (também será solicitada a instalação do runtime)
- Ao criar um instalador customizado que instale tanto o aplicativo quanto o runtime. É necessário obter uma aprovação do Adobe para distribuir o runtime do AIR dessa maneira. Você poderá solicitar uma aprovação na página Licença do runtime do Adobe. Observe que o Adobe não fornece ferramentas para criar tal instalador. No entanto, muitos kits de ferramenta de instalador terceirizado estão disponíveis.
- Ao instalar um aplicativo do AIR que cria um conjunto do AIR como um runtime cativo. Um runtime cativo é usado somente pelo aplicativo que cria o conjunto. Ele não é usado para executar outros aplicativos do AIR. Criar um conjunto do runtime é uma opção no Mac e no Windows. No iOS, todos os aplicativos incluem um runtime em conjunto. A partir do AIR 3.7, todos os aplicativos Android incluem um runtime em conjunto por padrão (embora você possa usar um runtime separado).
- Configurando um ambiente de desenvolvimento AIR como o AIR SDK, Adobe® Flash® Builder™, ou o Adobe Flex® SDK (que inclui as ferramentas de desenvolvimento de linha de comando do AIR). O runtime incluído no SDK somente é utilizado para a depuração de aplicativos — não é utilizado para executar aplicativos AIR instalados.

Os requisitos do sistema para instalar o AIR e executar aplicativos do AIR são detalhados aqui: Adobe AIR: Requisitos do sistema (http://www.adobe.com/br/products/air/systemreqs/).

Tanto o programa de instalação do runtime quanto o programa de instalação do aplicativo do AIR criam arquivos de registro quando instalam, atualizam ou removem aplicativos AIR ou o próprio runtime do AIR. Você pode consultar estes arquivos de registro para ajudar a determinar a causa de problemas de instalação. Consulte Registros de instalação.

Instalação do Adobe AIR

Para instalar ou atualizar o runtime, o usuário deve ter privilégios administrativos no computador.

Instalação do runtime em um computador com Windows

1. Faça download do arquivo de instalação do runtime em http://get.adobe.com/air.
2. Clique duas vezes no arquivo de instalação do runtime.
3. Na janela de instalação, siga os avisos para concluir a instalação.

Instalação do runtime em um computador com Mac

1. Faça download do arquivo de instalação do runtime em http://get.adobe.com/air.
2. Clique duas vezes no arquivo de instalação do runtime.
3. Na janela de instalação, siga os avisos para concluir a instalação.
4. Se o instalador exibir a janela Autenticação, insira seu nome de usuário e sua senha do Mac OS.

Instalação do runtime em um computador com Linux

Nota: *Atualmente, o AIR 2.7 e versões posteriores não são compatíveis com Linux. Os aplicativos do AIR implementados para Linux deverão continuar usando o AIR 2.6 SDK.*

Utilizando o instalador binário:

1. Localize o arquivo binário de instalação em http://kb2.adobe.com/cps/853/cpsid_85304.html e faça o download.
2. Defina as permissões de arquivo para que seja possível executar o aplicativo instalador. A partir da linha de comando, é possível definir as permissões de arquivo com:

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

   Algumas versões do Linux permitem definir as permissões de arquivo na caixa de diálogo Propriedades aberta por meio de um menu de contexto.
3. Execute o instalador da linha de comando ou clicando duas vezes no arquivo de instalação do runtime.
4. Na janela de instalação, siga os avisos para concluir a instalação.

O Adobe AIR está instalado como pacote nativo. Ou seja, como rpm e uma distribuição com base em rpm e deb em uma distribuição Debian. Atualmente o AIR não possui suporte a outro tipo de formato de pacote.

Utilizando os instaladores de pacote:

1. Localize o arquivo de pacote do AIR em http://kb2.adobe.com/cps/853/cpsid_85304.html. Faça download do pacote rpm ou Debian, dependendo do formato de pacote suportado pelo sistema.
2. Caso necessário, clique duas vezes no arquivo do pacote AIR para instalar o aplicativo.

   Opcionalmente, é possível instalar o pacote a partir da linha de comando:

   a. Em um sistema Debian:

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

   b. Em um sistema com base em rpm:

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

      Ou, caso esteja atualizando uma versão existente (AIR 1.5.3 ou posterior):

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

A instalação de aplicativos AIR 2 e AIR necessitam que você possua privilégios administrativos em seu computador.

O Adobe AIR é instalado no caminho a seguir: /opt/Adobe AIR/Versions/1.0

O AIR registra o mime-type "application/vnd.adobe.air-application-installer-package+zip", significando que arquivos .air pertencem a esse mime-type e são registrados com o runtime AIR.

Instale o runtime em um dispositivo Android

Você pode instalar a versão mais recente do runtime AIR a partir do Android Market.

Você pode instalar versões de desenvolvimento do runtime AIR a partir de um link em um site ou usando o comando ADT `-installRuntime`. Somente uma versão do runtime AIR pode ser instalado em um tempo; você não pode ter uma nova versão e uma versão de desenvolvimento instaladas.

Consulte “Comando installRuntime do ADT” na página 180 para obter mais informações.

Instale o runtime em um dispositivo iOS

O código do runtime AIR necessário é fornecido com cada aplicativo criado para os dispositivos iPhone, iTouch e iPad. Você não precisa instalar um componente de runtime separado.

Mais tópicos da Ajuda

"AIR for iOS" na página 70

Remoção do Adobe AIR

Após instalar o runtime, você poderá removê-lo usando os seguintes procedimentos.

Remoção do runtime em um computador com Windows

1. No menu Iniciar do Windows, selecione Configurações > Painel de controle.
2. Abra o painel de controle Programas, Programas e Recursos ou Adicionar ou Remover Programas (dependendo de qual versão do Windows que estiver em uso).
3. Selecione "Adobe AIR" para remover o runtime.
4. Clique no botão Alterar/remover.

Remoção do runtime em um computador com Mac

- Clique duas vezes em "Desinstalador do Adobe AIR", localizado na pasta /Aplicativos/Utilitários.

Remoção do runtime em um computador com Linux

Execute um dos seguintes procedimentos:

- Selecione o comando "Adobe AIR Uninstaller" no menu Aplicativos.
- Execute o binário instalador do AIR com a opção `-uninstall`.
- Remova os pacotes do AIR (`adobeair` e `adobecerts`) com o gerenciador de pacotes.

Remova o runtime de um dispositivo Android

1. Abra o aplicativo Configurações do dispositivo.
2. Toque a entrada do Adobe AIR em Aplicativos > Gerenciar aplicativos.
3. Toque o botão Desinstalar.

Você também pode usar o comando ADT `-uninstallRuntime`. Consulte "Comando uninstallRuntime do ADT" na página 181 para obter mais informações.

Remoção de um runtime em conjunto

Para remover um runtime em conjunto, é necessário remover o aplicativo com o qual ele está instalado. Observe que os tempos de execução cativos são usados somente para executar o aplicativo de instalação.

Instalação e execução de aplicativos de amostra do AIR

Para instalar ou atualizar um aplicativo do AIR, o usuário deve ter privilégios administrativos no computador.

Alguns aplicativos de amostra estão disponíveis para demonstrar recursos do AIR Você pode acessá-los e desinstalá-los usando as seguintes instruções:

1. Faça download e execute os aplicativos de amostra do AIR. Os aplicativos compilados, bem como o código fonte, estão disponíveis.

2 Para fazer download e executar o aplicativo de amostra, clique no botão Instalar agora do aplicativo de amostra. Você é solicitado a instalar e executar o aplicativo.

3 Se você optar por fazer download de aplicativos de amostra e executá-los mais tarde, selecione os links de download. Você pode executar aplicativos AIR a qualquer momento:

- No Windows, clicando duas vezes no ícone do aplicativo na área de trabalho, ou selecionando-o no menu Iniciar do Windows.
- No Mac OS, clicando duas vezes no ícone do aplicativo, que está instalado na pasta Aplicativos do diretório do usuário (por exemplo, no Macintosh, HD/Users/JoeUser/Applications/), por padrão.

Nota: *Verifique as notas de versão do AIR para atualizações dessas instruções, que estão localizadas aqui: http://www.adobe.com/go/learn_air_relnotes_br.*

Atualizações do Adobe AIR

Periodicamente, a Adobe atualiza o Adobe AIR com novos recursos ou o corrige para diminuir problemas. Os recursos Notificação Automática e Atualizar permitem à Adobe notificar automaticamente os usuários quando um versão atualizada do Adobe AIR está disponível.

As atualizações do Adobe AIR garantem que o Adobe AIR funcione adequadamente e possam conter modificações importantes na segurança. A Adobe recomenda que os usuários atualizem para a versão mais recente do Adobe AIR sempre que uma nova versão estiver disponível, especialmente quando uma atualização de segurança for indicada.

Por padrão, quando um aplicativo do AIR é lançado, o runtime verifica se uma atualização está disponível. Ele realiza esta verificação sempre que decorrem mais de duas semanas da última verificação de atualização. Se uma atualização estiver disponível, o AIR baixará a atualização em segundo plano.

Os usuários podem desativar a capacidade de atualização automática usando o aplicativo do AIR SettingsManager. O aplicativo do AIR SettingsManager está disponível para download em http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air.

O processo de instalação normal do Adobe AIR inclui a conexão com http://airinstall.adobe.com para enviar informações básicas sobre o ambiente de instalação, tais como versão e idioma do sistema operacional. Esta informação só é transmitida uma vez em cada instalação e permite que o Adobe confirme se a instalação foi concluída com êxito. Nenhuma informação que possa identificar o usuário é coletada ou transmitida.

Atualização dos tempos de execução cativos

Se você distribuir o aplicativo com um conjunto de runtime cativo, o runtime cativo não será atualizado automaticamente. Para a segurança dos usuários, é necessário monitorar as atualizações publicadas pela Adobe e atualizar o aplicativo com a nova versão do runtime quando uma alteração de segurança importante for publicada.

Capítulo 3: Trabalhando com APIs do AIR

O Adobe® AIR® inclui funcionalidade que não está disponível para conteúdo do SWF em execução no Adobe® Flash® Player.

Desenvolvedores de ActionScript 3.0
As APIs do Adobe AIR são documentadas nos seguintes livros:

- Guia do desenvolvedor do ActionScript 3.0
- Referência do ActionScript® 3.0 para Adobe® Flash® Platform

Desenvolvedores de HTML
Se você estiver criando aplicativos AIR com base em HTML, as APIs disponíveis em JavaScript via arquivo AIRAliases.js (consulte Acessando classes de API do AIR a partir do JavaScript) estão documentadas nos seguintes livros:

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

Classes do ActionScript 3.0 específicas do AIR

A tabela a seguir contém classes de tempos de execução específicas do Adobe AIR. Elas não estão disponíveis no conteúdo SWF executado no Adobe® Flash® Player em um navegador.

Desenvolvedores de HTML
As classes disponíveis no JavaScript via arquivo AIRAliases.js estão indicadas em Referência de API do Adobe AIR para desenvolvedores de HTML.

Classe	Pacote ActionScript 3.0	Incluído na versão 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

Classe	Pacote ActionScript 3.0	Incluído na versão 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
DatagramSocketDataEvent	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 e anteriores; desativado a partir da versão 3.7
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 e anteriores; desativado a partir da versão 3.7
GameInputHand	flash.ui	3.6 e anteriores; desativado a partir da versão 3.7
Geolocation	flash.sensors	2.0
DatagramSocketDataEvent	flash.events	2.0
HTMLHistoryItem	flash.html	1.0
HTMLHost	flash.html	1.0
HTMLLoader	flash.html	1.0

Classe	Pacote ActionScript 3.0	Incluído na versão AIR
HTMLPDFCapability	flash.html	1.0
HTMLSWFCapabiltiy	flash.html	2.0
HTMLUncaughtScriptExceptionEvent	flash.events	1.0
HTMLWindowCreateOptions	flash.html	1.0
Icon	flash.desktop	1.0
IFilePromise	flash.desktop	2.0
ImageDecodingPolicy	flash.system	2.6
InteractiveIcon	flash.desktop	1.0
InterfaceAddress	flash.net	2.0
InvokeEvent	flash.events	1.0
InvokeEventReason	flash.desktop	1.5.1
IPVersion	flash.net	2.0
IURIDereferencer	flash.security	1.0
LocationChangeEvent	flash.events	2.5
MediaEvent	flash.events	2.5
MediaPromise	flash.media	2.5
MediaType	flash.media	2.5
MXRecord	flash.net.dns	2.0
NativeApplication	flash.desktop	1.0
NativeDragActions	flash.desktop	1.0
NativeDragEvent	flash.events	1.0
NativeDragManager	flash.desktop	1.0
NativeDragOptions	flash.desktop	1.0
NativeMenu	flash.display	1.0
NativeMenuItem	flash.display	1.0
NativeProcess	flash.desktop	2.0
NativeProcessExitEvent	flash.events	2.0
NativeProcessStartupInfo	flash.desktop	2.0
NativeWindow	flash.display	1.0
NativeWindowBoundsEvent	flash.events	1.0
NativeWindowDisplayState	flash.display	1.0
NativeWindowDisplayStateEvent	flash.events	1.0
NativeWindowInitOptions	flash.display	1.0
NativeWindowRenderMode	flash.display	3.0

Classe	Pacote ActionScript 3.0	Incluído na versão 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
MXRecord	flash.net.dns	2.0
ReferencesValidationSetting	flash.security	1.0
ResourceRecord	flash.net.dns	2.0
RevocationCheckSettings	flash.security	1.0
Screen	flash.display	1.0
ScreenMouseEvent	flash.events	1.0
SecureSocket	flash.net	2.0
SecureSocketMonitor	air.net	2.0
ServerSocket	flash.net	2.0
ServerSocketConnectEvent	flash.events	2.0
ServiceMonitor	air.net	1.0
SignatureStatus	flash.security	1.0
SignerTrustSettings	flash.security	1.0
SocketMonitor	air.net	1.0
SoftKeyboardType	flash.text	3.0
SQLCollationType	flash.data	1.0
SQLColumnNameStyle	flash.data	1.0
SQLColumnSchema	flash.data	1.0
SQLConnection	flash.data	1.0
SQLError	flash.errors	1.0
SQLErrorEvent	flash.events	1.0
SQLErrorOperation	flash.errors	1.0
SQLEvent	flash.events	1.0
SQLIndexSchema	flash.data	1.0

Classe	Pacote ActionScript 3.0	Incluído na versão 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

Classes do Flash Player com funcionalidade específica do AIR

As classes a seguir estão disponíveis para conteúdo do SWF em execução no navegador, mas o AIR fornece métodos ou propriedades adicionais:

Pacote	Classe	Propriedade, método ou evento	Incluído na versão 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	2.6
		listOwnedWindows	2.6
	NativeWindowInitOptions	owner	2.6

Pacote	Classe	Propriedade, método ou evento	Incluído na versão 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	1.0
		responseHeaders	1.0
		responseURL	1.0
	KeyboardEvent	commandKey	1.0
		controlKey	1.0

Pacote	Classe	Propriedade, método ou evento	Incluído na versão AIR
flash.net	FileReference	`extension`	1.0
		Evento httpResponseStatus	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
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

Pacote	Classe	Propriedade, método ou evento	Incluído na versão AIR
flash.system	Capabilities	`idiomas`	1.1
	LoaderContext	`allowLoadBytesCodeExecution`	1.0
	Security	`APPLICATION`	1.0
flash.ui	KeyLocation	D_PAD	2.5

A maioria dessas novas propriedades e métodos estão disponíveis apenas para conteúdo na caixa de proteção de segurança de aplicativos do AIR. No entanto, os novos membros nas classes URLRequest também estão disponíveis para conteúdo em execução em outras caixas de proteção.

Os métodos `ByteArray.compress()` e `ByteArray.uncompress()` incluem, cada um, um novo parâmetro `algorithm`, permitindo que você escolha entre a compactação deflate e zlib. Esse parâmetro só está disponível para conteúdo em execução no AIR.

Componentes Flex específicos do AIR

Os seguintes componentes MX do Adobe® Flex™ estão disponíveis ao desenvolver conteúdo para o Adobe AIR:

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

Além disso, o Flex 4 inclui os seguintes componentes spark do AIR:

- Window
- WindowedApplication

Para mais informações sobre os componentes AIR Flex, consulte Using the Flex AIR components.

Capítulo 4: Ferramentas da Plataforma Adobe Flash para desenvolvimento do AIR

Você pode desenvolver aplicativos AIR com as seguintes ferramentas de desenvolvimento da plataforma Adobe Flash.

Para desenvolvedores em ActionScript 3.0 (Flash e Flex):

- Adobe Flash Professional (consulte Publicações para o AIR)
- SDKs do Adobe Flex 3.x e 4.x (consulte "Definição do SDK do Flex" na página 18 e "AIR Developer Tool (ADT)" na página 168)
- Adobe Flash Builder (consulte, Desenvolvimento de Aplicativos AIR com o Flash Builder)

Para desenvolvedores em HTML e Ajax:

- Adobe AIR SDK (consulte "Instalação do SDK do AIR" na página 16 e "AIR Developer Tool (ADT)" na página 168)
- Adobe Dreamweaver CS3, CS4, CS5 (consulte AIR Extension para Dreamweaver)

Instalação do SDK do AIR

O Adobe AIR SDK contém as seguintes ferramentas de linha de comando usadas para lançar e empacotar aplicativos:

AIR Debug Launcher (ADL) Permite executar aplicativos AIR sem ter que os instalar primeiro. "AIR Debug Launcher (ADL)" na página 162

AIR Development Tool (ADT) Empacota aplicativos AIR em pacotes de instalação prontos para distribuição. Consulte "AIR Developer Tool (ADT)" na página 168.

As ferramentas de linha de comando do AIR requerem Java para ser instaladas em seu computador. Você pode usar a máquina virtual Java a partir do JRE ou do JDK (versão 1.5 ou superior). O Java JRE e o Java JDK estão disponíveis em http://java.sun.com/.

É necessário pelo menos 2GB de memória de computador para executar a ferramenta ADT.

Nota: *O Java não é necessário para os usuários finais executarem aplicativos AIR.*

Para obter uma visão geral sobre a criação de um aplicativo do AIR com o AIR SDK, consulte "Criação do seu primeiro aplicativo do AIR baseado em HTML com o SDK do AIR" na página 31.

Download e instalação do SDK do AIR

Você pode fazer download do AIR SDK e instalá-lo seguindo estas instruções:

Como instalar o AIR SDK no Windows

- Faça download do arquivo de instalação do AIR SDK.
- O AIR SDK é distribuído como um arquivo morto de arquivos padrão. Para instalar o AIR, extraia o conteúdo do SDK para uma pasta no seu computador (por exemplo: C:\Program Files\Adobe\AIRSDK ou C:\AIRSDK).
- As ferramentas ADL e ADT estão contidas na pasta bin no AIR SDK; adicione o caminho para esta pasta a sua variável de ambiente PATH.

Como instalar o AIR SDK no Mac OS X

- Faça download do arquivo de instalação do AIR SDK.
- O AIR SDK é distribuído como um arquivo morto de arquivos padrão. Para instalar o AIR, extraia o conteúdo do SDK para uma pasta no seu computador (por exemplo: Users/<userName>/Aplicativos/AIRSDK).
- As ferramentas ADL e ADT estão contidas na pasta bin no AIR SDK; adicione o caminho para esta pasta a sua variável de ambiente PATH.

Como instalar o AIR SDK no Linux

- O SDK está disponível no formato tbz2.
- Para instalar o SDK, crie uma pasta na qual você deseja descompactar o SDK e, em seguida, use o seguinte comando: tar -jxvf <path to AIR-SDK.tbz2>

Para obter informações sobre como começar a utilização das ferramentas AIR SDK, consulte Como criar um aplicativo do AIR usando ferramentas de linha de comando.

O que está incluído no SDK do AIR

A tabela a seguir descreve o objetivo dos arquivos contidos no AIR SDK:

Pasta SDK	Descrição dos arquivos/ferramentas
bin	O AIR Debug Launcher (ADL) permite executar um aplicativo do AIR sem empacotá-lo e instalá-lo primeiro. Para obter informações sobre como usar esta ferramenta, consulte "AIR Debug Launcher (ADL)" na página 162. O AIR Developer Tool (ADT) empacota seu aplicativo como um arquivo AIR para distribuição. Para obter informações sobre como usar esta ferramenta, consulte "AIR Developer Tool (ADT)" na página 168.
frameworks	O diretório libs contém bibliotecas de código para uso em aplicações AIR. O diretório projects contém o código para as bibliotecas compiladas SWF e SWC.
incluir	O diretório incluso contém o arquivo de cabeçalho da linguagem C- para escrever extensões nativas.
Instalação	O diretório install contém os drivers USB do Windows para dispositivos Android. (Estes são os drivers fornecidos pelo Google no Android SDK).
lib	Contém o código de suporte para as ferramentas do AIR SDK.
tempos de execução	Os tempos de execução AIR para desktop e para dispositivos móveis. O runtime é usado pelo ADL para ativar seus aplicativos AIR antes de serem compactados ou instalados. Os tempos de execução AIR para o Android (pacotes APK) podem ser instalados em aparelhos com Android ou emuladores para desenvolvimento e testes. Pacotes APL separados são usados para dispositivos e emuladores. (O runtime AIR público para o Android está disponível no Android Market.)
amostras	Esta pasta contém uma amostra de arquivo de descrição de aplicativo, uma amostra do recurso de instalação direta (badge.swf) e os ícones padrão do aplicativo do AIR.
modelos	descriptor-template.xml - Um modelo do arquivo de descrição do aplicativo, necessário para cada aplicativo do AIR. Para obter uma descrição detalhada do arquivo do descritor do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208. Os arquivos de esquema para a estrutura XML do descritor do aplicativo para cada versão do AIR também são encontrados nesta pasta.

Definição do SDK do Flex

Para desenvolver aplicativos Adobe® AIR® com o Adobe® Flex™, você tem as seguintes opções:

- Você pode baixar e instalar o Adobe® Flash® Builder™, que fornece ferramentas integradas para criar projetos Adobe AIR e testar, depurar e empacotar seus aplicativos AIR. Consulte "Criando seu primeiro aplicativo desktop Flex AIR no Flash Builder" na página 19
- Você pode baixar o Adobe® Flex™ SDK e desenvolver aplicativos Flex AIR usando seu editor de texto e as ferramentas de linha de comando favoritas.

Para obter uma visão geral sobre a criação de um aplicativo do AIR com o Flex SDK, consulte "Criando seu primeiro aplicativo desktop do AIR com o Flex SDK" na página 35.

Instalação do SDK do Flex

A criação de aplicativos AIR com ferramentas de linha de comando requer que o Java esteja instalado no seu computador. Você pode usar a máquina virtual Java a partir do JRE ou do JDK (versão 1.5 ou superior). O Java JRE e o Java JDK estão disponíveis em http://java.sun.com/.

Nota: *O Java não é necessário para os usuários finais executarem aplicativos AIR.*

O Flex SDK oferece ferramentas de linha de comando e API do AIR que você pode usar para empacotar, compilar e depurar seus aplicativos AIR.

1. Efetue o download do Flex SDK a partir de http://opensource.adobe.com/wiki/display/flexsdk/Downloads.
2. Salve o conteúdo do SDK em uma pasta (por exemplo, Flex SDK).
3. Copie o conteúdo do AIR SDK sobre os arquivos do Flex SDK.

   Nota: *Em computadores Mac, não se esqueça de copiar ou substituir os arquivos individuais nas pastas individuais SDK - não diretórios inteiros. Por padrão, a cópia de um diretório no Mac para um diretório de mesmo nome remove os arquivos existentes no diretório de destino, sem mesclar o conteúdo das duas pastas. Você pode usar o comando* `ditto` *em uma janela de terminal para mesclar o AIR SDK no Flex SDK:*`ditto air_sdk_folder flex_sdk_folder`
4. Os utilitários de linha de comando do AIR estão localizados na pasta bin.

Configuração de SDKs externos

O desenvolvimento de aplicações para o Android e iOS requer que você baixe arquivos de aprovisionamento, SDKs ou outras ferramentas de desenvolvimento dos criadores da plataforma.

Para obter informações sobre como baixar e instalar o Android SDK, consulteDesenvolvedores do Android: instalação do SDK. A partir do AIR 2.6 você não é obrigado a baixar o Android SDK. O SDK do AIR agora inclui os componentes básicos necessários para instalar e lançar pacotes APK. Além disso, o Android SDK pode ser útil para uma variedade de tarefas de desenvolvimento, incluindo a criação e a execução de emuladores de software e de dispositivos para captura de tela.

Não é necessário um SDK para o desenvolvimento do IOS. Todavia, os certificados especiais e perfis de aprovisionamento são necessários. Para obter mais informações, consulte Obtenção dos arquivos de desenvolver da Apple.

Capítulo 5: Criando seu primeiro aplicativo do AIR

Criando seu primeiro aplicativo desktop Flex AIR no Flash Builder

Para ver uma ilustração rápida e prática de como funciona o Adobe® AIR®, use estas instruções para criar e empacotar um aplicativo simples "Hello World" do AIR baseado no arquivo SWF, usando o Adobe® Flash® Builder.

Caso ainda não o tenha feito, efetue o download e instale o Flash Builder. Além disso, baixe e instale a versão mais recente do Adobe AIR, que está localizada aqui: www.adobe.com/go/air_br.

Crie um projeto do AIR

O Flex Builder inclui ferramentas para desenvolver e empacotar aplicativos AIR.

Você começa a criar aplicativos AIR no Flash Builder ou Flex Builder da mesma forma que cria projetos de aplicativo com base no Flex: definindo um novo projeto.

1 Abra o Flash Builder.

2 Selecione Arquivo > Novo > Projeto Flex.

3 Insira o nome do projeto: AIRHelloWorld.

4 No Flex, os aplicativos AIR são considerados um tipo de aplicativo. Você tem duas opções de tipos:

- um aplicativo da web que executa no Adobe® Flash® Player
- um aplicativo de computador pessoal que executa no Adobe AIR

Selecione Desktop como tipo de aplicativo.

5 Clique em Concluir para criar o projeto.

Os projetos do AIR consistem inicialmente em dois arquivos: o arquivo MXML principal e o arquivo XML do aplicativo (conhecido como arquivo de descrição do aplicativo). O último arquivo especifica as propriedades do aplicativo.

Para obter mais informações, consulte Desenvolvimento de aplicativos do AIR com Flash Builder.

Gravação do código do aplicativo do AIR

Para gravar o código do aplicativo "Hello World", você edita o arquivo MXML do aplicativo (AIRHelloWorld.mxml), que é aberto no editor. (Se o arquivo não estiver aberto, use o Navegador de Projeto para abri-lo.)

Os aplicativos Flex AIR no desktop estão contidos na tag WindowedApplication de MXML. A tag WindowedApplication de MXML cria uma janela simples que inclui controles básicos de janela, como barra de título e o botão fechar.

1 Acrescente um atributo `title` ao componente WindowedApplication e atribua-o ao 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 Acrescente um componente Label ao aplicativo (coloque-o dentro da tag WindowedApplication), ajuste a propriedade `text` do componente Label como `"Hello AIR"` e ajuste as restrições de layout para mantê-lo centrado, como mostrado abaixo:

```
<?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 Acrescente o bloco do estilo a seguir imediatamente depois de abrir a tag WindowedApplication e antes da tag de componente de rótulo que você acabou de inserir:

```
<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>
```

Essas configurações de estilo se aplicam a todo o aplicativo, garantindo um plano de fundo da janela um cinza levemente transparente.

O código do aplicativo agora se parece com o seguinte:

```
<?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>
```

Em seguida, você vai alterar algumas configurações no descritor do aplicativo para permitir que o aplicativo seja transparente:

1 No painel de navegação do Flex, localize o arquivo de descrição do aplicativo no diretório de origem do projeto. Se você chamou o seu projeto de AIRHelloWorld, esse arquivo se chama AIRHelloWorld-app.xml.

2 Clique duas vezes no arquivo de descrição do aplicativo para editá-lo no Flash Builder.

3 No código XML, localize as linhas comentadas para as propriedades `systemChrome` e `transparent` (da propriedade `initialWindow`). Retire os comentários. (Retire os delimitadores de comentário `"<!--"` e `"-->"` delimitadores de comentário.)

4 Defina o valor de texto da propriedade `systemChrome` como `none`, como aparece a seguir:

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

5 Defina o valor de texto da propriedade `transparent` como `true`, como aparece a seguir:

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

6 Salve o arquivo.

Teste do aplicativo do AIR

Para testar o código do aplicativo que você escreveu, execute-o no modo de depuração.

1 Clique no botão Depurar na barra de ferramentas principal do

Você também pode selecionar o comando Executar > Depurar > AIRHelloWorld.

O aplicativo do AIR resultante deve ter a aparência do exemplo a seguir:

2 Usando as propriedades `horizontalCenter` e `verrticalCenter` do controle Label, o texto é colocado no centro da janela. Move ou redimensione a janela como você faria em qualquer outro aplicativo do computador.

Nota: *Se o aplicativo não for compilado, conserte a sintaxe ou os erros de digitação que você inseriu de forma inadvertida no código. Os erros e advertências são exibidos na visualização Problemas no Flash Builder.*

Empacotamento, assinatura e execução do aplicativo do AIR

Agora você está pronto para empacotar o aplicativo "Hello World" em um arquivo AIR para distribuição. O arquivo AIR é um arquivo de compactação que contém os arquivos do aplicativo, os quais são todos arquivos contidos na pasta bin do projeto. Neste exemplo simples, esses arquivos são os arquivos SWF e XML do aplicativo. Você distribui o pacote do AIR aos usuários, que então o utilizam para instalar o aplicativo. Uma etapa necessária neste processo é assiná-lo digitalmente.

1. Garanta que o aplicativo não contenha erros de compilação e seja executado como esperado.
2. Selecione Projeto > Exportar versão da compilação.
3. Verifique se o projeto AIRHelloWorld e o aplicativo AIRHelloWorld.mxml estão listados no projeto e no aplicativo.
4. Selecione a opção Exportar como pacote assinado do AIR. Em seguida, clique em Avançar.
5. Se você já tiver um certificado digital, clique em Procurar para localizá-lo e selecioná-lo.
6. Se precisar criar um novo certificado digital autoassinado, selecione Criar.
7. Insira as informações necessárias e clique em OK.
8. Clique em Concluir para gerar o pacote do AIR, que se chama AIRHelloWorld.air.

Agora você pode instalar e executar o aplicativo do Navegador de Projeto no Flash Builder ou no sistema de arquivos clicando duas vezes no arquivo AIR.

Criação do seu primeiro aplicativo do AIR desktop usando o Flash Professional

Para ver uma demonstração rápida e prática de como o Adobe® AIR® funciona, siga as instruções neste tópico para criar e empacotar um aplicativo do AIR “Hello World” simples utilizando o Adobe® Flash® Professional.

Caso ainda não tenha feito isso, baixe e instale o Adobe AIR, que está localizado aqui: www.adobe.com/go/air_br.

Criar aplicativo Hello World no Flash

A criação de um aplicativo Adobe AIR no Flash é semelhante à criação de qualquer outro arquivo FLA. O procedimento a seguir orienta-o no processo de criação de um aplicativo Hello World simples usando o Flash Professional.

Para criar o aplicativo Hello World

1. Inicie o Flash.
2. Na tela de boas-vindas, clique em AIR para criar um arquivo FLA vazio com as configurações de publicação do Adobe AIR.
3. Selecione a ferramenta Texto no painel Ferramentas e crie um campo de texto estático (o padrão) no centro do Palco. Faça-o grande o suficiente para conter de 15 a 20 caracteres.
4. Digite o texto "Hello World" no campo de texto.
5. Salve o arquivo, nomeando-o (por exemplo, HelloAIR).

Testar o aplicativo

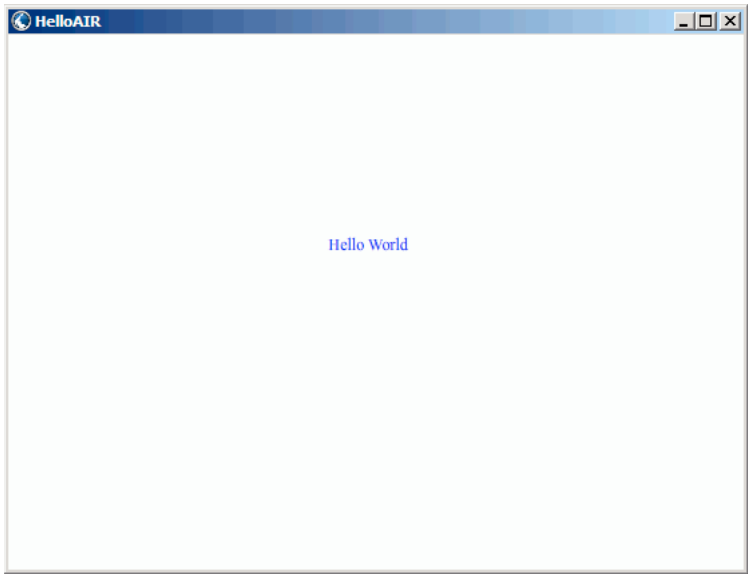
1 Pressione Ctrl + Enter ou selecione Controlar -> Testar filme -> Testar para testar o aplicativo no Adobe AIR.

2 Para utilizar o recurso Depurar filme, primeiro adicione o código ActionScript ao aplicativo. Você pode fazer isso rapidamente adicionando uma instrução trace como esta:

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

3 Pressione Ctrl + Shift + Enter ou selecione Depurar -> Depurar filme -> Depurar para executar o aplicativo com Depurar filme.

O aplicativo Hello World é semelhante a esta ilustração:



Empacotar o aplicativo

1 Selecione Arquivo > Publicar.

2 Assine o pacote Adobe AIR com um certificado digital existentes ou cria um certificado autoassinado passando pelas seguintes etapas:

a Clique no botão Novo próximo ao campo Certificado.

b Preencha as entradas para Nome do editor, Unidade organizacional, Nome da organização, Email, País, Senha e Confirmar senha.

c Especifique o tipo de certificado. A opção Tipo de certificado refere-se ao nível de segurança: 1024-RSA usa uma chave de 1024 bits (menos segura) e 2048-RSA usa uma chave de 2048 bits (mais segura).

d Salve as informações em um arquivo de certificado preenchendo a entrada Salvar como ou clicando no botão Procurar... para procurar o local da pasta. (Por exemplo, *C:/Temp/mycert.pfx*). Quando terminar, clique em OK.

e O Flash retorna para o diálogo Assinatura digital. O caminho e o nome de arquivo do certificado autoassinado que você criou são exibidos na caixa de texto Certificado. Se isso não ocorrer, digite o caminho e o nome de arquivo ou clique no botão Procurar para localizá-lo e selecioná-lo.

f Insira a mesma senha do campo de texto Senha da caixa de diálogo Assinatura Digital como a senha que você assinou na etapa "b". Para mais informações sobre assinatura dos aplicativos Adobe AIR, consulte “Assinatura digital de um arquivo AIR” na página 193.

3 Para criar o arquivo do aplicativo e do programa de instalação, clique no botão Publicar. (No Flash CS4 e CS5, clique no botão Ok.) Execute Testar filme ou Depurar filme para criar os arquivos SWF e application.xml antes de criar o arquivo AIR.

4 Para instalar o aplicativo, clique duas vezes no arquivo AIR (*application*.air) na mesma pasta em que salvou seu aplicativo.

5 Clique no botão Instalar no diálogo Instalação do aplicativo.

6 Revise as Preferências da instalação e as Configurações locais e verifique se a caixa de seleção 'Iniciar aplicativo após a instalação' está marcada. Em seguida, clique em Continuar.

7 Clique em Concluir quando a mensagem Instalação concluída for exibida.

Crie seu primeiro aplicativo do AIR for Android no Flash Professional

Para desenvolver aplicativos do AIR for Android, você deve baixar a extensão do Flash Professional CS5 para o Android a partir de Adobe Labs.

Você também deve baixar e instalar o Android SDK no site do Android, conforme descrito em: Desenvolvedores para Android: Instalação do SDK.

Criar um projeto

1 Abra o Flash Professional CS5

2 Crie um novo projeto do AIR for Android.

A tela inicial do Flash Professional inclui um link para criar um aplicativo do AIR for Android. Você também pode selecionar Arquivo > Novo e, em seguida, selecionar o modelo AIR for Android.

3 Salve o documento como HelloWorld.fla

Escreva o código

Como este tutorial não é realmente sobre como escrever código, basta usar a ferramenta de texto para escrever, "Hello, World!" no palco.

Defina as propriedades do aplicativo

1 Selecione as configurações Arquivo > AIR for Android.

2 Na guia Geral, especifique as seguintes configurações:

- Arquivo de saída: HelloWorld.apk
- Nome do aplicativo: HelloWorld
- ID do aplicativo: HelloWorld
- Número da versão: 0.0.1
- Proporção do aspecto: retrato

3 Na guia Implantação, especifique as seguintes configurações:

- Certificado: Aponte para um certificado de assinatura por código do AIR. Você pode clicar no botão Criar para criar um novo certificado. (Os aplicativos Android implementados através do Android Marketplace devem ter certificados válidos pelo menos até o ano 2033.) Digite a senha do certificado no campo Senha.

- Tipo de implementação do Android: depuração
- Após publicação: selecione ambas as opções
- Digite o caminho para a ferramenta do ADB no subdiretório de ferramentas do Android SDK.

4 Fechar a caixa de diálogo das configurações do Android clicando em OK.

O aplicativo não precisa de ícones ou permissões nesta fase do desenvolvimento. A maioria dos aplicativos AIR for Android exige algumas permissões para acessar recursos protegidos. Você só deve definir as permissões que seu aplicativo realmente exige, pois os usuários podem rejeitar seu aplicativo se ele pedir permissões demais.

5 Salve o arquivo.

Compactar e instalar o aplicativo no dispositivo do Android.

1 Confira se a depuração por USB está ativada no dispositivo. Você pode ativar a depuração por USB nas Configurações do aplicativo em Aplicativos > Desenvolvimento.

2 Conecte seu dispositivo ao computador com um cabo USB.

3 Instale o runtime do AIR. Se você não tiver feito isso, vá para o Android Market e baixe o Adobe AIR. (Você também pode instalar o AIR localmente usando o comando "Comando installRuntime do ADT" na página 180. Pacotes do Android para uso em emuladores e dispositivos Android são incluídos no AIR SDK).

4 Selecione Arquivo > Publicar.

O Flash Professional cria o arquivo APK, instala o aplicativo no dispositivo Android conectado, e ativa-o.

Criação de seu primeiro aplicativo do AIR for iOS

AIR 2.6 ou posterior, iOS 4.2 ou posterior

Você pode codificar, compilar e testar os recursos básicos de um aplicativo do IOS usando apenas ferramentas Adobe. Contudo, para instalar um aplicativo iOS em um dispositivo e distribuí-lo você deve aderir ao programa Apple iOS Developer (que é um serviço pago). Depois de entrar no programa de desenvolvedor do iOS você pode acessar o Portal de aprovisionamento do iOS onde poderá obter os itens a seguir e arquivos da Apple que são necessários para instalar um aplicativo em um dispositivo para testes e para posterior distribuição. Estes itens e arquivos incluem:

- Certificados de distribuição e desenvolvimento
- IDs do aplicativo
- Arquivos de aprovisionamento de desenvolvimento e distribuição

Crie o conteúdo do aplicativo

Crie um arquivo SWF que exibe o texto "Hello world!" Você pode executar essa tarefa usando o Flash Professional, Flash Builder ou outro IDE. Este exemplo simplesmente usa um editor de texto e o compilador SWF de linha de comando incluído no Flex SDK.

1 Crie um diretório em um local conveniente para armazenar seus arquivos de aplicativo. Crie um arquivo chamado *HelloWorld.as* e edite-o no seu editor de código favorito.

2 Adicione o código a seguir:

```
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 a classe usando o compilador amxmlc:

```
amxmlc HelloWorld.as
```

Um arquivo SWF *HelloWorld.swf* é criado na mesma pasta.

Nota: *Este exemplo assume que você configurou a variável de caminho do ambiente para incluir o diretório que contém amxmlc. Para obter mais informações sobre definição de caminho, consulte "Variáveis de ambiente do caminho" na página 308. Alternativamente, você pode digitar o caminho completo para amxmlc e para as outras ferramentas de linha de comando usadas neste exemplo.*

Criar ícone arte e tela inicial arte do aplicativo

Todos os aplicativos para iOS têm ícones que aparecem na interface de usuário do aplicativo iTunes e na tela do dispositivo.

1 Crie um diretório no diretório de seu projeto e nomeie-o "ícones".

2 Crie três arquivos PNG no diretório "ícones". Nomeie os arquivos Icon_29.png, Icon_57.png e Icon_512.png.

3 Edite os arquivos PNG para criar a arte adequada ao seu aplicativo. Os arquivos devem ter 29 por 29 pixels, 57 por 57 pixels e 512 por 512 pixels. Neste teste, use simplesmente quadrados de cor sólida como arte.

Nota: *Ao submeter um aplicativo à Apple App Store, use a versão JPG (não a versão PNG) do arquivo com 512 pixels. Use a versão PNG ao testar as versões de desenvolvimento de um aplicativo.*

Todos os aplicativos para iPhone exibem uma imagem inicial quando o aplicativo carrega no iPhone. Defina a imagem inicial em um arquivo PNG:

1 No diretório principal de desenvolvimento, crie um arquivo PNG com o nome Default.png. (*Não* coloque esse arquivo no subdiretório "ícones": Certifique-se de nomear o arquivo Default.png com D maiúsculo).

2 Edite o arquivo para 320 pixels de largura e 480 pixels de altura. Por enquanto, o conteúdo pode ser retângulo branco liso. (Você mudará isso mais tarde).

Para informações detalhadas sobre estes gráficos, consulte "Ícones de aplicativos" na página 88.

Crie o arquivo descritor do aplicativo

Crie um arquivo de descritor do aplicativo que especifique as propriedades básicas para a aplicação. Você pode concluir esta tarefa utilizando um IDE como o Flash Builder ou um editor de texto.

1 Na pasta do projeto que contém HelloWorld.as, crie um arquivo XML chamado *HelloWorld-app.xml*. Edite este arquivo no seu editor de XML favorito.

2 Adicione o XML a seguir:

```
<?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 razões de simplicidade, este exemplo define apenas algumas das propriedades disponíveis.

Nota: *Se você estiver utilizando o AIR 2 ou anterior, deverá utilizar o elemento* `<version>` *em vez do elemento* `<versionNumber>`.

3 Altere o ID do aplicativo para coincidir com o ID do aplicativo especificado no Portal de Aprovisionamento do iOS. (Não inclua a parte da distribuição do conjunto aleatório no início do ID).

4 Teste o aplicativo usando o ADL:

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

O ADL deve abrir uma janela no seu desktop que exibe o texto: *Hello World!*Se isto não acontecer, procure por erros no código-fonte e no descritor do aplicativo.

Compile o arquivo IPA

Agora é possível compilar o arquivo do instalador do IPA usando o ADT. Você deve ter o certificado de desenvolvedor da Apple e a chave privada em formato de arquivo P12, além do perfil de aprovisionamento de desenvolvimento da Apple.

Execute o utilitário do ADT com as opções a seguir, substituindo o armazenamento de chaves, a senha do armazenamento, e os valores de perfil por aprovisionamento pelos seus próprios:

```
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
```

(Utilize uma única linha de comando; as quebras de linha neste exemplo são apenas para torná-lo mais fácil de ler.)

ADT gera o arquivo de instalação do aplicativo do iOS, *HelloWorld.ipa*, no diretório do projeto. A compilação do arquivo IPA pode demorar alguns minutos.

Instale o aplicativo em um dispositivo

Para instalar o aplicativo do iOS para testes:

1. Abra o aplicativo iTunes.
2. Se você ainda não fez isso, adicione o perfil de aprovisionamento desse aplicativo ao iTunes. No iTunes, selecione Arquivo > Adicionar à Biblioteca. Selecione o arquivo de perfil de aprovisionamento (que tem mobileprovision como tipo de arquivo).

   Por enquanto, para testar o aplicativo em seu dispositivo de desenvolvedor, use o perfil de aprovisionamento de desenvolvimento.

   Mais tarde, ao distribuir o aplicativo para a iTunes Store, use o perfil de distribuição. Para distribuir o aplicativo ad hoc (para múltiplos dispositivos sem passar pela iTunes Store), use o arquivo de aprovisionamento ad hoc.

   Para obter mais informações sobre perfis de aprovisionamento, consulte "Configuração do iOS" na página 66.
3. Algumas versões do iTunes não substituem o aplicativo, se a mesma versão do aplicativo estiver instalada. Nesse caso, exclua o aplicativo do seu dispositivo e da lista de aplicativos no iTunes.
4. Clique duas vezes no arquivo IPA do aplicativo. O arquivo deve aparecer na lista de aplicativos no iTunes.
5. Conecte seu dispositivo à porta USB em seu computador.
6. No iTunes, selecione a guia Aplicativos do dispositivo e certifique-se de que o aplicativo está selecionado na lista de aplicativos para instalação.
7. Selecione o dispositivo na lista à esquerda do aplicativo iTunes. Em seguida, clique no botão Sync. Quando a sincronização terminar, o aplicativo Hello World aparecerá em seu iPhone.

Se a nova versão não estiver instalada, exclua-a do dispositivo e da lista de aplicativos no iTunes e, em seguida, execute novamente esse procedimento. Talvez esse seja o caso, se a versão instalada atualmente usa o mesmo ID do aplicativo e versão.

Editar o gráfico da tela inicial

Antes de compilar o aplicativo, você criou o arquivo Default.png (consulte "Criar ícone arte e tela inicial arte do aplicativo" na página 26). Esse arquivo PNG atua como a imagem de inicialização quando o aplicativo é carregado. Quando você testou o aplicativo em seu iPhone, talvez tenha observado a tela branca na inicialização.

Altere essa imagem para coincidir com a tela de inicialização de seu aplicativo ("Hello World!"):

1. Abra o aplicativo em seu dispositivo. Quando o primeiro texto "Hello World" aparecer, pressione e mantenha pressionado o botão Início (abaixo da tela). Mantendo pressionado o botão Início, pressione o botão Despertar/Repousar (na parte superior do iPhone). Isso executa uma captura de tela e envia-a para o rolo da câmera.
2. Transfira a imagem para seu computador de desenvolvimento, transferindo as fotos do iPhoto ou de outro aplicativo de transferência de fotos. (No Mac OS, é possível também usar o aplicativo Captura de Imagem).

   É possível também enviar a foto por email para seu computador de desenvolvimento:

   - Abra o aplicativo Fotos.
   - Abra Rolo da câmera.
   - Abra a imagem da captura de tela que capturou.

- Pressione a imagem e, em seguida, pressione o botão (seta) “Avançar” no canto inferior esquerdo. Em seguida, clique no botão Enviar foto por email e envie a imagem para você mesmo.

3 Substitua o arquivo Default.png (no seu diretório de desenvolvimento) pela versão PNG da imagem da captura de tela.

4 Compile novamente o aplicativo (consulte “Compile o arquivo IPA” na página 27) e reinstale-o em seu dispositivo.

O aplicativo usa a nova tela de inicialização ao carregar.

Nota: *É possível criar qualquer arte que deseja para o arquivo Default.png, desde que tenha as dimensões corretas (320 x 480 pixels). No entanto, frequentemente é melhor que a imagem do arquivo Default.png coincida com o estado inicial do aplicativo.*

Criação do primeiro aplicativo do AIR baseado em HTML com o Dreamweaver

Para uma ver uma ilustração rápida e prática de como funciona o Adobe® AIR®, use essas instruções para criar e empacotar um aplicativo simples "Hello World" do AIR baseado em HTML usando o Adobe® AIR® Extension for Dreamweaver®.

Caso ainda não tenha feito isso, baixe e instale o Adobe AIR, que está localizado aqui: www.adobe.com/go/air_br.

Para obter instruções sobre a instalação do Adobe AIR Extension para Dreamweaver, consulte Instale o Adobe AIR Extension para Dreamweaver.

Para obter uma visão geral da extensão, incluindo os requisitos do sistema, consulte AIR Extension para Dreamweaver.

Nota: *Os aplicativos do AIR baseados em HTML só podem ser desenvolvidos para os perfis desktop e extendedDesktop. O perfil móvel não é compatível.*

Preparação dos arquivos do aplicativo

Seu aplicativo do Adobe AIR deve ter uma página inicial e todas as suas páginas relacionadas definidas em um site do Dreamweaver:

1 Inicie o Dreamweaver e certifique-se de ter um site definido.

2 Abra uma nova página HTML selecionando Arquivo > Novo, depois HTML na coluna Tipo de página, em seguida Nenhum na coluna Layout e clicando em Criar.

3 Na nova página, digite **Hello World!**

Este exemplo é extremamente simples, mas, se desejar, você pode aplicar estilo ao texto ao seu gosto, adicionar mais conteúdo à página, vincular outras páginas a essas página inicial e assim por diante.

4 Salve a página (Arquivo > Salvar) como hello_world.html. Certifique-se de salvar o arquivo em um site Dreamweaver.

Para obter mais informações sobre os sites Dreamweaver, consulte a Ajuda do Dreamweaver.

Criação do aplicativo Adobe AIR

1 Não se esqueça de abrir a página hello_world.html na janela do documento do Dreamweaver. (Consulte a seção anterior para obter instruções sobre sua criação.)

2 Selecione Site > Configurações do aplicativo do AIR.

A maioria das configurações exigidas na caixa de diálogo Configurações e aplicativo do AIR são preenchidas automaticamente para você. Você deve, no entanto, selecionar o conteúdo inicial (ou a página inicial) do seu aplicativo.

3 Clique no botão Procurar perto da opção Conteúdo inicial, vá até sua página hello_world.html e selecione-a.

4 Perto da opção Assinatura digital, clique no botão Definir.

A assinatura digital fornece a garantia de que o código para um aplicativo não foi alterado nem está corrompido desde sua criação pelo autor do software, sendo exigido em todos os aplicativos Adobe AIR.

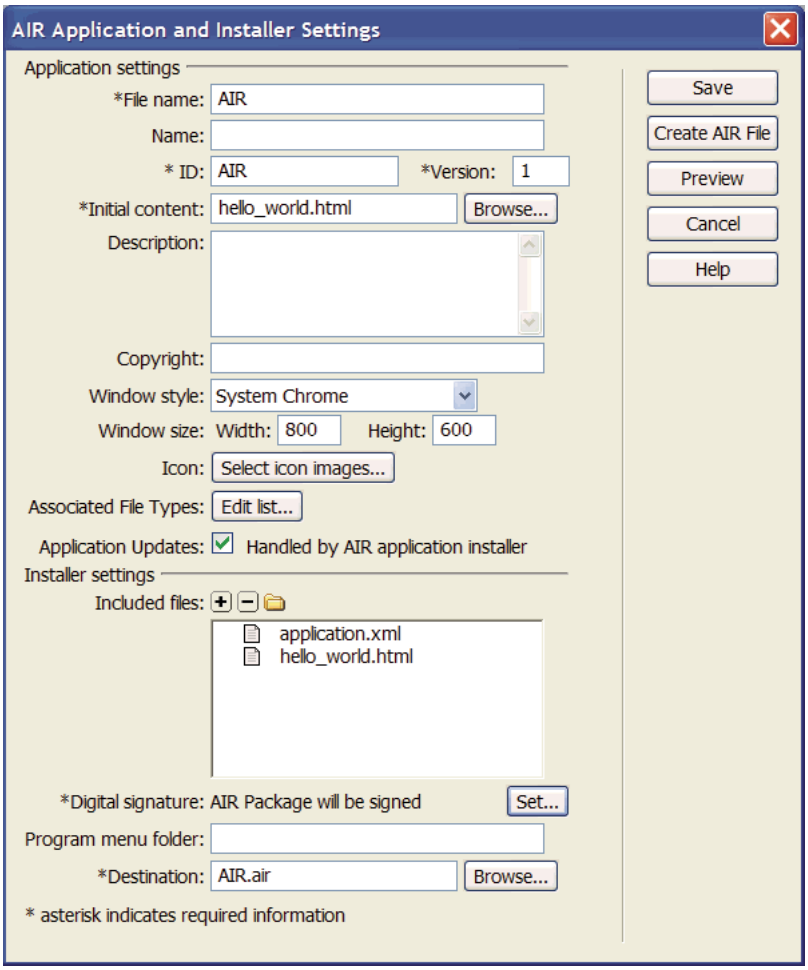
5 Na caixa de diálogo Assinatura digital, selecione Assinar o pacote AIR com um certificado digital, e clique no botão Criar. (Se você já tiver acesso a um certificado digital, poderá clicar no botão Procurar para selecioná-lo.)

6 Preencha os campos necessários na caixa de diálogo Certificado digital autoassinado. Você vai precisar inserir seu nome e uma senha e depois confirmá-la, e inserir um nome para o arquivo de certificado digital. O Dreamweaver salva o certificado digital na raiz do site.

7 Clique em OK para voltar para a caixa de diálogo Assinatura digital.

8 Na caixa de diálogo Assinatura digital, insira a senha especificada para seu certificado digital e clique em OK.

A caixa de diálogo completa do aplicativo do AIR e das configurações do instalador talvez se pareçam com o seguinte:



Para obter mais informações sobre todas as opções de caixa de diálogo e como editá-as, consulte Criação de um aplicativo do AIR no Dreamweaver.

9 Clique no botão Criar arquivo AIR.

O Dreamweaver cria o arquivo do aplicativo Adobe AIR e salva-o na pasta raiz do seu site. O Dreamweaver também cria um arquivo application.xml e o salva no mesmo lugar. Este arquivo serve como manifesto, definido várias propriedades do aplicativo.

Instalação do aplicativo em um computador

Agora que você criou o arquivo de aplicativo, poderá instalá-lo em qualquer computador.

1 Retire o arquivo do aplicativo do Adobe AIR do seu site Dreamweaver e para dentro do seu computador, ou para outro computador.

Essa etapa é opcional. Você pode, na verdade, instalar o novo aplicativo no seu computador direto do diretório do site do Dreamweaver, se preferir.

2 Clique duas vezes no arquivo executável do aplicativo (arquivo .air) para instalar o aplicativo.

Pré-visualização do aplicativo Adobe AIR

Você pode pré-visualizar as páginas que farão parte dos aplicativos do AIR a qualquer momento. Ou seja, você não precisa necessariamente empacotar o aplicativo antes de ver como ele ficará depois de instalado.

1 Certifique-se de que sua página hello_world.html está aberta na janela de documentos do Dreamweaver.

2 Na barra de ferramentas do documento, clique no botão Visualizar/depurar no navegador, e depois selecione Visualizar no AIR.

Você também pode pressionar Ctrl+Shift+F12 (Windows) ou Cmd+Shift+F12 (Macintosh).

Ao visualizar essa página, você está vendo essencialmente o que um usuário veria como página inicial do aplicativo depois de ter instalado o aplicativo em um computador.

Criação do seu primeiro aplicativo do AIR baseado em HTML com o SDK do AIR

Para uma ver uma ilustração rápida e prática de como funciona o Adobe® AIR®, use essas instruções para criar e empacotar um aplicativo simples "Hello World" do AIR baseado em HTML.

Para começar, você deve ter instalado o runtime e configurar o SDK do AIR. Você via usar o *AIR Debug Launcher* (ADL) e a *AIR Developer Tool* (ADT) neste tutorial. O ADL e o ADT são programas utilitários de linha de comando e podem ser encontrados no diretório `bin` do SDK do AIR (consulte "Instalação do SDK do AIR" na página 16). Este tutorial parte do princípio de que você já está familiarizado com os programas em execução da linha de comando e sabe como configurar as variáveis do ambiente do caminho necessário para o seu sistema operacional.

Nota: *Se você for um usuário do Adobe® Dreamweaver®, leia "Criação do primeiro aplicativo do AIR baseado em HTML com o Dreamweaver" na página 29.*

Nota: *Os aplicativos do AIR baseados em HTML só podem ser desenvolvidos para os perfis desktop e extendedDesktop. O perfil móvel não é compatível.*

Criar os arquivos do projeto

Cada projeto do AIR baseado em HTML deve conter os dois arquivos a seguir: um arquivo descritor do aplicativo, que especifica os metadados do aplicativo, e uma página HTML de nível superior. Além desses arquivos necessários, este projeto inclui um arquivo de código em JavaScript, `AIRAliases.js`, que define as variáveis alias para as classes API do AIR.

1. Crie um diretório chamado `HelloWorld` para conter os arquivos de projeto.
2. Crie um arquivo XML, chamado `HelloWorld-app.xml`.
3. Crie um arquivo HTML chamado `HelloWorld.html`.
4. Copie `AIRAliases.js` da pasta de estruturas do SDK do AIR no diretório do projeto.

Criar o arquivo descritor do aplicativo do AIR

Para começar a desenvolver seu aplicativo do AIR, crie um arquivo descritor do aplicativo XML com a seguinte estrutura:

```
<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 edição.
2. Acrescente o elemento raiz `<application>`, incluindo o atributo de espaço de nome do AIR:

   <application xmlns="http://ns.adobe.com/air/application/2.7"> O último segmento do namespace, "2.7", especifica a versão do runtime requerida pelo aplicativo.

3. Acrescente o elemento `<id>`:

   <id>examples.html.HelloWorld</id> O id do aplicativo identifica exclusivamente seu aplicativo junto com o ID de editor (que o AIR cria do certificado usado para assinar o pacote do aplicativo). O ID de aplicativo é usada para instalação, acesso ao diretório privado de armazenamento do sistema de arquivos, acesso ao armazenamento criptografado privado e comunicação entre aplicativos.

4. Acrescente o elemento `<versionNumber>`:

   <versionNumber>0.1</versionNumber> Ajuda os usuários a determinar qual versão do aplicativo eles estão instalando.

   Nota: *Se você estiver utilizando o AIR 2 ou anterior, deverá utilizar o elemento* `<version>` *em vez do elemento* `<versionNumber>`*.*

5. Acrescente o elemento `<filename>`:

   <filename>HelloWorld</filename> O nome usado para o executável do aplicativo, diretório de instalação e outras referências ao aplicativo no sistema operacional.

6. Acrescente o elemento `<initialWindow>` que contém os seguintes elementos filho para especificar as propriedades da janela inicial do seu aplicativo:

<content>HelloWorld.html</content> Identifica o arquivo HTML raiz que o AIR deve carregar.

<visible>true</visible> Torna a janela visível imediatamente.

<width>400</width> Ajusta a largura da janela (em pixels).

<height>200</height> Ajusta a altura da janela.

7 Salve o arquivo. O arquivo descritor completo do aplicativo deve se parecer com o seguinte:

```
<?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>
```

Este exemplo só define algumas propriedades possíveis do aplicativo. Para ver o conjunto completo de propriedades do aplicativo, que lhe permitem especificar coisas como cromo e tamanho da janela, transparência, diretório padrão de instalação, tipos de arquivo associados e ícones do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208.

Criar a página HTML do aplicativo

Agora você precisa criar uma página HTML simples que serve como arquivo principal para o aplicativo do AIR.

1 Abra o arquivo `HelloWorld.html` para edição. Acrescente o seguinte código HTML:

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

2 Na seção `<head>` do HTML, importe o arquivo `AIRAliases.js`:

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

O AIR define uma propriedade chamada `runtime` no objeto da janela HTML. A propriedade de runtime fornece acesso às classes incorporadas do AIR, usando o nome do pacote totalmente qualificado da classe. Por exemplo, para criar um objeto File do AIR, você poderia acrescentar a instrução seguinte em JavaScript:

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

O arquivo `AIRAliases.js` define os aliases convenientes para as APIs mais úteis do AIR. Usando o `AIRAliases.js`, você poderia encurtar a referência ao arquivo File para o seguinte:

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

3 Abaixo da tag de script AIRAliases, acrescente outra tag de script contendo uma função de JavaScript para lidar com o evento `onLoad`:

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

A função `appLoad()` simplesmente chama a função `air.trace()`. A mensagem do traçado é impressa no console de comando quando você executa o aplicativo usando o ADL. As instruções de traçado podem ser muito úteis para depuração.

4 Salve o arquivo.

Seu arquivo `HelloWorld.html` deve se parecer com o seguinte:

```
<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>
```

Testar o aplicativo

Para executar e testar o aplicativo da linha de comando, use o utilitário AIR Debug Launcher (ADL). O executável ADL pode ser encontrado no diretório `bin` do SDK do AIR. Se você ainda não configurou o SDK do AIR, consulte "Instalação do SDK do AIR" na página 16.

1 Abra um console ou shell de comando. Troque para o diretório que você criou para este projeto.

2 Execute o seguinte comando:

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

É aberta uma janela do AIR, exibindo seu aplicativo. Além disso, a janela de console exibe a mensagem que resulta da chamada `air.trace()`.

Para mais informações, consulte "Arquivos descritores do aplicativo do AIR" na página 208.

Criar o arquivo de instalação do AIR

Quando seu aplicativo é executado com sucesso, você pode usar o utilitário ADT para empacotar o aplicativo em um arquivo de instalação do AIR. Um arquivo de instalação do AIR é um arquivo de compactação que contém todos os arquivos do aplicativo, que você pode distribuir para seus usuários. Você deve instalar o Adobe AIR antes de instalar um arquivo AIR empacotado.

Para garantir a segurança do aplicativo, todos os arquivos de instalação do AIR devem ser digitalmente assinados. Para fins de desenvolvimento, você pode gerar um certificado básico autoassinado com o ADT ou outra ferramenta de geração de certificados. Também é possível comprar um certificado comercial com assinatura de código de uma autoridade de certificado comercial como a VeriSign ou a Thawte. Quando os usuários instalam um arquivo AIR autoassinado, o editor é exibido como "desconhecido" durante o processo de instalação. Isso ocorre porque um

certificado autoassinado só garante que o arquivo AIR não tenha sido alterado desde sua criação. Não há nada que impeça alguém de autoassinar um arquivo AIR disfarçado e apresentá-lo como seu aplicativo. Para os arquivos AIR liberados ao público, um certificado comercial verificável é altamente recomendado. Para uma visão geral de questões de segurança do AIR, consulte Segurança do AIR (para desenvolvedores em ActionScript) ou AIR security (para desenvolvedores em HTML).

Gerar um certificado autoassinado e par de chaves

❖ No prompt de comando, insira o comando a seguir (o executável ADT está no diretório `bin` do SDK do AIR):

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

O ADT gera um arquivo de keystore chamado *sampleCert.pfx* que contém um certificado e uma chave privada relacionada.

Este exemplo usa o número mínimo de atributos que podem ser definidos para um certificado. O tipo de chave deve ser *1024-RSA* ou *2048-RSA* (consulte "Assinatura de aplicativos AIR" na página 193).

Criar o arquivo de instalação do AIR

❖ No prompt de comando, digite o seguinte comando (em uma única linha):

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

Você terá que fornecer a senha do arquivo do keystore.

O argumento HelloWorld.air é o arquivo AIR que o ADT produz. HelloWorld-app.xml é o arquivo descritor do aplicativo. Os argumentos subsequentes são os arquivos usados pelo seu aplicativo. Esse exemplo só usa dois arquivos, mas você pode incluir qualquer número de arquivos e diretórios. O ADT verifica se o principal arquivo de conteúdo (HelloWorld.html) está incluído no pacote, mas se você esquecer de incluir AIRAliases.js, seu aplicativo simplesmente não funcionará.

Depois da criação do pacote AIR, você pode instalar e executar o aplicativo clicando duas vezes no arquivo de pacote Você também pode digitar o nome do arquivo AIR como comando em um shell ou janela de comando.

Próximas etapas

No AIR, o código de HTML e JavaScript geralmente se comporta da mesma forma que se comportaria em um navegador típico. (Na verdade, o AIR usa o mesmo mecanismo de renderização WebKit usado pelo navegador da Web Safari.) No entanto, há algumas diferenças importantes que você deve entender ao desenvolver aplicativos HTML no AIR. Para mais informações sobre estas diferenças e outros tópicos importantes, consulte Programming HTML and JavaScript.

Criando seu primeiro aplicativo desktop do AIR com o Flex SDK

Para obter ilustrações práticas e rápidas sobre o funcionamento do Adobe® AIR®, use estas instruções para criar um simples aplicativo do AIR com base em SWF "Hello World" utilizando o Flex SDK. Este tutorial mostra como compilar, testar e empacotar um aplicativo do AIR com as ferramentas de linha de comando fornecidas com o Flex SDK (o Flex SDK inclui o AIR SDK).

Para começar, você deve ter instalado o runtime e configurar o Adobe® Flex™. Este tutorial utiliza o compilador *AMXMLC*, o *AIR Debug Launcher* (ADL) e a *AIR Developer Tool* (ADT). Estes programas podem ser encontrados no diretório `bin` do Flex SDK (consulte "Definição do SDK do Flex" na página 18).

Criar o arquivo descritor do aplicativo do AIR

Esta seção descreve como criar o descritor do aplicativo, que é um arquivo XML com a estrutura a seguir:

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

1 Crie um arquivo XML com o nome `HelloWorld-app.xml` e salve-o no diretório do projeto.

2 Acrescente o elemento `<application>`, incluindo o atributo de espaço de nome do AIR:

<application xmlns="http://ns.adobe.com/air/application/2.7"> O último segmento do espaço de nome, "2.7", especifica a versão do runtime de que o aplicativo necessita.

3 Acrescente o elemento `<id>`:

<id>samples.flex.HelloWorld</id> O id do aplicativo identifica exclusivamente seu aplicativo junto com o ID de editor (que o AIR cria do certificado usado para assinar o pacote do aplicativo). A forma recomendado é uma sequência de caractere separados por pontos, do tipo DNS reverso, como, por exemplo, `"com.company.AppName"`. O ID de aplicativo é usada para instalação, acesso ao diretório privado de armazenamento do sistema de arquivos, acesso ao armazenamento criptografado privado e comunicação entre aplicativos.

4 Acrescente o elemento `<versionNumber>`:

<versionNumber>1.0</versionNumber> Ajuda os usuários a determinar qual versão do aplicativo eles estão instalando.

Nota: *Se você estiver utilizando o AIR 2 ou anterior, deverá utilizar o elemento `<version>` em vez do elemento `<versionNumber>`.*

5 Acrescente o elemento `<filename>`:

<filename>HelloWorld</filename> O nome usado para o executável do aplicativo, diretório de instalação e outras referências ao aplicativo no sistema operacional.

6 Acrescente o elemento `<initialWindow>` que contém os seguintes elementos filho para especificar as propriedades da janela inicial do seu aplicativo:

<content>HelloWorld.swf</content> Identifica o arquivo SWF de raiz para o AIR carregar.

<visible>true</visible> Torna a janela visível imediatamente.

<width>400</width> Ajusta a largura da janela (em pixels).

<height>200</height> Ajusta a altura da janela.

7 Salve o arquivo. O arquivo descritor de aplicativo completo deve assemelhar-se ao seguinte:

```
<?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>
```

Este exemplo só define algumas propriedades possíveis do aplicativo. Para ver o conjunto completo de propriedades do aplicativo, que lhe permitem especificar coisas como cromo e tamanho da janela, transparência, diretório padrão de instalação, tipos de arquivo associados e ícones do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208.

Gravação do código do aplicativo

Nota: *Aplicativos com SWF com base no AIR podem utilizar a classe principal definida no MXML ou com o Adobe® ActionScript® 3.0. Este exemplo utiliza o arquivo MXML para definir a classe principal. O processo de criação de um aplicativo do AIR com a classe principal ActionScript é similar. Em vez de compilar um arquivo MXML em um arquivo SWF, compile o arquivo da classe ActionScript. Ao utilizar ActionScript, a classe principal deve estender-se a flash.display.Sprite.*

Como todos os aplicativos com base em Flex, os aplicativos AIR criados com o Flex contém o arquivo principal MXML. Aplicativos desktop do AIR usam o componente *WindowedApplication* como elemento raiz, em vez do componente Application. O componente WindowedApplication fornece propriedades, métodos e eventos para controlar seu aplicativo e a janela inicial. Em plataformas e perfis para os quais o AIR não oferece suporte a várias janelas, continue a utilizar o componente Application. Nos aplicativos Flex móveis, você também pode usar o componente Visualização ou o componente TabbedViewNavigatorApplication.

O procedimento a seguir cria o aplicativo Hello World:

1 Utilizando um editor de texto, crie um arquivo chamado `HelloWorld.mxml` e adicione o código MXML a seguir:

```
<?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 Em seguida, adicione um componente Label ao aplicativo (insira-o na tag WindowedApplication).

3 Defina propriedade `text` do componente Label para `"Hello AIR"`.

4 Defina as restrições de layout para sempre manter centralizado.

O exemplo a seguir mostra o código até agora:

```
<?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>
```

Compile o aplicativo

Antes de executar e depurar o aplicativo, compile o código MXML em um arquivo SWF utilizando o compilador amxmlc. O compilador amxmlc pode localizar o diretório `bin` do Flex SDK. Se preferir, você pode definir a variável de ambiente do caminho de seu computador para incluir o diretório bin o Flex SDK. A configuração do caminho facilita a execução dos utilitários na linha de comando.

1 Abra um prompt de comando ou um terminal e navegue até a pasta de projeto do aplicativo do AIR.

2 Digite o seguinte comando:

```
amxmlc HelloWorld.mxml
```

O compilador `amxmlc` cria o arquivo`HelloWorld.swf`, que contém o código compilado do aplicativo.

Nota: *Se o aplicativo não compilar, corrija erros de sintaxe ou ortografia. Erros e avisos serão exibidos na janela do console utilizado para executar o compilador amxmlc.*

Para obter mais informações, consulte "Compilação de arquivos de origem do MXML e ActionScript para AIR" na página 158.

Testar o aplicativo

Para executar e testar o aplicativo a partir da linha de comando, utilize o AIR Debug Launcher (ADL) para executar o aplicativo utilizando o arquivo do descritor do aplicativo. (O ADL pode ser encontrado no diretório bin do Flex SDK.)

❖ No prompt de comando, insira o comando a seguir:

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

O aplicativo do AIR resultante é similar à figura a seguir:

Usando as propriedades horizontalCenter e verrticalCenter do controle Label, o texto é colocado no centro da janela. Move ou redimensione a janela como você faria em qualquer outro aplicativo do computador.

Para obter mais informações, consulte "AIR Debug Launcher (ADL)" na página 162.

Criar o arquivo de instalação do AIR

Quando seu aplicativo é executado com sucesso, você pode usar o utilitário ADT para empacotar o aplicativo em um arquivo de instalação do AIR. Um arquivo de instalação do AIR é um arquivo de compactação que contém todos os arquivos do aplicativo, que você pode distribuir para seus usuários. Você deve instalar o Adobe AIR antes de instalar um arquivo AIR empacotado.

Para garantir a segurança do aplicativo, todos os arquivos de instalação do AIR devem ser digitalmente assinados. Para fins de desenvolvimento, você pode gerar um certificado básico autoassinado com o ADT ou outra ferramenta de geração de certificados. Você também pode adquirir um certificado de assinatura de código comercial de uma autoridade de certificação. Quando os usuários instalam um arquivo AIR autoassinado, o editor é exibido como "desconhecido" durante o processo de instalação. Isso ocorre porque um certificado autoassinado só garante que o arquivo AIR não tenha sido alterado desde sua criação. Não há nada que impeça alguém de autoassinar um arquivo AIR disfarçado e apresentá-lo como seu aplicativo. Para os arquivos AIR liberados ao público, um certificado comercial verificável é altamente recomendado. Para uma visão geral de questões de segurança do AIR, consulte Segurança do AIR (para desenvolvedores em ActionScript) ou AIR security (para desenvolvedores em HTML).

Gerar um certificado autoassinado e par de chaves

❖ No prompt de comando, insira o comando a seguir (o executável ADT está no diretório `bin` do SDK do Flex):

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

Este exemplo usa o número mínimo de atributos que podem ser definidos para um certificado. O tipo de chave deve ser *1024-RSA* ou *2048-RSA* (consulte “Assinatura de aplicativos AIR” na página 193).

Crie o pacote do AIR

❖ No prompt de comando, digite o seguinte comando (em uma única linha):

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

Você terá que fornecer a senha do arquivo do keystore. Digite a senha e pressione Enter. Os caracteres da senha não são exibidos por motivos de segurança.

O argumento HelloWorld.air é o arquivo AIR que o ADT produz. HelloWorld-app.xml é o arquivo descritor do aplicativo. Os argumentos subsequentes são os arquivos usados pelo seu aplicativo. Esse exemplo só usa três arquivos, mas você pode incluir qualquer número de arquivos e diretórios.

Depois da criação do pacote AIR, você pode instalar e executar o aplicativo clicando duas vezes no arquivo de pacote Você também pode digitar o nome do arquivo AIR como comando em um shell ou janela de comando.

Para obter mais informações, consulte “Compactação de um arquivo de instalação AIR desktop.” na página 52.

Criando seu primeiro aplicativo do AIR for Android com o Flex SDK

Para começar, você deve ter instalado e configurado o AIR e o Flex SDKs. Este tutorial usa o compilador *AMXMLC* do Flex SDK, o *AIR Debug Launcher* (ADL) e o *AIR Developer Tool* (ADT) do AIR SDK. “Definição do SDK do Flex” na página 18.

Você também deve baixar e instalar o Android SDK no site do Android, conforme descrito em: Desenvolvedores para Android: Instalação do SDK.

Nota: *Para mais informações sobre desenvolvimento para iPhone, consulte Criação do aplicativo Hello World para iPhone com o Flash Professional CS5.*

Criar o arquivo descritor do aplicativo do AIR

Esta seção descreve como criar o descritor do aplicativo, que é um arquivo XML com a estrutura a seguir:

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

1. Crie um arquivo XML com o nome `HelloWorld-app.xml` e salve-o no diretório do projeto.
2. Acrescente o elemento `<application>`, incluindo o atributo de espaço de nome do AIR:

   <application xmlns="http://ns.adobe.com/air/application/2.7"> O último segmento do espaço de nome, "2.7", especifica a versão do runtime de que o aplicativo necessita.
3. Acrescente o elemento `<id>`:

   <id>samples.android.HelloWorld</id> O id do aplicativo identifica exclusivamente seu aplicativo junto com o ID de editor (que o AIR cria do certificado usado para assinar o pacote do aplicativo). O formulário recomendado é uma sequência de caracteres "reverse-DNS-style", delimitada por pontos, como `"com.company.AppName"`.
4. Acrescente o elemento `<versionNumber>`:

   <versionNumber>0.0.1</versionNumber> Ajuda os usuários a determinar qual versão do aplicativo eles estão instalando.
5. Acrescente o elemento `<filename>`:

   <filename>HelloWorld</filename> O nome usado para o executável do aplicativo, diretório de instalação e outras referências ao aplicativo no sistema operacional.
6. Acrescente o elemento `<initialWindow>` que contém os seguintes elementos filho para especificar as propriedades da janela inicial do seu aplicativo:

   <content>HelloWorld.swf</content> Identifica o arquivo HTML raiz que o AIR deve carregar.
7. Acrescente o elemento `<supportedProfiles>`.

   <supportedProfiles>mobileDevice</supportedProfiles> Especifica que o aplicativo só funciona no perfil móvel.
8. Salve o arquivo. O arquivo descritor de aplicativo completo deve assemelhar-se ao seguinte:

```
<?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>
```

Este exemplo só define algumas propriedades possíveis do aplicativo. Existem outras configurações que você pode usar no arquivo de descritor do aplicativo. Por exemplo, você pode adicionar <fullScreen>true</fullScreen> para o elemento initialWindow para compilar um aplicativo de tela cheia. Para ativar a depuração remota e recursos controlados por acesso no Android, você também terá que adicionar as permissões Android para o descritor do aplicativo. As permissões não são necessárias para este aplicativo simples, por isso você não precisa adicioná-las agora.

Para obter mais informações, consulte "Configuração de propriedades do aplicativo móvel" na página 71.

Gravação do código do aplicativo

Crie um arquivo chamado HelloWorld.as e adicione o seguinte código utilizando um 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 );
        }
    }
}
```

Compile o aplicativo

Antes de executar e depurar o aplicativo, compile o código MXML em um arquivo SWF utilizando o compilador amxmlc. O compilador amxmlc pode localizar o diretório `bin` do Flex SDK. Se preferir, você pode definir a variável de ambiente do caminho de seu computador para incluir o diretório bin o Flex SDK. A configuração do caminho facilita a execução dos utilitários na linha de comando.

1. Abra um prompt de comando ou um terminal e navegue até a pasta de projeto do aplicativo do AIR.
2. Digite o seguinte comando:

   ```
   amxmlc HelloWorld.as
   ```

O compilador `amxmlc` cria o arquivo`HelloWorld.swf`, que contém o código compilado do aplicativo.

Nota: *Se o aplicativo não compilar, corrija erros de sintaxe ou ortografia. Erros e avisos serão exibidos na janela do console utilizado para executar o compilador amxmlc.*

Para obter mais informações, consulte "Compilação de arquivos de origem do MXML e ActionScript para AIR" na página 158.

Testar o aplicativo

Para executar e testar o aplicativo a partir da linha de comando, utilize o AIR Debug Launcher (ADL) para executar o aplicativo utilizando o arquivo do descritor do aplicativo. (O ADL pode ser encontrado no diretório bin do Air e Flex SDKs.)

- No prompt de comando, insira o comando a seguir:

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

Para obter mais informações, consulte "Simulação de dispositivos utilizando ADL" na página 101.

Crie o arquivo do pacote APK

Quando seu aplicativo é executado com sucesso, você pode usar o utilitário ADT para empacotar o aplicativo em um arquivo do pacote APK. Um arquivo de pacote APK é o formato de arquivo nativo do aplicativo do Android, que você pode distribuir para seus usuários.

Todos os aplicativos do Android devem ser assinados. Ao contrário dos arquivos do AIR, costuma-se assinar aplicativos do Android com um certificado autoassinado. O sistema operacional Android não tenta estabelecer a identidade do desenvolvedor do aplicativo. Você pode usar um certificado gerado pelo ADT para assinar pacotes do Android. Os certificados usados para os aplicativos apresentados ao Android Market devem ter um período de validade de pelo menos 25 anos.

Gerar um certificado autoassinado e par de chaves

❖ No prompt de comando, insira o comando a seguir (o executável ADT está no diretório `bin` do SDK do Flex):

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

Este exemplo usa o número mínimo de atributos que podem ser definidos para um certificado. O tipo de chave deve ser *1024-RSA* ou *2048-RSA* (consulte o "Comando certificate do ADT" na página 177).

Crie o pacote do AIR

❖ No prompt de comando, digite o seguinte comando (em uma única linha):

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

Você terá que fornecer a senha do arquivo do keystore. Digite a senha e pressione Enter.

Para obter mais informações, consulte "Compactação de um aplicativo do AIR móvel" na página 94.

Instale o runtime do AIR

Você pode instalar a versão mais recente do runtime do AIR a partir do Android Market. Você também pode instalar o runtime incluído no seu SDK em qualquer dispositivo ou um emulador do Android.

❖ No prompt de comando, digite o seguinte comando (em uma única linha):

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

Defina o sinalizador `-platformsdk` para seu diretório SDK do Android SDK (especifique o pai da pasta de ferramentas).

O ADT instala o Runtime.apk incluído no SDK.

Para obter mais informações, consulte "Instale o runtime do AIR e os aplicativos para desenvolvimento" na página 109.

Instale o aplicativo do AIR

❖ No prompt de comando, digite o seguinte comando (em uma única linha):

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

Defina o sinalizador `-platformsdk` para seu diretório SDK do Android SDK (especifique o pai da pasta de ferramentas).

Para obter mais informações, consulte “Instale o runtime do AIR e os aplicativos para desenvolvimento” na página 109.

Você pode iniciar seu aplicativo clicando no ícone do aplicativo na tela do dispositivo ou emulador.

Capítulo 6: Desenvolvendo aplicativos AIR para desktop

Fluxo de trabalho para desenvolver um aplicativo do AIR para desktop

O fluxo de trabalho básico para o desenvolvimento de um aplicativo do AIR é igual ao da maioria dos modelos tradicionais de desenvolvimento: codificar, compilar, testar e, no final do ciclo, compactar em um arquivo de instalação.

Você pode escrever o código do aplicativo usando o Flash, Flex e ActionScript, e compilar usando o Flash Professional, Flash Builder ou os compiladores de linha de comando mxmlc e compc. Você também pode escrever o código do aplicativo usando HTML e JavaScript e ignorando a etapa de compilação.

Você pode testar os aplicativos do AIR desktop com a ferramenta ADL que executa um aplicativo sem exigir que ele seja compactado e instalado pela primeira vez. O Flash Professional, Flash Builder, Dreamweaver e Aptana IDE todos se integram com o depurador do Flash. Você também pode iniciar manualmente a ferramenta do depurador, FDB, ao usar ADL na linha de comando. O ADL, por si, não exibe erros e saída de instrução de rastreio.

Todos os aplicativos AIR devem ser compactados em um arquivo de instalação. O formato de arquivo do AIR de plataforma cruzada é recomendado exceto se:

- Você precisar acessar as APIs dependentes de plataforma como a classe NativeProcess.
- O aplicativo utilizar extensões nativas.

Nestes casos, você pode compactar um aplicativo do AIR como um arquivo de instalação nativo específico da plataforma.

Aplicativos com base em SWF

1. Escreva o código MXML ou ActionScript.
2. Crie recursos necessários, como arquivos de bitmap de ícone.
3. Crie o descritor do aplicativo.
4. Compile o código ActionScript.
5. Testar o aplicativo.
6. Faça o empacotamento e assine como um arquivo do AIR usando o destino *air*.

Aplicativos com base HTML

1. Escreva o código HTML e JavaScript.
2. Crie recursos necessários, como arquivos de bitmap de ícone.
3. Crie o descritor do aplicativo.
4. Testar o aplicativo.
5. Faça o empacotamento e assine como um arquivo do AIR usando o destino *air*.

Criação de instaladores nativos para aplicativos do AIR

1 Escreva o código (ActionScript ou HTML e JavaScript).

2 Crie recursos necessários, como arquivos de bitmap de ícone.

3 Crie a indexação do aplicativo, especificando o perfil *extendedDesktop*.

4 Compile qualquer código ActionScript.

5 Testar o aplicativo.

6 Empacote o aplicativo em cada plataforma de destino usando o destino *native*.

***Nota:** O instalador nativo de uma plataforma de destino deverá ser criado naquela plataforma. Não é possível, por exemplo, criar um instalador de Windows em um Mac. É possível utilizar uma máquina virtual como o VMWare para executar múltiplas plataformas no mesmo hardware de computador.*

Criação de aplicativos do AIR com um conjunto de runtime cativo

1 Escreva o código (ActionScript ou HTML e JavaScript).

2 Crie recursos necessários, como arquivos de bitmap de ícone.

3 Crie a indexação do aplicativo, especificando o perfil *extendedDesktop*.

4 Compile qualquer código ActionScript.

5 Testar o aplicativo.

6 Empacote cada plataforma de destino usando o destino *bundle*.

7 Crie um programa de instalação usando os arquivos de conjunto. (O AIR SDK não fornece ferramentas para criar tal instalador, mas muitos kits de ferramentas terceirizados estão disponíveis).

***Nota:** O conjunto de uma plataforma de destino deverá ser criado naquela plataforma. Não é possível, por exemplo, criar um conjunto de Windows em um Mac. É possível utilizar uma máquina virtual como o VMWare para executar múltiplas plataformas no mesmo hardware de computador.*

Configuração de propriedades do aplicativo desktop

Defina as propriedades básicas do aplicativo no arquivo de descritor do aplicativo. Esta seção aborda as propriedades relevantes para aplicativos do AIR desktop. Os elementos do arquivo de descritor do aplicativo estão completamente descritos em "Arquivos descritores do aplicativo do AIR" na página 208.

Versão de runtime exigida pelo AIR

Especifique a versão do runtime do AIR exigido pelo seu aplicativo usando o namespace do arquivo do descritor do aplicativo.

O namespace, atribuído no elemento do `aplicativo`, determina, em grande parte, que recursos seu aplicativo pode usar. Por exemplo, se seu aplicativo usa o namespace AIR 1.5, e o usuário tem o AIR 3.0 instalado, seu aplicativo vê o comportamento do AIR 1.5 (mesmo que o comportamento tenha sido alterado no AIR 3.0). Somente quando mudar o namespace e publicar uma atualização sua aplicação terá acesso ao novo comportamento e recursos. Alterações de segurança e WebKit são as principais exceções a esta política.

Especifique o namespace usando o atributo xmlns do elemento do `application` raiz:

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

Mais tópicos da Ajuda

"application" na página 213

Identidade do aplicativo

Diversas configurações devem ser exclusivas para cada aplicativo que você publicar. As configurações exclusivas incluem o ID, o nome e o nome do arquivo.

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

Mais tópicos da Ajuda

"id" na página 230

"filename" na página 224

"name" na página 238

Versão do aplicativo

Nas versões AIR anteriores ao AIR 2.5, especifique o aplicativo no elemento `version`. Você pode usar qualquer sequência de caracteres. O runtime do AIR não interpreta a sequência de caracteres; "2.0" não é tratado como uma versão maior de "1.0".

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

No AIR 2.5 e posterior, especifique a versão do aplicativo no elemento `versionNumber`. O elemento `version` não pode mais ser usado. Ao especificar um valor para versionNumber, você deve usar uma sequência de até três números separados por pontos, como: "0.1.2". Cada segmento do número de versão pode ter até três dígitos. (Em outras palavras, "999.999.999" é o maior número de versão autorizada.) Você não precisa incluir todos os três segmentos do número; "1" e "1.0" são números de versão legal.

Você também pode especificar um rótulo para a versão com o elemento `versionLabel`. Ao adicionar um rótulo de versão, este é exibido em vez do número da versão em lugares como as caixas de diálogo do instalador dos aplicativos do AIR.

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

Mais tópicos da Ajuda

"version" na página 245

"versionLabel" na página 246

"versionNumber" na página 246

Propriedades da janela principal

Quando o AIR inicia um aplicativo no desktop, cria uma janela e carrega o arquivo SWF principal ou a página HTML neste. O AIR usa os elementos filho do elemento `initialWindow` que controla a aparência inicial e comportamento da janela inicial do aplicativo.

- **content** - O arquivo SWF do aplicativo principal no filho `content` do elemento `initalWindow`. Ao direcionar dispositivos no perfil de desktop, você pode usar um arquivo SWF ou HTML.

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

  Você deve incluir o arquivo no pacote AIR (usando ADT ou sua IDE). Simplesmente fazer referência do nome no descritor do aplicativo não faz com que o arquivo seja incluído no pacote de maneira automática.

- **depthAndStencil** — Especifica o uso do buffer de profundidade ou de estêncil. Normalmente, você usa esses buffers ao trabalhar com conteúdo 3D.

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

- **height** — A altura da janela inicial.
- **maximizable** — Se o cromo do sistema para maximizar a janela for exibido.
- **maxSize** — O tamanho máximo permitido.
- **minimizable** — Se o cromo do sistema para minimizar a janela for exibido.
- **minSize** — O tamanho mínimo permitido.
- **renderMode** — No AIR 3 ou posterior, o modo de renderização deverá ser definido como *auto*, *cpu*, *direct* ou *gpu* para os aplicativos de desktop. Em versões anteriores do AIR, essa configuração é ignorada nas plataformas de desktop. A configuração renderMode não pode ser alterada no runtime.
  - auto — basicamente o mesmo que o modo da cpu.
  - cpu — exibe objetos que são renderizados e copiados para exibir a memória no software. StageVideo está disponível somente quando uma janela está no modo tela cheia. Stage3D utiliza o renderizador de software.
  - direct — exibe objetos que são renderizados pelo software do runtime, mas a cópia de quadro renderizado para exibir a memória (blitting) é acelerada pelo hardware. StageVideo está disponível. Stage3D utiliza a aceleração de hardware, caso seja possível. Caso uma transparência de janela seja definida como true, então a janela "retrocederá" a renderização e o blitting do software.

    Nota: *Para aproveitar a aceleração GPU do conteúdo Flash com o AIR para plataformas móveis, o Adobe recomenda que você use renderMode="direct", o Stage3D e não o renderMode="gpu". A Adobe oferece suporte e recomenda oficialmente as seguintes estruturas baseadas em Stage3D: Starling (2D) e Away3D (3D). Para obter mais detalhes do Stage3D e do Starling/Away3D (3D), consulte http://gaming.adobe.com/getstarted.*

  - gpu — a aceleração por hardware é usada, se disponível.
- **requestedDisplayResolution** — Se o seu aplicativo tiver que usar o modo de resolução *standard* ou *high* em computadores MacBook Pro com telas de alta resolução. Em todas as outras plataformas o valor é ignorado. Se o valor é *standard*, cada pixel do palco é renderizado como quatro pixels na tela. Se o valor é *high*, cada pixel do palco corresponde a um único pixel físico na tela. O valor especificado é usado para todas as janelas do aplicativo. O uso do elemento `requestedDisplayResolution` para aplicativos de desktop AIR (como um filho do elemento`intialWindow`) está disponível no AIR 3.6 e posterior.
- **resizable** — Se o cromo do sistema para redimensionar a janela for exibido.

- **systemChrome** — Se o adorno da janela do sistema operacional padrão for usado. A configuração do systemChrome de uma janela não pode ser alterada em runtime.
- **title** — O título da janela.
- **transparent** — Se a janela for de mesclagem de alfa em relação ao plano de fundo. A janela não pode usar o cromo do sistema se a transparência estiver ativada. A configuração transparent de uma janela não pode ser alterada em runtime.
- **visible** - Se a janela fica visível assim que é criada. A janela não fica visível por padrão inicialmente para que o aplicativo possa chamar seu conteúdo antes de se tornar visível.
- **width** — A largura da janela.
- **x** — A posição horizontal da janela.
- **y** — A posição vertical da janela.

Mais tópicos da Ajuda

"content" na página 218

"depthAndStencil" na página 220

"height" na página 229

"maximizable" na página 236

"maxSize" na página 237

"minimizable" na página 237

"minimizable" na página 237

"minSize" na página 237

"renderMode" na página 240

"requestedDisplayResolution" na página 240

"resizable" na página 241

"systemChrome" na página 244

"title" na página 245

"transparent" na página 245

"visible" na página 247

"width" na página 247

"x" na página 248

"y" na página 248

Recursos do desktop

Os elementos a seguir controlam os recursos de instalação e atualização do desktop.

- customUpdateUI — Permite que você forneça suas próprias caixas de diálogo para atualizar um aplicativo. Se configurado para `false` o padrão, as caixas de diálogo padrão do AIR são usadas.

- fileTypes — Especifica os tipos de arquivos que seu aplicativo gostaria de registrar como o aplicativo de abertura padrão. Se outro aplicativo já for o padrão para abertura de um tipo de arquivo, o AIR não substitui o registro existente. Contudo, seu aplicativo pode substituir o registro em runtime usando o método `setAsDefaultApplication()` do objeto NativeApplication. É uma boa forma de pedir a permissão do usuário antes de substituir suas associações do tipo de arquivo existente.

  Nota: *O registro do tipo de arquivo é ignorado ao fazer a compactação de um aplicativo como um conjunto de runtime cativo (usando o destino `-bundle`). Para registrar um determinado tipo de arquivo, é necessário criar um programa de instalador que realize o registro.*

- installFolder — Especifica um caminho relativo para a pasta padrão de instalação de aplicativos em que o aplicativo é instalado. Você pode usar esta configuração para fornecer um nome de pasta personalizada, bem como agrupar vários aplicativos dentro de uma pasta comum.
- programMenuFolder — Especifica a hierarquia do menu para o menu Todos os programas do Windows. Você pode usar esta definição para agrupar vários aplicativos em um menu comum. Se nenhuma pasta do menu for especificada o atalho do aplicativo é adicionado diretamente ao menu principal.

Mais tópicos da Ajuda

"customUpdateUI" na página 219

"fileTypes" na página 226

"installFolder" na página 233

"programMenuFolder" na página 239

Perfis disponíveis

Se seu aplicativo só faz sentido no desktop, você pode evitar que ele seja instalado em dispositivos em outro perfil, excluindo esse perfil da lista de perfis disponíveis. Caso o aplicativo utilize a classe NativeProcess ou extensões nativas, então será necessário suportar o perfil `extendedDesktop`.

Se você deixar o elemento `supportedProfile` fora do descritor do aplicativo, este presume que seu aplicativo dá suporte para todos os perfis definidos. Para restringir seu aplicativo a uma lista específica de perfis, liste os perfis, separados por espaços em branco:

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

Para uma lista de classes ActionScript disponíveis nos perfis `desktop` e `extendedDesktop`, consulte "Capacidades de perfis diferentes" na página 250.

Mais tópicos da Ajuda

"supportedProfiles" na página 243

Extensões nativas necessárias

Os aplicativos compatíveis com o perfil `extendedDesktop` podem usar extensões nativas.

Indique todas as extensões nativas que o aplicativo do AIR utiliza no descritor do aplicativo. O seguinte exemplo ilustra a sintaxe para especificar duas extensões nativas necessárias:

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

O elemento `extensionID` tem o mesmo valor que o elemento `id` no arquivo descritor de extensão. O arquivo descritor da extensão é um arquivo XML chamado extension.xml. Ela está compactada no arquivo ANE recebido do desenvolvedor da extensão nativa.

Ícones de aplicativos

No desktop, os ícones especificados no descritor do aplicativo são usados como ícones do menu do programa, atalhos e arquivo do aplicativo. O ícone do aplicativo deve ser fornecido como um conjunto de imagens PNG de 16x16, 32x32, 48x48 e 128x128 pixels. Especifique o caminho para os arquivos de ícone no elemento ícone do arquivo descritor do aplicativo:

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

Se você não fornecer um ícone de determinado tamanho, o maior tamanho seguinte é utilizado e adaptado para se ajustar. Se você não fornecer nenhum ícone, um ícone do sistema padrão será usado.

Mais tópicos da Ajuda

"icon" na página 229

"imageNxN" na página 230

Configurações ignoradas

Aplicativos no desktop ignoram as configurações do aplicativo que se aplicam a recursos de perfil móvel. As configurações ignoradas são:

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

Depuração de um aplicativo do AIR desktop

Se você estiver desenvolvendo seu aplicativo com um IDE como o Flash Builder, Flash Professional ou Dreamweaver, as ferramentas de depuração são normalmente incorporadas. Você pode depurar seu aplicativo simplesmente lançando-o em modo de depuração. Se você não estiver usando um IDE com suporte a depuração diretamente, pode usar o AIR Debug Launcher (ADL) e o depurador do Flash (FDB) para ajudar na depuração do seu aplicativo.

Mais tópicos da Ajuda

De Monsters: Depurador Monster

"Depuração com o AIR HTML Introspector" na página 286

Execução de um aplicativo com ADL

Você pode executar um aplicativo do AIR sem compactação e instalação usando ADL. Passe o arquivo de descritor de aplicativo para o ADL como um parâmetro, como demonstrado no exemplo a seguir (o código ActionScript no aplicativo deve ser compilado primeiro):

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

O ADL imprime instruções de rastreio, exceções de runtime e erros de análise de HTML para a janela de terminal. Se um processo FDB estiver esperando por uma conexão de entrada, o ADL vai conectar-se ao depurador.

Também é possível utilizar a ADL para depurar um aplicativo do AIR que utilize extensões nativas. Por exemplo:

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

Mais tópicos da Ajuda

"AIR Debug Launcher (ADL)" na página 162

Impressão de instruções de rastreamento

Para imprimir instruções de rastreamento para o console usado para executar o ADL, adicione instruções de rastreamento ao seu código com a função `trace()`.

Nota: *Se suas declarações `trace()` não forem exibidas no console, verifique se você não especificou `ErrorReportingEnable` ou `TraceOutputFileEnable` no arquivo mm.cfg. Para obter mais informações sobre o local específico de plataforma desse arquivo, consulte Edição do arquivo mm.cfg.*

Exemplo do ActionScript:

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

Exemplo do JavaScript:

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

No código do JavaScript, você pode usar as funções `alert()` e `confirm()` para exibir mensagens de depuração do seu aplicativo. Além disso, os números de linha para erros de sintaxe, bem como qualquer exceção JavaScript não capturada, serão impressos para o console.

Nota: *Para utilizar o prefixo air mostrado no exemplo do JavaScript, você deve importar o arquivo AIRAliases.js para a página. Esse arquivo está localizado no diretório frameworks do SDK do AIR.*

Conexão ao FDB (depurador do Flash)

Para depurar aplicativos do AIR com o Flash Debugger, inicie a sessão do FDB e inicie seu aplicativo usando ADL.

Nota: *Nos aplicativos do AIR baseados em SWF, os arquivos de origem do ActionScript devem ser compilados com o sinalizador* `-debug`*. (No Flash Professional, marque a opção Permitir depuração no diálogo de Configurações de publicação.)*

1. Inicie o FDB. O programa FDB pode ser encontrado no diretório `bin` do Flex SDK.

   O console exibe o prompt do FDB: `<fdb>`

2. Execute o comando `run`. `<fdb>run [Enter]`
3. Em um comando diferente ou console do shell, inicie uma versão de depuração do seu aplicativo:

   ```
   adl myApp.xml
   ```

4. Usando os comandos do FDB, defina pontos de interrupção como desejado.
5. Digite: `continue [Enter]`

Se um aplicativo do AIR for baseado em SWF, o depurador só controlará a execução do código ActionScript. Se o aplicativo do AIR for baseado em HTML, o depurador só controlará a execução do código JavaScript.

Para executar ADL sem conectar-se ao depurador, inclua a opção `-nodebug`.

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

Para obter informações básicas sobre comandos do FDB, execute o comando `help`:

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

Para obter detalhes sobre comandos FDB, consulte Using the command-line debugger commands na documentação do Flex.

Compactação de um arquivo de instalação AIR desktop.

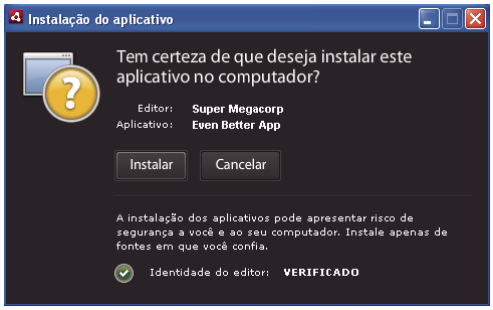
Cada aplicativo do AIR deve, no mínimo, possuir um arquivo do descritor do aplicativo e um arquivo principal do SWF ou HTML. Quaisquer outros ativos a instalar com o aplicativo devem ser também empacotados no arquivo AIR.

Este artigo descreve como compactar um aplicativo do AIR usando as ferramentas de linha de comando incluídas com o SDK. Para obter informações sobre como compactar um aplicativo usando uma das ferramentas de criação da Adobe, consulte o seguinte:

- Adobe® Flex® Builder™; consulte Packaging AIR applications with Flex Builder.
- Adobe® Flash® Builder™; consulte Packaging AIR applications with Flash Builder.
- Adobe® Flash® Professional, consulte Publicação para Adobe AIR.
- Adobe® Dreamweaver®, consulte Criação de um aplicativo do AIR em Dreamweaver.

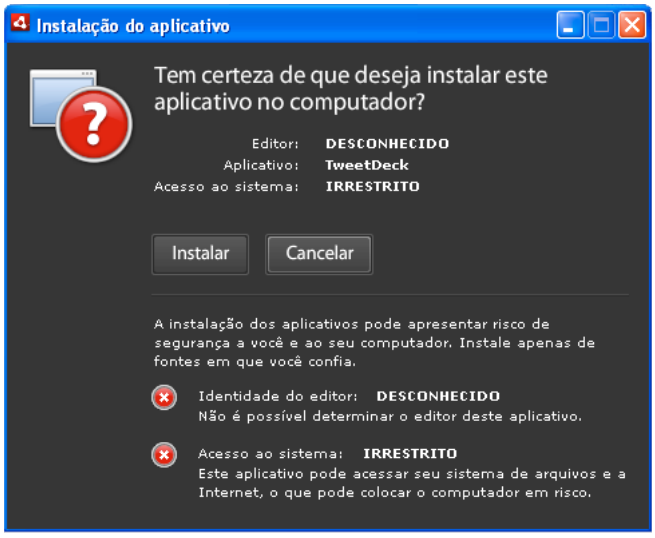
Todos os arquivos do instalador do AIR devem ser assinados usando um certificado digital. O instalador do AIR usa a assinatura para verificar que seu arquivo de aplicativo não foi alterado desde que você o assinou. Você pode usar um certificado de assinatura de código de uma autoridade de certificação ou um certificado autoassinado.

Quando você usa um certificado emitido por uma autoridade de certificação confiável fornece aos usuários do seu aplicativo alguma garantia da sua identidade como editor. A caixa de diálogo da instalação reflete o fato de que sua identidade é verificada pela autoridade de certificação:



Caixa de diálogo da confirmação da instalação assinada por certificado confiável

Quando você usa um certificado autoassinado, os usuários não podem verificar sua identidade como signatário Um certificado autoassinado também enfraquece a garantia de que o pacote não foi alterado. (Isto ocorre porque um arquivo de instalação legítimo pode ter sido substituído por uma falsificação antes de chegar ao usuário.) A caixa de diálogo da instalação reflete o fato de que a identidade do publicador não pode ser verificada. Os usuários assumem um risco de segurança maior quando instalam o seu aplicativo.



A caixa de diálogo da confirmação da instalação do aplicativo, assinada por certificado autoassinado.

Você pode empacotar e assinar um arquivo AIR em uma única etapa usando o comando `-package` do ADT. Você também pode criar um pacote intermediário, não assinado com o comando `-prepare` e assinar o pacote intermediário com o comando `-sign` em uma etapa separada.

Nota: *Versões do Java 1.5 e acima não aceitam caracteres ASCII superiores nas senhas usadas para proteger arquivos de certificado PKCS12. Ao criar ou exportar seu código assinando um arquivo de certificado, use apenas caracteres ASCII comuns na senha.*

Ao assinar o pacote de instalação, o ADT automaticamente entra em contato com um servidor de autoridade com carimbo de data/hora para verificar a hora. As informações de carimbo de data/hora estão incluídas no arquivo AIR. Um arquivo AIR que inclui um carimbo de data/hora verificado pode ser instalado em qualquer momento do futuro. Se o ADT não puder se conectar ao servidor de carimbo de data/hora, o empacotamento será cancelado. Você pode substituir a opção de carimbo de data/hora, mas sem um carimbo de data/hora, um aplicativo do AIR deixa de poder ser instalado após o certificado usado para assinar o arquivo de instalação expirar.

Se você estiver criando um pacote para atualizar um aplicativo do AIR existente, o pacote deverá ser assinado com o mesmo certificado do aplicativo original. Se o certificado original foi renovado ou expirou nos últimos 180 dias ou se você quiser mudar para um novo certificado, poderá aplicar uma assinatura de migração. Uma assinatura de migração envolve a assinatura do arquivo AIR do aplicativo tanto com o certificado novo quanto com o antigo. Use o comando `-migrate` para aplicar a assinatura de migração como descrito em "Comando migrate do ADT" na página 176.

Importante: *É dado um período de graça improrrogável de 180 dias para solicitar a assinatura de migração depois da expiração do certificado original. Sem uma assinatura de migração, os usuários existentes deverão instalar seus aplicativos existentes antes de instalar a nova versão do desenvolvedor. O período de graça abrange somente aplicativos que especificam o AIR versão 1.5.3 ou superior no namespace de descrição do aplicativo. Não existe período de graça destinado a versões do runtime do AIR.*

Antes do AIR 1.1, as assinaturas de migração não eram suportadas. Você deve fazer um pacote que englobe um aplicativo e um SDK da versão 1.1 ou superior para solicitar uma assinatura de migração.

As aplicações implementadas com arquivos AIR são conhecidas como aplicações de perfil de desktop. Você não poderá usar ADT para empacotar um programa de instalação nativo para um aplicativo do AIR se o arquivo de descrição do aplicativo não suportar o perfil de desktop. Você pode restringir este perfil usando o elemento `supportedProfiles` no arquivo de descrição do aplicativo. Consulte "Perfis de dispositivo" na página 249 e "supportedProfiles" na página 243.

Nota: *As configurações no arquivo do descritor do aplicativo determinam a identidade de um aplicativo do AIR e seu caminho de instalação padrão. Consulte "Arquivos descritores do aplicativo do AIR" na página 208.*

IDs do publicador

Como no AIR 1.5.3, os IDs de publicador são deprecados. Os novos aplicativos (originalmente publicados com o AIR 1.5.3 ou superior) não precisam e não devem especificar um ID de publicador.

Ao atualizar os aplicativos publicados com versões mais antigas do AIR, você deve especificar o ID do publicador original no arquivo de descrição do aplicativo. Do contrário, a versão instalada do seu aplicativo e a versão atualizada serão tratados como aplicativos diferentes. Se você usar um ID diferente ou omitir a marca publisherID, um usuário deverá desinstalar a versão antiga antes de instalar a versão nova.

Para descobrir o ID do editor original, localize o arquivo`publisherid` no subdiretório META-INF/AIR onde o aplicativo original está instalado. A sequência de caracteres no arquivo é o ID do editor. O descritor o aplicativo deve especificar o runtime AIR 1.5.3 (ou posterior) na declaração de espaço de nome do arquivo descritor do aplicativo para especificar o ID do editor manualmente.

Para aplicativos antes do AIR 1.5.3 — ou que são publicados com o SDK AIR 1.5.3, enquanto especificam um versão mais antiga do AIR no namespace de descrição do aplicativo — um ID de publicador é computado com base no certificado de assinatura. Este ID é usado com o ID do aplicativo para determinar a identidade de um aplicativo. O ID do editor, quando existir, é utilizado para os seguintes fins:

- Como verificar se um arquivo AIR é uma atualização em vez de um novo aplicativo a ser instalado
- Como parte da chave de criptografia o armazenamento local criptografado
- Como parte o caminho para o diretório de armazenamento do aplicativo
- Como parte da sequência de caracteres de conexão para conexões locais
- Como parte da sequência de caracteres de identidade utilizada para chamar o aplicativo com a API interna de navegador AIR
- Como parte o OSID (utilizado na criação de programas personalizados de instalação/desinstalação)

Antes do AIR 1.5.3, o ID de publicador de um aplicativo podia ser alterado se você assinasse uma atualização do aplicativo com uma assinatura de migração que usasse um certificado novo ou renovado. Quando um ID de editor muda, o comportamento de qualquer recurso AIR dependente do ID também muda. Por exemplo, os dados existentes no armazenamento local criptografado não podem mais ser acessados e qualquer instância do Flash ou AIR que cria uma conexão local para o aplicativo deve utilizar o novo ID na sequência de caracteres de conexão.

NO AIR 1.5.3 ou superior, o ID de publicador não se baseia no certificado de assinatura, mas é atribuído exclusivamente quando a marca publisherID é incluída na descrição do aplicativo. Um aplicativo não poderá ser atualizado se o ID de publicador especificado para o pacote AIR atualizado não corresponder ao ID de publicador atual.

Compactação com ADT

Você pode usar a ferramenta de linha de comando ADT AIR para compactar um aplicativo do AIR. Antes de compactar todo o código MXML, ActionScript e qualquer outra extensão devem ser compilados. Você deve também ter um certificado de assinatura de código.

Para uma referência pormenorizada sobre as opções e os comandos ADT, consulte "AIR Developer Tool (ADT)" na página 168.

Criação de um pacote AIR

Para criar um pacote AIR, use o comando package do ADT, definindo o tipo de destino para compilações de versão *air*.

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

O exemplo pressupõe que o caminho para a ferramenta ADT está na definição do caminho do shell de linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308 para obter ajuda.)

Você deve executar o comando do diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, e um diretório de ícones.

Quando você executa o comando, como demonstrado, a ADT solicitará a senha do armazenamento de chaves. (Os caracteres da senha digitada não são sempre exibidos; basta pressionar Enter quando terminar de digitar.)

Criação de um pacote AIR a partir de um arquivo AIRI

Você pode assinar um arquivo AIRI para criar um pacote AIR instalável:

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

Compactação de um instalador desktop nativo

Como no AIR 2, você pode usar ADT para criar programas de instalação de aplicativo nativo com a finalidade de distribuir aplicativos AIR. Por exemplo, você pode criar um arquivo de instalação EXE para a distribuição de um aplicativo do AIR no Windows. Você pode criar um arquivo de instalação DMG para a distribuição de um aplicativo do AIR no sistema operacional Mac. No AIR 2.5 e no AIR 2.6, é possível criar um arquivo de instalação DEB ou RPM para a distribuição de um aplicativo do AIR no Linux.

Os aplicativos instalados com um programa de instalação de aplicativo nativo são conhecidos como aplicativos de perfil desktop estendido. Você não poderá usar ADT para empacotar um programa de instalação nativo para um aplicativo do AIR se o arquivo de descrição do aplicativo não suportar o perfil de desktop estendido. Você pode restringir este perfil usando o elemento `supportedProfiles` no arquivo de descrição do aplicativo. Consulte "Perfis de dispositivo" na página 249 e "supportedProfiles" na página 243.

Você pode criar uma versão de instalação nativa do aplicativo do AIR de duas maneiras básicas:

- Você pode criar o instalador nativo com base no arquivo de descrição do aplicativo e em outros arquivos de origem. (Outros arquivos de origem podem incluir arquivos SWF, arquivos HTML e outros ativos.)
- Você pode criar o instalador nativo com base em um arquivo AIR ou com base em um arquivo AIRI.

É obrigatório usar ADT no mesmo sistema operacional do arquivo de instalação nativo que você deseja gerar. Portanto, para criar u arquivo EXE para Windows, execute ADT no Windows. Para criar um arquivo DMG para o Mac OS, execute ADT no Mac OS. Para criar um arquivo DEB ou RPG para Linux, execute o ADT a partir do SDK do Air 2.6 no Linux.

Quando você cria um programa de instalação nativo para distribuir um aplicativo do AIR, o aplicativo ganha estas capacidades:

- Ele pode lançar e interagir com processos nativos usando a classe NativeProcess. Para mais detalhes, consulte um dos seguintes tópicos:
  - Comunicação com processos nativos do AIR (para desenvolvedores em ActionScript)
  - Communicating with native processes in AIR (para desenvolvedores em HTML)
- Ele poderá usar extensões nativas.
- Ele pode usar o método `File.openWithDefaultApplication()` para abrir qualquer arquivo que tenha o aplicativo de sistema padrão definido para abri-lo, independentemente do tipo do seu arquivo. (Existem restrições para aplicativos que *não* são instalados com um programa de instalação nativo. Para saber detalhes, consulte entrada correspondente à entrada `File.openWithDefaultApplication()` na referência de linguagem.)

No entanto, quando compactado como um instalador nativo, o aplicativo perde alguns dos benefícios do formato de arquivo do AIR. Um único arquivo já não pode ser distribuído a todos os computadores desktop. A função de atualização incorporada (bem como a estrutura do atualizador) não funciona.

Quando o usuário clica duas vezes no arquivo de instalação nativo, isto instala o aplicativo do AIR. Se a versão requerida do Adobe AIR ainda não estiver instalada na máquina, o programa de instalação baixa a versão da rede e a instala primeiro. Se não houver conexão com a rede da qual se possa obter a versão correta do Adobe AIR (se necessário), a instalação falhará. Igualmente, a instalação falhará se o sistema operacional não for suportado no Adobe AIR2.

Nota: *Se você desejar que um arquivo seja executável no aplicativo instalado, certifique-se de que ele é executável no sistema de arquivos, durante a criação do pacote do aplicativo. (No Mac e Linux, você pode utilizar o comando chmod para definir o indicador de executável, se necessário.)*

Como criar um programa de instalação nativo a partir de arquivos-fonte do aplicativo

Para criar um programa de instalação nativo a partir de arquivos-fonte do aplicativo, use o comando `-package` com a seguinte sintaxe (em uma linha de comando simples):

```
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 sintaxe é semelhante à sintaxe para empacotar um arquivo AIR (sem um programa de instalação nativo) No entanto, existem algumas diferenças:

- Você adiciona a opção `-target native` ao comando. (Se você especificar `-target air`, então o ADT irá gerar um arquivo AIR em vez de um arquivo de instalação nativo.)
- Você especifica o arquivo DMG ou EXE de destino como o `installer_file`.
- Opcionalmente, no Windows você pode adicionar um segundo conjunto de opções de assinatura, indicado como `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` na listagem de sintaxe. No Windows, além de assinar o arquivo AIR, você pode assinar o arquivo Windows Installer. Use o mesmo tipo de certificado e sintaxe de opção de assinatura que usaria para assinar o arquivo AIR (consulte "Opções de assinatura de código ADT" na página 182). Você pode utilizar o mesmo certificado para assinar o arquivo AIR e o arquivo de instalação ou pode especificar certificados diferentes. Quando um usuário faz um download da Web de um arquivo Windows Installer assinado, o Windows identifica a origem do arquivo com base no certificado.

Para obter mais detalhes sobre opções do ADT além de outra opção `-target`, consulte "AIR Developer Tool (ADT)" na página 168.

O exemplo a seguir cria um arquivo DMG (um arquivo de instalação nativo para o sistema operacional Mac):

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

O exemplo a seguir cria um arquivo EXE (um arquivo de instalação nativo para o sistema operacional Windows):

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

O exemplo a seguir cria um arquivo EXE e o assina:

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

Criação de um instalador nativo para um aplicativo que utilize extensões nativas

É possível criar um instalador nativo dos arquivos de origem para o aplicativo e os pacotes de extensão nativa que o aplicativo exige. Use o comando `-package` com a seguinte sintaxe (em uma única linha de comando):

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

Essa sintaxe é a mesma usada para empacotar um instalador nativo, com duas opções adicionais. Use a opção `-extdir` *`extension-directory`* para especificar o diretório que contém os arquivos ANE (extensões nativas) que o aplicativo utiliza. Use o sinalizador `-migrate` opcional e os parâmetros `MIGRATION_SIGNING_OPTIONS` para assinar uma atualização de um aplicativo com uma assinatura de migração quando o certificado de assinatura de código principal for diferente do usado pela versão anterior. Para obter mais informações, consulte “Assinatura de uma versão atualizada de um aplicativo do AIR” na página 203.

Para obter detalhes sobre as opções do ADT, consulte a“AIR Developer Tool (ADT)” na página 168.

O seguinte exemplo cria um arquivo DMG (um arquivo de instalador nativo para o Mac OS) para um aplicativo que utilize extensões nativas:

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

Como criar um programa de instalação nativo com base em um arquivo AIR ou em um arquivo AIRI.

Você pode usar ADT para gerar um arquivo de instalação nativo com base em um arquivo AIR ou em um arquivo AIRI. Para criar um programa de instalação nativo com base em um arquivo AIR, use o comando `-package` do ADT com a seguinte sintaxe (em uma linha de comando simples):

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

Esta sintaxe é semelhante à sintaxe para criar um programa de instalação nativa com base em arquivos-fonte para o aplicativo do AIR. No entanto, existem algumas diferenças:

- Você especifica um arquivo AIR como origem, em vez de um arquivo de descrição de aplicativo e outros arquivos de origem para o aplicativo do AIR.
- Não especifique opções de assinatura para o arquivo AIR, porque ele já está assinado.

Para criar um programa de instalação nativo com base em um arquivo *AIRI*, use o comando `-package` do ADT com a seguinte sintaxe (em uma linha de comando simples):

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

Esta sintaxe é semelhante à sintaxe para criar um programa de instalação nativo com base em um arquivo AIR. No entanto, existem algumas diferenças:

- Como origem, você deve especificar um arquivo AIRI.
- Você deve especificar opções de assinatura para o aplicativo do AIR de destino.

O exemplo a seguir cria um arquivo DMG (um arquivo de instalação nativo para o sistema operacional Mac) com base em um arquivo AIR:

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

O exemplo a seguir cria um arquivo EXE (um arquivo de instalação nativo para o sistema operacional Windows) com base em um arquivo AIR:

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

O exemplo a seguir cria um arquivo EXE (com base em um arquivo AIR) e o assina:

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

O exemplo a seguir cria um arquivo DMG (um arquivo de instalação nativo para o sistema operacional Mac) com base em um arquivo AIRI:

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

O exemplo a seguir cria um arquivo EXE (um arquivo de instalação nativo para o sistema operacional Windows) com base em um arquivo AIRI:

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

O exemplo a seguir cria um arquivo EXE (com base em um arquivo AIRI) e assina-o com uma assinatura nativa do Windows e um AIR:

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

Compactação de um conjunto de runtime cativo para computadores pessoais

Um conjunto de runtime cativo é um pacote que inclui o código do aplicativo junto com uma versão customizada do runtime. Um aplicativo compactado dessa maneira utiliza o runtime integrado, em vez de um runtime compartilhado instalado em outro local no computador do usuário.

O conjunto gerado é uma pasta independente de arquivos do aplicativo no Windows e um conjunto .app no Mac OS. Você deve gerar o conjunto para um sistema operacional de destino enquanto executa aquele sistema operacional. (Uma máquina virtual como o VMWare pode ser utilizada para executar vários sistemas operacionais em um computador.)

O aplicativo pode ser executado daquela pasta ou conjunto sem a instalação.

Benefícios

- Gera um aplicativo independente
- Não é necessário o acesso à Internet para a instalação
- O aplicativo é isolado das atualizações de runtime
- As empresas podem certificar a combinação do aplicativo e do runtime específicos.
- Suporta o tradicional modelo de implementação do software
- Não é necessária a redistribuição separada de runtime
- Pode utilizar a API do NativeProcess
- Pode utilizar extensões nativas
- Pode utilizar a função `File.openWithDefaultApplication()` sem restrição
- Pode ser executado de um USB ou disco óptico sem a instalação

Desvantagens

- Correções importantes de segurança não estão automaticamente disponíveis para os usuários quando a Adobe publica um patch de segurança
- Não pode utilizar o formato de arquivo .air
- Você deve criar seu próprio instalador, se necessário
- Sistema e API de atualização do AIR não são suportados
- A API embutida no navegador do AIR para instalar e lançar um aplicativo do AIR a partir de uma página da web não é suportada
- No Windows, o registro de arquivo deve ser feito pelo instalador
- Maior memória do disco do aplicativo

Criação de um conjunto de runtime cativo no Windows

Para criar um conjunto de runtime cativo para o Windows, é necessário compactar o aplicativo enquanto ele é executado em um sistema operacional do Windows. Empacote o aplicativo usando o destino *bundle* da ADT:

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

Esse comando cria o conjunto em um diretório denominado myApp. Esse diretório contém os arquivos do aplicativo, além dos arquivos de runtime. É possível executar o programa diretamente da pasta. No entanto, para criar uma entrada de menu do programa, tipos de arquivos de registro ou o controlador do esquema URI, você deve criar um programa de instalação que defina as entradas de registro necessárias. O AIR SDK não inclui as ferramentas para criar tais instaladores, mas várias opções terceirizadas estão disponíveis, incluindo ambos os kits de ferramentas do instalador de código aberto gratuito e comercial.

É possível assinar o executável nativo no Windows, especificando um segundo conjunto de opções de assinatura após a entrada `-target bundle` na linha de comando. Essas opções de assinatura identificam a chave privada e o certificado associado a serem utilizados ao aplicar a assinatura nativa do Windows. (Um certificado de assinatura de código do AIR geralmente pode ser utilizado.) Somente o executável primário é assinado. Quaisquer executáveis adicionais compactados com o aplicativo não são assinados através desse processo.

Associação do tipo de arquivo
Para associar o aplicativo com os tipos de arquivo públicos ou customizados no Windows, o programa de instalação deverá definir as entradas de registro adequadas. Os tipos de arquivos também deverão estar listados no elemento fileTypes do arquivo descritor do aplicativo.

Para obter mais informações sobre os tipos de arquivo do Windows, consulte Biblioteca MSDN: Tipos de Arquivo e Associações de Arquivo

Registro do controlador URI
Para o aplicativo manipular o lançamento de um URL utilizando um determinado esquema de URI, o instalador deverá definir as entradas de registro necessárias.

Para obter mais informações sobre o registro de um aplicativo para efetuar um esquema de URI, consulte Biblioteca MSDN: Registro de um Aplicativo em um Protocolo de URL

Criação de um conjunto de runtime cativo no Mac OS X

Para criar um conjunto de runtime cativo para o Mac OS X, é necessário compactar o aplicativo enquanto ele é executado no sistema operacional Macintosh. Empacote o aplicativo usando o destino *bundle* da ADT:

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

Esse comando cria o conjunto de aplicativo nomeado myApp.app. O conjunto contém os arquivos para o aplicativo, além dos arquivos de runtime. É possível executar o aplicativo clicando duas vezes no ícone do myApp.app e instalá-lo arrastando-o até um local apropriado, como a pasta Aplicativos. No entanto, para registrar tipos de arquivos ou controladores de esquemas de URI, é necessário editar o arquivo da lista de propriedades dentro do pacote do aplicativo.

Para a distribuição, você poderá criar um arquivo de imagem de disco (.dmg). O Adobe AIR SDK não fornece as ferramentas para criar um arquivo dmg para um conjunto de runtime cativo.

Associação do tipo de arquivo
Para associar o aplicativo aos tipos de arquivos públicos ou customizados no Mac OS X, é necessário editar o arquivo info.plist no conjunto para definir as propriedades de CFBundleDocumentTypes. Consulte Biblioteca do Desenvolvedor do Mac OS X: Referências de Informação da Chave da Lista de Propriedades, CFBundleURLTypes.

Registro do controlador URI
Para o aplicativo executar o lançamento de um URL utilizando um determinado esquema de URI, é necessário editar o arquivo info.plist no conjunto para definir as propriedades de CFBundleURLTypes Consulte Biblioteca do Desenvolvedor do Mac OS X: Referências de Informação da Chave da Lista de Propriedades, CFBundleDocumentTypes.

Distribuição de pacotes do AIR para computadores desktop

Os aplicativos AIR podem ser distribuídos como um pacote AIR, que contém o código do aplicativo e todos os ativos. Você pode distribuir esse pacote por qualquer um dos meios típicos, como por download, email ou mídia física, como CD-ROM. Os usuários podem instalar o aplicativo clicando duas vezes no arquivo AIR. Você pode usar a API no navegador AIR (uma biblioteca ActionScript com base web) para permitir que os usuários instalem seu aplicativo do AIR (e Adobe® AIR®, se necessário), clicando em um único link em uma página da web.

Os aplicativos AIR podem também ser compactados e distribuídos como instaladores nativos (em outras palavras, como arquivos EXE no Windows, DMG no Mac, e DEB ou RPM no Linux). Os pacotes de instalação nativa podem ser distribuídos e instalados de acordo com as convenções da plataforma em questão. Ao distribuir seu aplicativo como um pacote nativo, você perde alguns dos benefícios do formato de arquivo do AIR. Ou seja, um único arquivo de instalação não pode mais ser usado na maioria das plataformas, a estrutura de atualização do AIR não pode mais ser usado, e a API no navegador não pode mais ser usada.

Instalação e execução de um aplicativo do AIR na área de trabalho

Você pode simplesmente enviar o arquivo AIR ao destinatário. Por exemplo, você pode enviar o arquivo AIR como um anexo de email ou um link em uma página da Web.

Depois que o usuário baixar o aplicativo do AIR, ele deverá seguir estas instruções para instalá-lo:

1 Clique duas vezes no arquivo AIR.

O Adobe AIR já deve estar instalado no computador.

2 Na janela de instalação, deixe as configurações padrão selecionadas e clique em Continuar.

No Windows, o AIR faz automaticamente o seguinte:

- Instala o aplicativo no diretório Arquivos de Programas
- Cria um atalho na área de trabalho para o aplicativo
- Cria um atalho no menu Iniciar
- Adiciona uma entrada para o aplicativo no Painel de Controle Adicionar ou Remover Programas

No Mac OS, por padrão, o aplicativo é adicionado ao diretório Aplicativos.

Se o aplicativo já estiver instalado, o instalador oferece ao usuário a opção de abrir a versão existente do aplicativo ou atualizar para a versão no arquivo AIR obtido por download. O instalador identifica o aplicativo usando o ID do aplicativo e o ID do editor no arquivo AIR.

3 Quando a instalação estiver concluída, clique em Concluir.

No Mac OS, para instalar uma versão atualizada de um aplicativo, o usuário precisa de privilégios adequados do sistema para instalar no diretório do aplicativo. No Windows e no Linux, um usuário precisa de privilégios administrativos.

Um aplicativo também pode instalar uma versão nova via ActionScript ou JavaScript. Para obter mais informações, consulte “Atualização de aplicativos do AIR” na página 262.

Depois que o aplicativo do AIR está instalado, um usuário simplesmente clica duas vezes no ícone do aplicativo para executá-lo, como qualquer outro aplicativo de área de trabalho.

- No Windows, clique duas vezes no ícone do aplicativo (que está instalado na área de trabalho ou em uma pasta) ou selecione o aplicativo no menu Iniciar.
- No Linux, clique duas vezes no ícone do aplicativo (que está instalado na área de trabalho ou em uma pasta) ou selecione o aplicativo no menu aplicativos.
- No Mac OS, clique duas vezes no aplicativo na pasta em que ele foi instalado. O diretório de instalação padrão é o diretório /Aplicativos.

Nota: *Somente os aplicativos do AIR desenvolvidos para o AIR 2.6 ou anterior podem ser instalados no Linux.*

O recurso de *instalação direta* do AIR permite que um usuário instale um aplicativo do AIR clicando em um link em uma página da Web. O recurso de *invocação do navegador* do AIR permite que um usuário execute um aplicativo do AIR instalado clicando em um link em uma página da Web. Esses recursos são descritos na seção a seguir.

Instalação e execução de aplicativos do AIR desktop de uma página da Web

A API no navegador AIR permite que você instale e execute o aplicativo do AIR de uma página web. A API no navegador AIR é fornecida em uma biblioteca de SWF, *air.swf*, hospedada pela Adobe. O AIR SDK inclui um aplicativo de "emblema" de amostra que utiliza esta biblioteca para instalar, atualizar ou iniciar um aplicativo do AIR (e o runtime, se necessário). Você pode modificar o emblema de amostra fornecido ou criar seu próprio aplicativo web de emblema que usa a biblioteca air.swf online diretamente.

Qualquer aplicativo do AIR pode ser instalado através de um emblema de página da web. Mas apenas aplicativos que incluem o elemento `<allowBrowserInvocation>true</allowBrowserInvocation>` nos arquivos de descritor do aplicativo podem ser emblema da web.

Mais tópicos da Ajuda

"API no navegador AIR.SWF" na página 253

Implementação comercial em computadores desktop.

Os administradores de TI podem instalar o runtime do Adobe AIR e aplicativos do AIR de modo silencioso usando ferramentas de implantação de área de trabalho padrão. Os administradores de TI podem fazer o seguinte:

- Instalar silenciosamente o runtime do Adobe AIR usando ferramentas como Microsoft SMS, IBM Tivoli, ou qualquer ferramenta de implantação que permita instalações silenciosas que usem um inicializador
- Instalar silenciosamente o aplicativo do AIR usando as mesmas ferramentas usadas para implantar o runtime

Para obter mais informações, consulte o *Guia do administrador do Adobe AIR (http://www.adobe.com/go/learn_air_admin_guide_br).*

Registros de instalação nos computadores desktop

Os registros de instalação são gravados quando o próprio runtime do AIR ou um aplicativo do AIR é instalado. Você pode examinar os arquivos de log para ajudar a determinar a causa de alguns problemas de instalação ou atualização que ocorrerem.

Os arquivos de registro são criados nos seguintes locais:

- Mac: o registro do sistema padrão (`/private/var/log/system.log`)

  Você pode exibir o registro do sistema do Mac abrindo o aplicativo Console (normalmente encontrado na pasta Utilitários).

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

Nota: *Estes arquivos de registro não foram criados nas versões do AIR anteriores ao AIR 2.*

Capítulo 7: Desenvolvendo aplicativos AIR para dispositivos móveis

Os aplicativos AIR em dispositivos móveis são implementados como aplicativos nativos. Eles usam o formato de aplicativo do dispositivo, e não o formato de arquivo do AIR. Atualmente o AIR suporta pacotes Android APK e pacotes IPA iOS. Depois de ter criado a versão do seu pacote de aplicativos, você pode distribuir seu aplicativo através do mecanismo de plataforma padrão. Para o Android, isso normalmente significa o Android Market; para o iOS, a Apple App Store.

Você pode usar o AIR SDK AIR SDK e Flash Professional, Flash Builder, ou outra ferramenta de desenvolvimento ActionScript para criar aplicativos AIR para dispositivos móveis. Aplicativos AIR móveis com base HMTL não estão disponíveis no momento.

Nota: *O BlackBerry Playbook Research In Motion (RIM) fornece seu próprio SDK para desenvolvimento no AIR. Para informações sobre o desenvolvimento do Playbook, consulte RIM: Desenvolvimento do SO do tablet da BlackBerry.*

Nota: *Este documento descreve como desenvolver aplicativos iOS usando o AIR 2.6 ou versões posteriores. Os aplicativos criados com o AIR 2.6+ podem ser instalados no iPhone 3Gs, no iPhone 4 e nos dispositivos iPad que executam o iOS 4 ou posterior. Para desenvolver aplicativos AIR para versões anteriores do iOS, você deve usar o AIR 2 Packager for iPhone, como descrito em Criação de aplicativos para o iPhone.*

Para obter mais informações sobre as práticas recomendadas de privacidade, consulte o Guia de privacidade do SDK do Adobe AIR.

Para saber os requisitos completos de sistema com a finalidade de executar aplicativos AIR, consulte Requisitos do sistema Adobe AIR.

Configuração do ambiente de desenvolvimento

As plataformas móveis têm requisitos adicionais de configuração além da configuração do ambiente de desenvolvimento AIR, Flex, Flash normais. (Para obter mais informações sobre como configurar o ambiente de desenvolvimento do AIR básico, consulte "Ferramentas da Plataforma Adobe Flash para desenvolvimento do AIR" na página 16).

Configuração do Android

Não é necessária nenhuma instalação especial para o Android no AIR 2.6+. A ferramenta ADB do Android está incluída no SDK do AIR (na pasta lib/android/bin). O AIR SDK usa a ferramenta ADB para instalar, desinstalar e executar os pacotes de aplicativos no dispositivo. Você também pode usar ADB para ver os logs do sistema. Para criar e executar um emulador do Android você deve baixar o Android SDK separado.

Se o seu aplicativo adicionar elementos ao elemento `<manifestAdditions>` no descritor de aplicativo que a versão atual do AIR não reconheça como válida, você deverá instalar uma versão mais recente do SDK do Android. Defina a variável de ambiente AIR_ANDROID_SDK_HOME ou o parâmetro da linha de comando `-platformsdk` ao caminho de arquivo do SDK. A ferramenta de empacotamento do AIR, ADT, usa esse SDK para validar as entradas no elemento `<manifestAdditions>`.

No AIR 2.5, você deve baixar uma cópia separada do Android SDK da Google. Você pode definir a variável de ambiente AIR_ANDROID_SDK_HOME para fazer referência à pasta do Android SDK. Se você não definir esta variável de ambiente, deve especificar o caminho para o Android SDK no argumento `-platformsdk` na linha de comando do ADT.

Mais tópicos da Ajuda

"Variáveis de ambiente ADT" na página 192

"Variáveis de ambiente do caminho" na página 308

Configuração do iOS

Para instalar e testar um aplicativo iOS em um dispositivo e distribuí-lo você deve aderir ao programa Apple iOS Developer (que é um serviço pago). Depois de entrar no programa de desenvolvedor do iOS você pode acessar o Portal de aprovisionamento do iOS onde poderá obter os itens a seguir e arquivos da Apple que são necessários para instalar um aplicativo em um dispositivo para testes e para posterior distribuição. Estes itens e arquivos incluem:

- Certificados de distribuição e desenvolvimento
- IDs de aplicativos
- Arquivos de aprovisionamento de desenvolvimento e distribuição

Considerações a respeito do design do aplicativo móvel

O contexto operacional e as características físicas dos dispositivos móveis exigem codificação e design cuidadosos. Por exemplo, a simplificação do código para que seja executado o mais rápido possível é crucial. A otimização de código só pode ir tão longe, é claro, se o design inteligente que trabalha dentro das limitações do dispositivo também ajudar a evitar que sua apresentação visual sobrecarregue o sistema de renderização.

Code

Enquanto fazer seu código funcionar mais rápido é sempre benéfico, a velocidade mais lenta do processador da maioria dos dispositivos móveis aumenta as recompensas do tempo gasto escrevendo códigos limpos. Além disso, os dispositivos móveis são quase sempre executados com a energia da bateria. Obter o mesmo resultado com menos trabalho requer menos energia da bateria.

Design

Fatores como o pequeno tamanho da tela, modo de interação da tela de toque, e até o ambiente em constante mudança de um usuário móvel devem ser considerados ao projetar a experiência do usuário do seu aplicativo.

Código e design juntos

Se seu aplicativo usa animação, a renderização da otimização é muito importante. Contudo, a otimização do código sozinha muitas vezes não é suficiente. Você deve projetar os aspectos visuais do aplicativo de tal forma que o código possa renderizá-los de forma eficiente.

Técnicas importantes de otimização são discutidas no guia Otimizando o desempenho para a plataforma Flash. As técnicas abordadas no guia se aplicam a todo o conteúdo em Flash e AIR, mas são essenciais para o desenvolvimento de aplicativos que funcionem bem em dispositivos móveis.

- Paul Trani: Dicas e Truques para o Desenvolvimento no Mobile Flash
- roguish: Aplicativo de Teste de GPU do AIR para Mobile

- Jonathan Campos: Técnicas de Otimização dos Aplicativos AIR for Android
- Charles Schulze: Desenvolvimento de Jogos no AIR 2.6: iOS incluso

Ciclo de vida do aplicativo

Quando o aplicativo perde o foco para outro aplicativo, o AIR cai a taxa de quadros para 4 quadros por segundo e para a renderização de gráficos. Abaixo desta taxa de quadros, o fluxo de rede e conexões de soquete tendem a serem interrompidos. Se seu aplicativo não usa estas conexões, você pode tornar a taxa de quadros ainda menor.

Quando adequado, você deve parar de reprodução de áudio e remover os listeners para os sensores de geolocalização e acelerômetro. O objeto AIR NativeApplication despacha eventos ativos e inativos. Use estes eventos para gerenciar a transição entre o estado ativo e o estado do plano de fundo.

A maioria dos sistemas operacionais móveis encerram aplicativos em segundo plano sem aviso prévio. Ao salvar o estado do aplicativo com frequência, seu aplicativo deve ser capaz de restaurar-se a um estado razoável seja retornando para o status ativo do plano de fundo ou sendo ativado um novo.

Densidade das informações

O tamanho físico da tela de dispositivos móveis é menor que na área de trabalho, embora sua densidade em pixels (pixels por polegada) seja mais alta. O mesmo tamanho de fonte produzirá letras que são fisicamente menores na tela de um dispositivo móvel do que em um computador desktop. Muitas vezes é necessário usar uma fonte maior para garantir a legibilidade. Em geral, 14 pontos é o menor tamanho de fonte que pode ser lido facilmente.

Os dispositivos portáteis são frequentemente usados em movimento e em fracas condições de iluminação. Leve em consideração quanta informação você pode realmente mostrar na tela de forma legível. Talvez seja menos que poderia exibir em uma tela com as mesmas dimensões em pixels em uma área de trabalho.

Além disso, considere que quando um usuário está tocando a tela, o dedo e a mão bloqueiam parte da tela de exibição. Coloque elementos interativos nas laterais e na parte inferior da tela quando o usuário tiver de interagir com estes em mais de um toque momentâneo.

Entrada de texto

Muitos dispositivos usam um teclado virtual para entrada de texto. Teclados virtuais ocultam parte da tela e muitas vezes são complicados de usar. Evite contar com eventos de teclado (exceto teclas de função).

Considere a implantação de alternativas para usar campos de texto de entrada. Por exemplo, não é necessário um campo de texto para o usuário digitar um valor numérico. É possível fornecer dois botões para aumentar ou diminuir o valor.

Teclas de função

Os dispositivos móveis incluem um número variável de teclas de função. As teclas de função são botões programáveis para funções diferentes. Siga as convenções de plataforma para essas teclas em seu aplicativo.

Alterações na orientação da tela

É possível visualizar o conteúdo móvel na orientação retrato ou paisagem. Considere a forma como o aplicativo lidará com as alterações na orientação da tela. Para obter mais informações, consulte Orientação do palco.

Escurecimento da tela

O AIR não evita automaticamente o escurecimento da tela enquanto o vídeo é reproduzido. Você pode usar a propriedade `systemIdleMode` do objeto AIR NativeApplication para controlar se o aparelho entrará no modo de economia de energia. (Em algumas plataformas você deve solicitar as permissões adequadas para que esse recurso funcione.)

Chamadas de telefone

O runtime de áudio silencia automaticamente quando o usuário faz ou recebe um telefonema. No Android você deve definir a permissão READ_PHONE_STATE no descritor do aplicativo se este reproduz áudio enquanto está no plano de fundo. Caso contrário, o Android impede o runtime de detectar chamadas telefônicas e de silenciar o áudio automaticamente. Consulte “Permissões do Android” na página 77.

Destinos de ocorrências

Considere o tamanho dos destinos de ocorrência ao projetar botões e outros elementos da interface do usuário que o usuário pressiona. Aumente esses elementos o suficiente para serem ativados confortavelmente com o dedo na tela de toque. Além disso, certifique-se de que existe espaço suficiente entre os destinos. A área de destino deve atingir cerca de 44 pixels por 57 pixels em cada lado de uma tela comum de celular de alta dpi.

Tamanho da instalação do pacote do aplicativo

Os dispositivos móveis normalmente têm muito menos espaço de armazenamento para a instalação de aplicativos e dados do que os computadores desktop. Reduza o tamanho do pacote eliminando bibliotecas e ativos não utilizados.

No Android o pacote do aplicativo não é extraído em arquivos separados quando um aplicativo é instalado. Em vez disso, os ativos são descompactados em armazenamento temporário quando são acessados. Para reduzir este espaço de armazenamento de ativos descompactados, feche fluxos de URL e o arquivo quando os ativos estiverem completamente carregados.

Acesso ao sistema de arquivos

Sistemas operacionais móveis diferentes impõem diferentes restrições de sistema de arquivos, e essas restrições tendem a ser diferentes das impostas pelos sistemas operacionais de desktop. O lugar apropriado para salvar arquivos e dados, portanto, pode variar de plataforma para plataforma.

Uma consequência da variação no sistema de arquivos é que os atalhos para diretórios comuns fornecidos pela classe AIR File nem sempre estão disponíveis. A seguinte tabela mostra quais atalhos podem ser usados no Android e no iOS:

	Android	**iOS**
File.applicationDirectory	Somente leitura por URL (caminho não ativo)	Somente leitura
File.applicationStorageDirectory	Disponível	Disponível
File.cacheDirectory	Disponível	Disponível
File.desktopDirectory	Raiz de sdcard	Indisponível
File.documentsDirectory	Raiz de sdcard	Disponível

	Android	**iOS**
File.userDirectory	Raiz de sdcard	Indisponível
File.createTempDirectory()	Disponível	Disponível
File.createTempFile()	Disponível	Disponível

Diretrizes da Apple para aplicativos iOS fornecem regras específicas sobre os locais de armazenamentos que devem ser usados para arquivos em várias situações. Por exemplo, uma diretriz é que somente arquivos que contenham dados inseridos pelo usuário ou dados que não possam ser recuperados ou baixados novamente devem ser armazenados em um diretório designado para backup remoto. Para obter informações sobre como cumprir as diretrizes da Apple para backup de arquivos e armazenamento em cache, consulte Controlar o backup de arquivos e o armazenamento em cache.

Componentes da UI

A Adobe desenvolveu uma versão otimizada para dispositivos móveis da estrutura Flex. Para obter mais informações, consulte Desenvolvimento de aplicativos móveis com o Flex e o Flash Builder.

Também estão disponíveis projetos de componentes comunitários adequados para aplicativos móveis. Isso inclui:

- Controles de difusão de IU para Starling de Josh Tynjala
- Versão da Minimal Comps com visual personalizado de Derrick Grigg
- Componentes as3flobile de Todd Anderson

Renderização acelerada de gráficos Stage 3D

Começando no AIR 3.2, o AIR para dispositivos móveis oferece suporte à renderização de gráficos Stage 3D acelerada. As APIs Stage3D do ActionScript são um conjunto de APIs aceleradas por GPU de baixo nível que ativam as capacidades de 2D e 3D avançadas. Essas APIs de baixo nível fornecem aos desenvolvedores a flexibilidade para aproveitar os significativos ganhos de desempenho. Você também pode usar os mecanismos para jogos que suportem as APIs Stage3D do ActionScript.

Para obter mais informações, consulte Mecanismos para jogos, 3D e Stage 3D.

Suavização de vídeo

Para melhorar o desempenho, a suavização de vídeo é desabilitada no AIR.

Recursos nativos

AIR 3.0+

Muitas plataformas móveis oferecem recursos que ainda não estão acessíveis por meio da API AIR padrão. Começando no AIR 3, você pode estender o AIR com suas próprias bibliotecas de código nativas. Essas bibliotecas de extensão nativas podem acessar recursos disponíveis pelo sistema operacional ou até mesmo específicas de um determinado dispositivo. As extensões nativas podem ser criadas em C no iOS e em Java ou C no Android. Para obter informações sobre o desenvolvimento de extensões nativas, consulte Apresentação de extensões nativas para o Adobe AIR.

Fluxo de trabalho para a criação de aplicativos AIR para dispositivos móveis

O fluxo de trabalho para a criação de uma aplicativo do AIR para dispositivos móveis (ou outros) é, em geral, muito semelhante ao usado para criar um aplicativo desktop. As principais diferenças de fluxo de trabalho ocorrem quando é hora de compactar, depurar e instalar um aplicativo. Por exemplo, os aplicativos AIR for Android usam o formato nativo de pacote APK do Android ao invés do formato do pacote AIR. Assim, eles também usam os mecanismos padrão Android para instalar e atualizar.

AIR for Android

As etapas a seguir são típicas ao desenvolver um aplicativo do AIR para o Android:

- Escreva o código ActionScript ou MXML.
- Criar um arquivo descritor do aplicativo do AIR (usando o namespace, 2.5 ou superior).
- Compile o aplicativo.
- Compacte o aplicativo como um pacote Android (.apk).
- Instale o runtime do AIR em um dispositivo ou emulador do Android (se você estiver usando um runtime externo; o runtime cativo é o padrão do AIR 3.7 e posteriores).
- Instale o aplicativo no dispositivo (ou emulador do Android).
- Inicie o aplicativo no dispositivo.

Você pode usar o Adobe Flash Builder, Adobe Flash Professional CS5, ou as ferramentas de linha de comando para realizar essas etapas.

Uma vez que o aplicativo do AIR estiver pronto e compactado como um arquivo APK, você pode enviá-lo para o Android Market ou distribuí-lo através de outros meios.

AIR for iOS

As etapas a seguir são típicas no desenvolvimento de aplicativos AIR for iOS:

- Instale o iTunes.
- Gerar as IDs e os arquivos de desenvolvedor necessários no Portal de aprovisionamento Apple iOS. Estes itens incluem:
  - Certificado do desenvolvedor
  - ID do aplicativo
  - Perfil de aprovisionamento

  Você deve listar as identificações de todos os dispositivos de teste no qual pretende instalar o aplicativo ao criar o perfil de aprovisionamento.
- Converter o certificado de desenvolvimento e a chave privada para um arquivo de armazenamento de chave P12.
- Escreva o código MXML ou ActionScript do aplicativo.
- Compile o aplicativo com um compilador ActionScript ou MXML.
- Criar arte de ícone e de tela inicial do aplicativo.
- Crie o descritor do aplicativo (namespace, usando o 2.6 ou superior).

- Compacte o arquivo IPA usando ADT.
- Use o iTunes para colocar o seu perfil de aprovisionamento no dispositivo de teste.
- Instale e teste o aplicativo em seu dispositivo iOS. Você pode usar o iTunes ou o ADT por USB (suporte a USB no AIR 3.4 e superior) para instalar o arquivo IPA.

Assim que o aplicativo do AIR estiver concluído, você pode compactá-lo novamente usando um certificado de distribuição e perfil de aprovisionamento. Ele está pronto para enviado à Apple App Store.

Configuração de propriedades do aplicativo móvel

Tal como acontece com outros aplicativos AIR, você define as propriedades básicas do aplicativo no arquivo descritor do aplicativo. Os aplicativos móveis ignoram algumas das propriedades específicas para ambiente de trabalho, como tamanho da janela e transparência. Aplicativos móveis também podem usar as suas próprias propriedades específicas da plataforma. Por exemplo, você pode incluir um elemento `android` para aplicativos Android e um elemento `iPhone` para aplicativos iOS.

Configurações comuns

Várias configurações de descritor de aplicativo são importantes para todos os aplicativos de dispositivo móvel.

Versão de runtime exigida pelo AIR

Especifique a versão do runtime do AIR exigido pelo seu aplicativo usando o namespace do arquivo do descritor do aplicativo.

O namespace, atribuído no elemento do `aplicativo`, determina, em grande parte, que recursos seu aplicativo pode usar. Por exemplo, se seu aplicativo usa o namespace AIR 2.7, e o usuário tem alguma versão futura instalada, o aplicativo ainda verá o comportamento do AIR 2.7 (mesmo que o comportamento for alterado na versão futura). Somente quando mudar o namespace e publicar uma atualização sua aplicação terá acesso ao novo comportamento e recursos. As correções de segurança são uma importante exceção a essa regra.

Em dispositivos que usam um runtime separado do aplicativo, como o Android no AIR 3.6 e posteriores, será solicitado ao usuário que instale ou atualize AIR se não tiver a versão necessária. Em dispositivos que incorporam o runtime, como o iPhone, esta situação não ocorre (desde que a versão necessária seja fornecida com o aplicativo em primeiro lugar).

Nota: *(AIR 3.7 e superiores) Como padrão, o ADT empacota o runtime com aplicativos Android.*

Especifique o namespace usando o atributo xmlns do elemento do `aplicativo` raiz. Os namespaces a seguir devem ser usados para aplicativos móveis (dependendo da plataforma móvel visada):

```
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: *O suporte para dispositivos iOS 3 é fornecido pelo Packager para iPhone SDK, com base nos AIR 2.0 SDK. Para obter informações sobre a criação de aplicativos AIR for iOS 3, consulte Criação de aplicativos para o iPhone. O AIR 2.6 SDK (e posteriores) suporta iOS 4 e posteriores nos dispositivos iPhone 3Gs, iPhone 4 e iPad.*

Mais tópicos da Ajuda

"application" na página 213

Identidade do aplicativo

Diversas configurações devem ser exclusivas para cada aplicativo que você publicar. Estas incluem o ID, o nome, e o nome de arquivo.

IDs do aplicativo Android

No Android, o ID é convertida para o nome do pacote do Android prefixando "air". para o ID do AIR. Dessa forma, se o ID do AIR for *com.example.MyApp*, o nome do pacote do Android é air.com.example.MyApp.

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

Além disso, se o ID não for um nome de pacote válido no sistema operacional Android, é convertido para um nome válido. Caracteres com hífen são alterados para dígitos de sublinhado e de entrelinha em qualquer componente de ID que seja precedida por um "A" maiúsculo. Por exemplo, o ID *3-goats.1-boat* é transformada para o nome do pacote *air.A3_goats.A1_boat*.

Nota: *O prefixo adicionado à ID do aplicativo pode ser usado para identificar os aplicativos AIR no Android Market. Se você não deseja que o aplicativo seja identificado como AIR devido ao prefixo, deve desempacotar o arquivo APK, mudar o ID do aplicativo, e compactá-lo novamente como descrito em Emissor de análise do aplicativo do AIR for Android.*

IDs do aplicativo iOS

Defina o ID do aplicativo do AIR para corresponder com o ID do aplicativo que você criou no Portal de aprovisionamento Apple iOS.

As IDs do aplicativo iOS contêm um ID da distribuição do conjunto seguida por um identificador do conjunto. O ID da distribuição de conjunto é uma sequência de caracteres como, por exemplo, 5RM86Z4DJM, que a Apple atribui à ID de aplicativo. O identificador de conjunto contém o nome em estilo de domínio reverso selecionado. Um identificador de conjunto pode terminar em um asterisco (*), indicando um ID de aplicativo curinga. Se o identificador de conjunto terminar com caractere curinga, você pode substituir este por qualquer sequência válida.

Por exemplo:

- Se o ID do aplicativo Apple for `5RM86Z4DJM.com.example.helloWorld`, você deve usar `com.example.helloWorld` no descritor do aplicativo.
- Se o ID do aplicativo Apple for `96LPVWEASL.com.example.*` (um ID de aplicativo curinga), você pode usar `com.example.helloWorld`, `com.example.anotherApp`, ou algum outro ID que inicie com `com.example`.
- Finalmente, se o ID de aplicativo Apple for apenas o ID de distribuição de conjunto e um curinga, como: `38JE93KJL.*`, você pode usar qualquer ID de aplicativo em AIR.

Ao especificar o ID do aplicativo, não inclua o ID de distribuição de conjunto do ID de aplicativo.

Mais tópicos da Ajuda

"id" na página 230

"filename" na página 224

"name" na página 238

Versão do aplicativo

No AIR 2.5 e posterior, especifique a versão do aplicativo no elemento `versionNumber`. O elemento `version` não pode mais ser usado. Ao especificar um valor para versionNumber, você deve usar uma sequência de até três números separados por pontos, como: "0.1.2". Cada segmento do número de versão pode ter até três dígitos. (Em outras palavras, "999.999.999" é o maior número de versão autorizada.) Você não precisa incluir todos os três segmentos do número; "1" e "1.0" são números de versão legal.

Você também pode especificar um rótulo para a versão com o elemento `versionLabel`. Ao adicionar um rótulo de versão este é exibido em vez do número da versão em lugares como a tela de informações dos aplicativos Android. Uma etiqueta de versão deve ser especificada para aplicativos distribuídos através do Android Market. Se você não especificar um valor `versionLabel` no descritor de aplicativo do AIR, o valor `versionNumber` é atribuído ao campo de rótulo da versão Android.

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

No Android, o AIR `versionNumber` é traduzido para o Android inteiro `versionCode` usando a fórmula: `a*1000000 + b*1000 + c`, onde a, b, e c são os componentes do número da versão do AIR: `a.b.c`.

Mais tópicos da Ajuda

"version" na página 245

"versionLabel" na página 246

"versionNumber" na página 246

SWF do aplicativo principal

Especifique o arquivo SWF do aplicativo principal no filho `content` do elemento `initalWindow`. Ao direcionar dispositivos no perfil móvel, você deve usar um arquivo SWF (aplicativos com base HTML não estão disponíveis).

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

Você deve incluir o arquivo no pacote AIR (usando ADT ou sua IDE). Simplesmente fazer referência do nome no descritor do aplicativo não faz com que o arquivo seja incluído no pacote de maneira automática.

Propriedades da tela principal

Vários elementos filho do elemento initialWindow controlam a aparência inicial e o comportamento da tela principal do aplicativo.

- **aspectRatio** — especifica se o aplicativo deverá ser exibido inicialmente em um formato *retrato* (altura maior do que largura), um formato *paisagem* (altura menor do que largura) ou *qualquer* formato (o palco orienta automaticamente em todas as orientações).

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

- **autoOrients** — Especifica se o palco deve mudar automaticamente a orientação quando o usuário gira o aparelho ou executa outro gesto relacionado à orientação, como abertura ou fechamento de um teclado deslizante. Se *false*, que é o padrão, o palco não mudará a orientação com o dispositivo.

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

- **depthAndStencil** — Especifica o uso do buffer de profundidade ou de estêncil. Normalmente, você usa esses buffers ao trabalhar com conteúdo 3D.

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

- **fullScreen** — Especifica se o aplicativo deve tomar a tela completa do dispositivo, ou se deve compartilhar a tela com o cromo, como uma barra de status do sistema.

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

- **renderMode** — Especifica se o runtime deve renderizar o aplicativo com a unidade de processamento gráfico (GPU) ou a unidade central de processamento (CPU) principal. Em geral, a renderização da GPU aumentará a velocidade de renderização, mas alguns recursos, como certos modos de mesclagem e filtros PixelBender, não estão disponíveis no modo de GPU. Além disso, diversos dispositivos e drivers de dispositivo diferentes têm diferentes capacidades e limitações de GPU. Você sempre deve testar seu aplicativo em uma ampla variedade de dispositivos possíveis, especialmente quando usando o modo de GPU.

  Você pode definir o modo de renderização como *gpu*, *cpu*, *direct* ou *auto*. O valor padrão é *auto*, que no momento volta para o modo de cpu.

  Nota: *Para aproveitar a aceleração GPU do conteúdo Flash com o AIR para plataformas móveis, o Adobe recomenda que você use renderMode="direct", o Stage3D e não o renderMode="gpu". A Adobe oferece suporte e recomenda oficialmente as seguintes estruturas baseadas em Stage3D: Starling (2D) e Away3D (3D). Para obter mais detalhes do Stage3D e do Starling/Away3D (3D), consulte http://gaming.adobe.com/getstarted.*

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

  Nota: *Não é possível usar renderMode="direct" para aplicativos executados em segundo plano.*

  As limitações do modo de GPU são as seguintes:

  - A estrutura do Flex não suporta o modo de renderização de GPU.
  - Não existe suporte para filtros
  - Mesclagens PixelBender e preenchimentos não estão disponíveis
  - Não há suporte para os seguintes modos de mistura: camada, alfa, apagar, sobrepor, realçar, clarear e escurecer
  - Não é recomendado o uso do modo de renderização pela GPU em um aplicativo que reproduz vídeo.
  - No modo de renderização pela GPU, os campos de texto não são devidamente movidos para um local visível quando o teclado virtual é aberto. Para assegurar que o campo de texto seja visível enquanto o usuário insere texto, use a propriedade softKeyboardRect do palco e os eventos de teclado virtual para mover o campo de texto para a área visível.
  - Se um objeto de exibição não puder ser renderizado pela GPU, ele não é exibido. Por exemplo, se um filtro for aplicado para um objeto de exibição, este não é exibido.

  Nota: *A implementação de GPU para iOS no AIR 2.6+ é muito diferente da usada na versão AIR 2.0 anterior. Diferentes considerações de otimização aplicáveis.*

Mais tópicos da Ajuda

"aspectRatio" na página 216

"autoOrients" na página 217

"depthAndStencil" na página 220

"fullScreen" na página 228

"renderMode" na página 240

Perfis disponíveis

Você pode adicionar o elemento `supportedProfiles` para especificar com quais perfis de dispositivo seu aplicativo é compatível. Use o perfil mobileDevice para dispositivos móveis. Ao executar o aplicativo com o Adobe Debug Launcher (ADL), este usa o primeiro perfil da lista como o perfil ativo. Você também pode usar o sinalizador `-profile` ao executar o ADL para selecionar um perfil específico na lista de suporte. Se o aplicativo for executado em todos os perfis, você pode excluir o elemento `supportedProfiles`. O ADL usa o perfil desktop como o perfil padrão ativo neste caso.

Para especificar que o aplicativo é compatível tanto com os perfis desktop quanto com o dispositivo móvel, e que normalmente você quer testar o aplicativo no perfil móvel, adicione o seguinte elemento:

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

Mais tópicos da Ajuda

"supportedProfiles" na página 243

"Perfis de dispositivo" na página 249

"AIR Debug Launcher (ADL)" na página 162

Extensões nativas necessárias

Os aplicativos que suportam o perfil `mobileDevice` podem usar extensões nativas.

Indique todas as extensões nativas que o aplicativo do AIR utiliza no descritor do aplicativo. O seguinte exemplo ilustra a sintaxe para especificar duas extensões nativas necessárias:

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

O elemento `extensionID` tem o mesmo valor que o elemento `id` no arquivo descritor de extensão. O arquivo descritor da extensão é um arquivo XML chamado extension.xml. Ela está compactada no arquivo ANE recebido do desenvolvedor da extensão nativa.

Comportamento do teclado virtual

Defina o elemento `softKeyboardBehavior` como `none` para desativar o comportamento de deslocamento e redimensionamento automático que o runtime usa para verificar se o campo de entrada de texto está centrado na exibição quando o teclado virtual é gerado. Se você desativar o comportamento automático, seu aplicativo fica responsável por conferir se a área de entrada de texto ou outros conteúdos relevantes estão visíveis após o teclado ser ativado. Você pode usar a propriedade `softKeyboardRect` do palco em conjunto com SoftKeyboardEvent para detectar quando o teclado é aberto e determinar a área que fica oculta.

Para ativar o comportamento automático defina o valor do elemento para `pan`:

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

Visto que `pan` é o valor padrão, a omissão do elemento `softKeyboardBehavior` também ativa o comportamento automático do teclado.

Nota: *Ao usar também renderização pela GPU o comportamento de deslocamento não é compatível.*

Mais tópicos da Ajuda

"softKeyboardBehavior" na página 242

Stage.softKeyboardRect

SoftKeyboardEvent

Configurações do Android

Na plataforma Android você pode usar o elemento `android` do descritor do aplicativo para adicionar informações ao manifesto do aplicativo do Android, que é um arquivo de propriedades do aplicativo usado pelo sistema operacional Android. O ADT gera automaticamente o arquivo Manifest.xml Android quando você cria o pacote APK. O AIR define algumas propriedades para os valores necessários para determinados recursos funcionarem. Outras propriedades definidas na seção android do descritor do aplicativo do AIR são adicionadas à seção correspondente do arquivo Manifest.xml.

Nota: *Para a maioria dos aplicativos AIR você deve definir as permissões do Android necessárias para seu aplicativo no elemento* `android`*, mas geralmente não precisa definir as outras propriedades.*

Você só pode definir os atributos que aceitam valores boolianos, inteiros ou sequências de caracteres. Não está disponível a definição de referências para recursos do pacote de aplicativo.

Nota: *O tempo de execução requer uma versão mínima do SDK igual ou maior que 14. Se você deseja criar um aplicativo apenas para versões superiores, certifique-se de que o Manifesto inclui* `<uses-sdk android:minSdkVersion=""></uses-sdk>` *com a versão correta.*

Configurações de manifesto Android reservadas

O AIR define várias entradas de manifesto no documento de manifesto Android gerado para garantir que os recursos de aplicação e runtime funcionem corretamente. Não é possível definir as configurações a seguir:

elemento de manifesto

Não é possível definir os seguintes atributos do elemento de manifesto:

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

elemento de atividade

Não é possível definir os seguintes atributos do elemento de atividade principal:

- android:label
- android:icon

elemento de aplicativo

Não é possível definir os seguintes atributos do elemento de aplicativo:

- android:theme
- android:name
- android:label

- android:windowSoftInputMode
- android:configChanges
- android:screenOrientation
- android:launchMode

Permissões do Android

O modelo de segurança do Android exige que cada aplicativo peça autorização para utilizar recursos que tenham implicações de segurança ou privacidade. Essas permissões devem ser especificadas quando o aplicativo é compactado, e não podem ser alteradas em runtime. O sistema operacional Android informa ao usuário quais permissões um aplicativo solicita quando o usuário o instala. Se a permissão necessária para um recurso não for solicitada, o sistema operacional Android pode gerar uma exceção quando o aplicativos acessa o recurso, mas não é garantida uma exceção. Exceções são transmitidas pelo runtime ao seu aplicativo. No caso de falha silenciosa, uma mensagem de falha de permissão é adicionada ao log do sistema Android.

No AIR, você pode especificar as permissões do Android dentro do elemento `Android` do descritor do aplicativo. O formato a seguir é usado para adicionar as permissões (onde PERMISSION_NAME é o nome de uma permissão Android):

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

As instruções de permissões por uso dentro do elemento `manifest` são adicionadas diretamente para o documento de manifesto do Android.

As permissões a seguir são obrigatórias para usar vários recursos AIR:

ACCESS_COARSE_LOCATION Permite ao aplicativo acessar dados de local de rede WIFI e celular por meio da classe Geolocation.

ACCESS_FINE_LOCATION Permite ao aplicativo acessar dados GPS por meio da classe Geolocation.

ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE Permite ao aplicativo acessar informações de rede por meio da classe NetworkInfo.

CAMERA Permite ao aplicativo acessar a câmera.

Nota: *Ao pedir permissão para usar o recurso de câmera, o Android assume que seu aplicativo também exige a câmera. Se a câmera for um recurso opcional do aplicativo, você deve adicionar um elemento* `recurso por uso` *para o manifesto para a câmera, definindo o atributo necessário para* `false`*. Consulte "Filtro de compatibilidade do Android" na página 79.*

INTERNET Permite ao aplicativo fazer solicitações de rede, além de permitir a depuração remota.

READ_PHONE_STATE Permite ao runtime do AIR silenciar o áudio durante as chamadas telefônicas. Você deve definir essa permissão se o aplicativo reproduzir áudio em plano de fundo.

RECORD_AUDIO Permite ao aplicativo acessar o microfone.

WAKE_LOCK e DISABLE_KEYGUARD Permite ao aplicativo impedir que o dispositivo entre no modo de suspensão por meio do uso das configurações da classe SystemIdleMode.

WRITE_EXTERNAL_STORAGE Permite ao aplicativo gravar no cartão de memória externo no dispositivo.

Por exemplo, para definir as permissões para um aplicativo que exige cada permissão de maneira impressionante, você pode adicionar o seguinte para o descritor de aplicativo:

```
<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>
```

Mais tópicos da Ajuda

Segurança e permissões do Android

Classe Manifest.permission do Android

Esquemas URI personalizados do Android

Você pode usar um esquema URI personalizado para ativar um aplicativo do AIR de uma página web ou um aplicativo Android nativo. O suporte URI personalizado depende de filtros de método especificados no manifesto do Android, por isso esta técnica não pode ser usada em outras plataformas.

Para usar um URI personalizado, adicione um filtro com método para o descritor do aplicativo no bloco `<android>`. Os elementos `intent-filter` no exemplo a seguir devem ser especificados. Edite a instrução `<data android:scheme="`*`my-customuri`*`"/>` para refletir a sequência de caracteres URI para o seu 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>
```

O filtro com método informa o sistema operacional Android que seu aplicativo está disponível para executar uma determinada ação. No caso de um URI personalizado, isto significa que o usuário clicou em um link usando esse esquema de URI (e o navegador não sabe como lidar com isso).

Quando o aplicativo é invocado através de um URI personalizado, o objeto NativeApplication envia um evento `invoke`. A URL do link, incluindo parâmetros de consulta, é colocada no array `arguments` do objeto InvokeEvent. Você pode usar qualquer número de filtros com método.

Nota: *Links em uma instância StageWebView não podem abrir URLs que usam um esquema de URI customizado.*

Mais tópicos da Ajuda

Filtros com método Android

Categorias e ações do Android

Filtro de compatibilidade do Android

O sistema operacional Android usa um número de elementos no arquivo de manifesto do aplicativo para determinar se o aplicativo é compatível com um determinado dispositivo. A inclusão desta informação ao manifesto é opcional. Se você não incluir esses elementos, o aplicativo pode ser instalado em qualquer dispositivo com Android. No entanto, ele pode não funcionar corretamente em qualquer dispositivo com Android. Por exemplo, um aplicativo de câmera não vai ser muito útil em um telefone que não tenha câmera.

As marcas do manifesto do Android que você pode usar para filtrar incluem:

- supports-screens
- uses-configuration
- uses-feature
- uses-sdk (no AIR 3 e superior)

Aplicativos de câmera

Se você pedir a permissão da câmera para seu aplicativo, o Android assume que o aplicativo exige que todos os recursos da câmera estejam disponíveis, incluindo foco automático e flash. Se seu aplicativo não exige que todos os recursos da câmera, ou se a câmera é um recurso opcional, você deve definir os vários elementos `uses-feature` para a câmera para indicar que estes são opcionais. Caso contrário, usuários com dispositivos que estão faltando um recurso ou que não tenham uma câmera não poderá encontrar o seu aplicativo na Android Market.

O exemplo a seguir ilustra como solicitar permissão para a câmera e tornar todos os recursos desta opcionais:

```
<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>
```

Aplicativos de gravação de áudio

Se você pedir a permissão para gravar o áudio, o Android também assume que o aplicativo requer um microfone. Se a gravação de áudio for um recurso opcional do seu aplicativo, você pode adicionar uma marca uses-feature para especificar que o microfone não é necessário. Caso contrário, usuários com dispositivos que não têm um microfone não poderão encontrar seu aplicativo na Android Market.

O exemplo a seguir ilustra como solicitar permissão para usar o microfone, enquanto ainda está tornando o hardware do microfone 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>
```

Mais tópicos da Ajuda

Desenvolvedores para Android: Compatibilidade do Android

Desenvolvedores para Android: Constantes do nome do recurso do Android

Local de instalação

Você pode permitir que seu aplicativo seja instalado ou transferido para o cartão de memória externo, definindo o atributo `installLocation` do elemento `manifest` do Android para `auto` ou `preferExternal`:

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

O sistema operacional Android não garante que seu aplicativo será instalado para a memória externa. Um usuário pode também mover o aplicativo entre memória interna e externa utilizando aplicativo de configurações do sistema.

Mesmo quando instalado para memória externa, cache de aplicativos e dados do usuário (como o conteúdo do diretório de armazenamento do aplicativo, objetos compartilhados e arquivos temporários) ainda estão armazenados na memória interna. Para evitar o uso de muita memória interna, seja seletivo sobre os dados que você salvar para o diretório de armazenamento do aplicativo. Grandes quantidades de dados devem ser salvos no SDCard usando os locais `File.userDirectory` ou `File.documentsDirectory` (ambos mapeiam para a raiz do cartão SD no Android).

Ativar Flash Player e outros plug-ins em um objeto de StageWebView

No Android 3.0+, um aplicativo deve habilitar a aceleração de hardware no elemento do aplicativo Android para que o conteúdo do plug-in seja exibido num objeto StageWebView. Para ativar a renderização de plug-in, defina o atributo `android:hardwareAccelerated` do elemento `application` como `true`:

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

Profundidade de cores

AIR 3+

No AIR 3 e em versões posteriores, o runtime define a exibição para renderizar cores de 32 bits. Nas versões anteriores do AIR, o runtime usa a cor de 16 bits. Você pode informar o runtime para usar a cor de 16 bits usando o elemento <colorDepth> do descritor de aplicativo:

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

A utilização da profundidade de cor de 16 bits pode aumentar o desempenho de renderização, mas em detrimento da fidelidade de cor.

Configurações do iOS

As configurações aplicáveis somente aos dispositivos iOS são colocadas no elemento `<iPhone>` no descritor do aplicativo. O elemento `iPhone` pode ter um elemento `InfoAdditions`, um elemento `requestedDisplayResolution`, um elemento `Entitlements`, um elemento `externalSwfs` e um elemento `forceCPURenderModeForDevices` como filhos.

O elemento `InfoAdditions` permite especificar pares de valores chave que são adicionados ao arquivo de configurações Info.plist do aplicativo. Nesse exemplo, os valores definem o estilo da barra de estilo do aplicativo e determinam que o aplicativo não requer acesso WiFi contínuo.

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

As configurações de InfoAdditions são anexadas a uma tag `CDATA`.

O elemento `Entitlements` permite especificar pares de valores chave que são adicionados ao arquivo de configurações Entitlements.plist do aplicativo. As configurações Entitlements.plist permitem que o aplicativo acesse determinados recursos do iOS, como por exemplo notificações por push.

Para obter informações mais detalhadas sobre as configurações de Info.plist e Entitlements.plist, consulte a documentação do desenvolvedor da Apple.

Tarefas em segundo plano de suporte no iOS

AIR 3.3

Adobe AIR 3.3 e superior oferece suporte a multitarefa no iOS ativando determinados comportamentos de segundo plano:

- Áudio
- Atualizações de local
- Sistemas de rede
- Cancelar a execução de aplicativos em segundo plano

Nota: *Com o swf versão 21 e suas versões anteriores, o AIR não oferece suporte a execução em segundo plano em iOS e Android quando o modo de renderização estiver definido como direto. Devido a essa restrição, os aplicativos baseados em Stage3D não conseguem executar tarefas em segundo plano, como reprodução de áudio, atualizações de locais, upload ou download de rede etc. O iOS não permite renderização de chamadas ou OpenGLES em segundo plano. Aplicativos que tentam fazer chamadas OpenGL em segundo plano são encerrados pelo iOS. O Android não restringe aplicativos de realizar chamadas OpenGLES em segundo plano ou outra tarefa em segundo plano, como reprodução de áudio. Com o swf versão 22 e posteriores, os aplicativos móveis do AIR podem ser executados em segundo plano quando o renderMode estiver definido como direto. O runtime do iOS AIR resulta em um erro de ActionScript (3768 - a API Stage3D não pode ser usada durante a execução em segundo plano) se as chamadas de OpenGLES forem executadas em segundo plano. Entretanto, não há erros em Android pois seus aplicativos nativos podem realizar chamadas de OpenGLES em segundo plano. Para o uso ideal do recurso remoto, não faça chamadas de renderização enquanto um aplicativo é executado em segundo plano.*

Áudio de fundo

Para ativar a reprodução e a gravação de áudio de fundo, inclua o seguinte par de valores-chave no elemento `InfoAdditions`:

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

Atualizações de local de fundo

Para ativar atualizações de local de fundo, inclua o seguinte par de valores-chave no elemento `InfoAdditions`:

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

Nota: *Use esse recurso apenas quando necessário, já que APIs de local são um consumidor significativo de bateria.*

Rede de fundo

Para executar tarefas curtas em segundo plano, seu aplicativo define a propriedade `NativeApplication.nativeApplication.executeInBackground` como `true`.

Por exemplo, seu aplicativo pode iniciar uma operação de carregamento de arquivo, depois da qual o usuário move outro aplicativo para a frente. Quando o aplicativo recebe um evento de conclusão de carregamento, ele pode definir `NativeApplication.nativeApplication.executeInBackground` como `false`.

A definição da propriedade `NativeApplication.nativeApplication.executeInBackground` como `true` não garante que o aplicativo será executado indefinidamente, já que o iOS impõe um limite de tempo para tarefas em segundo plano. Quando o iOS interrompe o processamento em segundo plano, o AIR despacha o evento `NativeApplication.suspend`.

Cancelar a execução em segundo plano

Seu aplicativo pode explicitamente cancelar a execução em segundo plano incluindo o seguinte par de valores-chave no elemento `InfoAdditions`:

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

Configurações reservadas de InfoAdditions do iOS

O AIR define várias entradas no arquivo INfo.plist gerado para garantir que os recursos do aplicativo e o runtime funcionem corretamente. Não é possível definir as configurações a seguir:

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

***Nota:** É possível definir o MinimumOSVersion. A definição de MinimumOSVersion está presente no Air 3.3 e em versões posteriores.*

Suporte a modelos diferentes de dispositivos iOS

Para oferecer suporte a iPad, inclua as configurações adequadas de valores chave para `UIDeviceFamily` no elemento `InfoAdditions`. A configuração `UIDeviceFamily` é uma matriz de sequências de caracteres. Cada sequência de caracteres define os dispositivos suportados. O parâmetro `<string>1</string>` define o suporte para iPhone e iPod Touch. O ajuste `<string>2</string>` define o suporte para o iPad. O ajuste `<string>3</string>` define o suporte para o tvOS. Se você especificar somente uma dessas sequências de caracteres, somente aquela família de dispositivos será suportada. Por exemplo, a configuração a seguir limita o suporte ao iPad:

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

As configurações a seguir suportam as duas famílias de dispositivos (iPhone/iPod Touch e iPad):

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

Além disso, no AIR 3.7 e superiores, é possível usar a tag `forceCPURenderModeForDevices` para forçar a CPU a entrar no modo de renderização para um conjunto de dispositivos especificado e ativar o modo de renderização da GPU para os dispositivos iOS restantes.

Adicione esta tag como filha da tag `iPhone` e especifique uma lista separada por espaços com os nomes dos modelos de dispositivos. Para obter uma lista de nomes de modelos de dispositivos válidos, consulte “forceCPURenderModeForDevices” na página 227.

Por exemplo, para usar o modo de CPU em iPods, iPhones e iPads antigos e ativar o modo de GPU para todos os demais dispositivos, especifique o seguinte no descritor do aplicativo:

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

Exibições em alta resolução

O elemento`requestedDisplayResolution` especifica se o seu aplicativo usará o modo de resolução *padrão* ou *alta* nos dispositivos iOS com tela de alta resolução.

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

No modo de alta resolução, você pode tratar individualmente cada pixel numa exibição de alta resolução. No modo padrão, a tela do dispositivo aparecerá para o seu aplicativo como uma tela de resolução padrão. O desenho de um único pixel nesse modo definirá a cor de quatro pixel na tela de alta resolução.

A definição proposta pelo aplicativo é `padrão`. Observe que, para o direcionamento de dispositivos iOS, você utiliza o elemento `requestedDisplayResolution` como um filho do elemento `iPhone` (não os elementos `InfoAdditions` ou `initialWindow`).

Se desejar utilizar configurações diferentes em dispositivos diferentes, especifique o seu valor padrão como o valor do elemento `requestedDisplayResolution`. Use o atributo `excludeDevices` para especificar dispositivos que devem usar o valor oposto. Por exemplo, com o código a seguir, o modo de alta resolução é usado para todos os dispositivos compatíveis, exceto os iPads de 3ª geração, que usam o modo padrão.

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

O atributo `excludeDevices` está disponível no AIR 3.6 e posterior.

Mais tópicos da Ajuda

"requestedDisplayResolution" na página 240

Renaun Erickson: Desenvolvimento para telas iOS com retina e sem retina utilizando o AIR 2.6

Esquemas URI personalizados do iOS

Você pode registrar um esquema URI personalizado para permitir que o seu aplicativo seja invocado por um link numa página da web ou em um outro aplicativo nativo no dispositivo. Para registrar um esquema URI, adicione uma chave CFBundleURLTypes no elemento InfoAdditions. O exemplo a seguir registra um esquema URI denominado *com.example.app* para permitir que o aplicativo seja invocado por URLs com a 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>
```

Quando o aplicativo é invocado através de um URI personalizado, o objeto NativeApplication envia um evento `invoke`. A URL do link, incluindo parâmetros de consulta, é colocada no array `arguments` do objeto InvokeEvent. Você pode usar qualquer número de esquemas URI personalizados.

Nota: *Links em uma instância StageWebView não podem abrir URLs que usam um esquema de URI customizado.*

Nota: *Se outro aplicativo já tiver registrado um esquema, então o seu aplicativo não poderá substituí-lo como um aplicativo registrado para aquele esquema URI.*

Filtragem de compatibilidade com o iOS

Adicione entradas a uma matriz UIRequiredDeviceCapabilities dentro do elemento `InfoAdditions` se o seu aplicativo tiver que ser usado somente em dispositivos com capacidades específicas de hardware ou software. Por exemplo, a entrada a seguir indica que um aplicativo requer uma câmara fotográfica digital e um microfone:

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

Se um dispositivo não tiver a capacidade correspondente, o aplicativo não poderá ser instalado. As configurações de capacidade relevantes para os aplicativos AIR incluem:

telefonia	câmara com flash
wifi	câmara de vídeo
sms	acelerômetro
câmara fotográfica digital	serviços de localização
câmara com autofoco	gps
câmara voltada para frente	microfone

O AIR 2.6+ adiciona automaticamente *armv7* e *opengles-2* à lista de capacidades requeridas.

Nota: *Não é necessário incluir essas capacidades no descritor do aplicativo para que o seu aplicativo as utilize. Use as configurações de UIRequiredDeviceCapabilities somente para impedir que os usuários instalem seu aplicativo em dispositivos nos quais não possam funcionar adequadamente.*

Sair em vez de pausar

Quando um usuário sai de um aplicativo do AIR, o aplicativo vai para o segundo plano e pausa. Se você quiser que o seu aplicativo seja encerrado completamente, em vez de pausar, defina a propriedade `UIApplicationExitsOnSuspend` como `YES`:

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

Minimizar o tamanho do download carregando SWFs externos com somente ativos

AIR 3.7

Você pode minimizar o tamanho inicial de download do aplicativo ao empacotar um subconjunto dos SWFs usados por seu aplicativo e carregar os SWFs externos restantes (apenas ativos) no runtime usando o método `Loader.load()`. Para usar esse recurso, é necessário empacotar o aplicativo para que o ADT mova todo o ActionScript ByteCode (ABC) dos arquivos SWF carregados externamente para o aplicativo SWF principal, deixando um arquivo SWF que contenha todos os recursos. Isso é feito para obedecer à regra da Apple Store que proíbe o download de códigos depois que um aplicativo é instalado.

O ADT realiza o seguinte procedimento para ser compatível com os SWFs carregados externamente (também chamados de SWFs sem código):

- Lê o arquivo de texto especificado no sub-elemento `<externalSwfs>` do elemento `<iPhone>` para acessar a lista determinada por linhas dos SWFs que serão carregados no momento de execução:

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

- Transfere o código ABC de cada SWF carregado externamente para o executável principal.
- Omite o SWF carregado externamente do arquivo .ipa.
- Copia os SWFs sem código para o diretório `.remoteStrippedSWFs`. Você hospeda esses SWFs em um servidor da Web e seu aplicativo os carrega, conforme necessário, no runtime.

Você indica que os arquivos SWF devem ser carregados no runtime especificando seus nomes, um por linha em um arquivo de texto, conforme o exemplo a seguir:

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

O caminho do arquivo especificado é relativo ao arquivo do descritor do aplicativo. Além disso, esses SWFs devem ser especificados como ativos no comando `adt`.

***Nota:** Esse recurso se aplica apenas ao empacotamento padrão. Para um empacotamento rápido (usando por exemplo, o intérprete, o simulador ou o depurador) o ADT não cria SWFs sem código.*

Para obter mais informações sobre esse recurso, incluindo amostras de código, consulte Hospedagem externa de SWFs secundários para os aplicativos AIR no iOS, uma postagem de blog do engenheiro da Adobe, Abhinav Dhandh.

Suporte à geolocalização

Para usar o suporte à geolocalização, adicione um dos seguintes pares de valores-chave ao 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>
```

Ícones de aplicativos

A tabela a seguir lista os tamanhos de ícones utilizados em cada plataforma móvel:

Tamanho do ícone	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
512 x 512	Android, iOS
1024 x 1024	iOS

Especifique o caminho para os arquivos de ícone no elemento ícone do arquivo descritor do aplicativo:

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

Se você não fornecer um ícone de determinado tamanho, o maior tamanho seguinte é utilizado e adaptado para se ajustar.

Ícones no Android

No Android, os ícones especificados no descritor do aplicativo são usados como o ícone Launcher do aplicativo. O ícone de inicialização do aplicativo deve ser fornecido como um conjunto de imagens PNG de 36x36, 48x48, 72x72, 96x96, 144x144 e 192x192 pixels. Estes tamanhos de ícones são utilizados para telas de baixa, média e alta densidade, respectivamente.

É preciso que os desenvolvedores enviem o ícone de 512x512 pixels no momento do envio do aplicativo na Play Store do Google.

Ícones do iOS

Os ícones definidos no descritor do aplicativo são usados nos seguintes locais para um aplicativo iOS:

- Um ícone de 29x29 pixels — ícone de pesquisa Spotlight para iPhones/iPods de resolução mais baixa e ícone Configurações para iPads de resolução mais baixa.
- Um ícone de 40 x 40 pixels: ícone de pesquisa do Spotlight para iPads de resolução mais baixa.
- Um ícone de 48x48 pixels — o AIR adicione uma borda a essa imagem e a usa como um ícone de 50x50 para pesquisa Spotlight em iPads de resolução mais baixa.
- Um ícone de 50x50 pixels — pesquisa Spotlight para iPads de resolução mais baixa.
- Um ícone de 57x57 pixels — ícone de aplicativo para iPhones/iPods de resolução mais baixa.
- Um ícone de 58x58 pixels — ícone Spotlight para iPhones/iPods com tela Retina e ícone de configurações para iPads com tela Retina.
- Um ícone de 60 x 60 pixels: ícone de aplicativo para iPhones/iPods de resolução mais baixa.
- Um ícone de 72 x 72 pixels (opcional): ícone de aplicativo para iPads de resolução mais baixa.
- Um ícone de 76 x 76 pixels (opcional): ícone de aplicativo para iPads de resolução mais baixa.
- Um ícone de 80 x 80 pixels: pesquisa do Spotlight para iPhones/iPods/iPads de resolução alta.
- Um ícone de 100x100 pixels — pesquisa Spotlight para iPads com tela Retina.
- Um ícone de 114 x 114 pixels: ícone de aplicativo para iPhone/iPods com tela Retina.
- Um ícone de 120 x 120 pixels: ícone de aplicativo para iPhones/iPods de resolução alta.
- Um ícone de 152 x 152 pixels: ícone de aplicativo para iPads de resolução alta.
- Um ícone de 167 x 167 pixels: ícone de aplicativo para iPads Pro de resolução alta.
- Um ícone de 512x512 pixels — ícone de aplicativos para iPhones/iPods/iPads de resolução mais baixa). O iTunes exibe esse ícone. O arquivo de 512 pixels PNG é utilizado somente para testar versões de desenvolvimento do aplicativo. Ao enviar o aplicativo final para a Apple App Store, envie a imagem de 512 pixels separadamente, como arquivo JPG. Isso não está incluído no IPA.
- Um ícone de 1024x1024 pixels — ícone de aplicativos para iPhones/iPods/iPads com tela Retina.

O iOS adiciona o efeito de brilho ao ícone. Não é necessário aplicar efeito à sua imagem original. Para remover o efeito de brilho padrão, adicione o seguinte ao elemento `InfoAdditions` no arquivo do descritor do aplicativo:

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

Nota: *No iOS, os metadados do aplicativo são inseridos como metadados png nos ícones do aplicativo para que o Adobe possa controlar o número de aplicativos AIR disponíveis na app store do Apple iOS. Se você não quiser que o aplicativo seja identificado como AIR devido a esses metadados de ícone, deverá descompactar o arquivo IPA, remover o metadado do ícone e reempacotá-lo. Esse procedimento é descrito no artigo Optar por não obter análises de aplicativos AIR for iOS.*

Mais tópicos da Ajuda

"icon" na página 229

"imageNxN" na página 230

Desenvolvedores para Android: Orientações de design de ícone

Diretrizes para a Interface Humana do iOS: Diretrizes para a Criação de Imagens e Ícones Customizados

Imagens de ativação do iOS

Além dos ícones de aplicativos, você também deverá fornecer pelo menos uma imagem de ativação chamada *Default.png*. Opcionalmente, você pode incluir imagens de ativação separadas para diferentes orientações iniciais, diferentes resoluções (incluindo a tela Retina de alta resolução e taxa de proporção de 16:9) e diferentes dispositivos. Você também pode incluir diferentes imagens de ativação a serem usadas quando seu aplicativo for invocado por um URL.

Os arquivos de imagem de ativação não são referidos na descrição do aplicativo e, portanto, devem ser colocados no diretório raiz do aplicativo. (*Não* coloque os arquivos em um subdiretório.)

Esquema de definição do nome de arquivo

Nome da imagem de acordo com o seguinte esquema:

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

A parte *basename* do nome do arquivo e a única parte obrigatória. Ela é *Default (Padrão)* (com D maiúsculo) ou o nome que você especificar usando a chave `UILaunchImageFile` no elemento `InfoAdditions` no descritor do aplicativo.

A parte *screen size modifier* designa o tamanho da tela quando ela não for um dos tamanhos de tela padrão. Esse modificador se aplica aos modelos do iPhone e do iPod Touch com telas com 16:9 de taxa de proporção, como o iPhone 5 e o iPod Touch (5ª geração). O único valor suportado para esse modificador é `-568h`. Como esses dispositivos suportam telas (retina) de alta resolução, o modificador do tamanho da tela é sempre usado com uma imagem que também tenha o modificador de escala `@2x`. O nome completo padrão da imagem de ativação desses dispositivos é `Default-568h@2x.png`.

A parte *urischeme* é a sequência de caracteres usada para identificar o esquema URI. Essa parte só se aplica se o seu aplicativo oferecer suporte a um ou mais esquemas de URL. Por exemplo, se o seu aplicativo puder ser invocado por um link como `example://foo`, então use `-example` como a parte do esquema do nome do arquivo de imagem de ativação.

A parte *orientation* oferece uma forma de especificar várias imagens de ativação para usar dependendo da orientação do dispositivo quando o aplicativo for ativado. Essa parte só se aplica a imagens para aplicativos do iPad. Ela pode ser um dos seguintes valores, indicando a orientação que o dispositivo usa quando o aplicativo é inicializado:

- -Retrato
- -PortraitUpsideDown (retrato, de cima para baixo)
- -Paisagem
- -LandscapeLeft (paisagem, esquerda)
- -LandscapeRight (paisagem, direita)

A parte da *escala* é `@2x` (para iPhone 4, iPhone 5 e iPhone 6) ou `@3x` (para iPhone 6 plus) para imagens de lançamento usadas em telas (de retina) de alta resolução. (Omita a parte escala completamente para as imagens usadas em exibições com resolução padrão.) Para imagens de ativação de dispositivos mais altos, como o iPhone 5 e o iPod Touch (5ª geração), você também deve especificar o modificador de tamanho da tela `-568h` depois da parte basename e antes de qualquer outra parte.

A parte *device* é usada para designar imagens de ativação para dispositivos portáteis e telefones. Essa parte é usada quando o seu aplicativo é um aplicativo universal com suporte para dispositivos portáteis e tablets com um único código binário de aplicativo. O valor possível deve ser `~ipad` ou `~iphone` (para iPhone e iPod Touch).

Para o iPhone, só é possível incluir imagens na proporção de retrato. No entanto, no caso do iPhone 6 plus, as imagens de paisagens também podem ser adicionadas. Use imagens com 320x480 pixels para dispositivos de resolução padrão, imagens com 640x960 pixels para dispositivos de alta resolução e imagens com 640x1136 pixels para dispositivos com 16:9 de taxa de proporção como o iPhone 5 e o iPod Touch (5ª geração).

Para iPad, inclua imagens da seguinte maneira:

- AIR 3.3 ou anterior - Sem imagem em tela inteira: inclui imagens em orientação paisagem, 1024x748 para resolução normal, 2048x 1496 para alta resolução e retrato, 768x1004 para resolução normal, 1536x 2008 para alta resolução.
- AIR 3.4 ou posterior - com imagens full screen: inclui imagens com orientação paisagem 1024x768 para resolução normal, 2048x 1536 para alta resolução, e orientação retrato 768x1024 para resolução normal, 1536x 2048 para alta resolução. Observe que quando você coloca uma imagem de tela inteira em um aplicativo sem tela inteira os 20 pixels superiores (40 pixels para resolução alta) são cobertos pela barra de status. Evite exibir informações importantes nesta área.

Exemplos

A tabela a seguir mostra um conjunto de exemplos de imagens que você poderia incluir num aplicativo hipotético que suportasse a mais ampla gama de dispositivos e orientações e pudesse ser ativado com URLs que utilizasse o esquema `example://`:

Nome do arquivo	Tamanho da imagem	Uso
Default.png	320 x 480	iPhone, resolução padrão
Default@2x.png	640 x 960	iPhone, alta resolução
Default-568h@2x.png	640 x 1136	iPhone, alta resolução, taxa de proporção de 16:9
Default-Portrait.png	768 x 1004 (AIR 3.3 e anteriores) 768 x 1024 (AIR 3.4 e superiores)	iPad, orientação de retrato

Nome do arquivo	Tamanho da imagem	Uso
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 e anteriores) 1536 x 2048 (AIR 3.4 e superiores)	iPad, alta resolução, orientação de retrato
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 e anteriores) 768 x 1024 (AIR 3.4 e superiores)	iPad, orientação de retrato de cima para baixo
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 e anteriores) 1536 x 2048 (AIR 3.4 e superiores)	iPad, alta resolução, orientação de retrato de cima para baixo
Default-Landscape.png	1024 x 768	iPad, orientação de paisagem à esquerda
Default-LandscapeLeft@2x.png	2048 x 1536	iPad, alta resolução, orientação de paisagem à esquerda
Default-LandscapeRight.png	1024 x 768	iPad, orientação de paisagem à direita
Default-LandscapeRight@2x.png	2048 x 1536	iPad, alta resolução, orientação de paisagem à direita
Default-example.png	320 x 480	exemplo:// URL no iPhone padrão
Default-example@2x.png	640 x 960	exemplo:// URL no iPhone de alta resolução
Default-example~ipad.png	768 x 1004	exemplo:// URL no iPad em orientações de retrato
Default-example-Landscape.png	1024 x 768	exemplo:// URL no iPad em orientações de paisagem

Esse exemplo ilustra somente uma abordagem. Você poderá, por exemplo, usar a imagem `Default.png` para o iPad e especificar imagens de lançamento específicas para iPhone e iPod com `Default~iphone.png` e `Default@2x~iphone.png`.

Consulte também

Guia de Programação de Aplicativos em iOS: Imagens de Lançamento de Aplicativo

Imagens de lançamento que criam pacotes para dispositivos iOS

Dispositivos	Resolução (pixels)	Nome da imagem de lançamento	Orientação
iPhones			
iPhone 4 (sem tela de retina)	640 x 960	Default~iphone.png	Foto
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Foto
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	Foto
iPhone 6, iPhone 7	750 x 1334	Default-375w-667h@2x~iphone.png	Foto
iPhone 6+, iPhone 7+	1242 x 2208	Default-414w-736h@3x~iphone.png	Foto
iPhone 6+, iPhone 7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	Paisagem

iPads			
iPad 1, 2	768 x 1024	Default-Portrait~ipad.png	Foto
iPad 1, 2	768 x 1024	Default-PortraitUpsideDown~ipad.png	Foto invertida
iPad 1, 2	1024 x 768	Default-Landscape~ipad.png	Paisagem à esquerda
iPad 1, 2	1024 x 768	Default-LandscapeRight~ipad.png	Paisagem à direita
iPad 3, Air	1536 x 2048	Default-Portrait@2x~ipad.png	Foto
iPad 3, Air	1536 x 2048	Default-PortraitUpsideDown@2x~ipad.png	Foto invertida
iPad 3, Air	2048 x 1536	Default-LandscapeLeft@2x~ipad.png	Paisagem à esquerda
iPad 3, Air	2048 x 1536	Default-LandscapeRight@2x~ipad.png	Paisagem à direita
iPad Pro	2048 x 2732	Default-Portrait@2x.png	Retrato
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Paisagem

Recomendações de arte

É possível criar qualquer arte desejada para um arquivo de imagem, desde que ela tenha as dimensões corretas. No entanto, frequentemente é melhor que a imagem do arquivo coincida com o estado inicial do aplicativo. Você pode criar essa imagem de ativação obtendo uma captura de tela da imagem de inicialização do seu aplicativo:

1. Abra o seu aplicativo no dispositivo iOS. Quando a primeira tela da interface do usuário aparecer, pressione e mantenha pressionado o botão Início (abaixo da tela). Mantendo pressionado o botão Início, pressione o botão Despertar/Repousar (na parte superior do dispositivo). Isso executa uma captura de tela e envia-a para o rolo da câmera.
2. Transfira a imagem para seu computador de desenvolvimento, transferindo as fotos do iPhoto ou de outro aplicativo de transferência de fotos.

Não inclua texto na imagem de ativação se o aplicativo estiver localizado em vários idiomas. A imagem de ativação é estática, e o texto não coincidiria com outros idiomas.

Consulte também

Diretrizes de Interface do iOS: Imagens de lançamento

Configurações ignoradas

Aplicativos em dispositivos móveis ignoram as configurações de aplicativo que se aplicam às janelas nativas, ou aos recursos do sistema operacional de área de trabalho. As configurações ignoradas são:

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize

- programMenuFolder
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

Compactação de um aplicativo do AIR móvel

Use o comando ADT -package para criar o pacote de aplicativo para um aplicativo do AIR destinado a um dispositivo móvel. O parâmetro -target especifica a plataforma móvel para A qual o pacote é criado.

Pacotes do Android

Os aplicativos AIR no Android usam o formato do pacote de aplicativos deste (APK), ao invés do formato do pacote AIR.

Os pacotes produzidos pela ADT usando o tipo de destino *APK* estão em um formato que pode ser enviado para o Android Market. O Android Market tem exigências que os aplicativos apresentados devem satisfazer para serem aceitos. Você deve rever as mais recentes exigências antes de criar o pacote final. Consulte Desenvolvedores para Android: Publicação no Market.

Diferentemente dos aplicativos iOS, você pode usar um certificado de assinatura de código AIR normal para assinar seu aplicativo Android; contudo, para apresentar um aplicativo para o Android Market, o certificado deve obedecer às regras do Market, que exigem que o certificado seja válido pelo menos até 2033. Você pode criar um certificado usando o comando -certificate ADT.

Para enviar um aplicativo para um mercado alternativo que não permite que o seu aplicativo solicite um download do AIR proveniente de um mercado do Google, você pode especificar um URL de download alternativo usando o parâmetro de ADT `-airDownloadURL`. Quando um usuário que não tenha solicitado uma versão do runtime do AIR iniciar o seu aplicativo, ele será direcionado para um URL especificado. Consulte “Comando package do ADT” na página 169 para obter mais informações.

Por padrão, o ADT cria pacotes do aplicativo Android com tempo de execução compartilhado. Para executar o aplicativo, o usuário precisa instalar o tempo de execução do AIR separado no dispositivo.

Nota: *Para forçar o ADT a criar uma APK que use tempo de execução cativo, use* `target apk-captive-runtime`.

Pacotes iOS

Os aplicativos AIR no iOS usam o formado de pacote do iOS (IPA), em vez do formato nativo do AIR.

Pacotes produzidos pelo ADT que usam o tipo de destino *ipa-app-store* e o certificado de assinatura com código correto e perfil de aprovisionamento estão no formato que pode ser enviado para a App Store da Apple. Use o tipo de destino *ipa-ad-hoc* para compactar um aplicativo para distribuição ad hoc.

Você deve usar o certificado de desenvolvedor emitido pela Apple para assinar seu aplicativo. Certificados diferentes são usados para a criação de versões de teste que são utilizadas para a compactação final antes do envio do aplicativo.

Para obter um exemplo de como empacotar um aplicativo iOS usando Ant, consulte Piotr Walczyszyn: Como empacotar um aplicativo do AIR para dispositivos iOS com o comando ADT e o script ANT

Compactação com ADT

As versões 2.6 e posteriores do AIR SDK suportam o empacotamento para iOS e Android. Antes de compactar todo o código MXML, ActionScript e qualquer outra extensão devem ser compilados. Você deve também ter um certificado de assinatura de código.

Para uma referência pormenorizada sobre as opções e os comandos ADT, consulte "AIR Developer Tool (ADT)" na página 168.

Pacotes APK do Android

Criação de um pacote APK

Para criar um pacote APK, use o comando package do ADT, definindo o tipo de destino para *apk* para compilações de versão, *apk-debug* para compilações de depuração, ou *apk-emulator* para compilações em modo de versão para executar em um emulador.

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

Digite o comando inteiro em uma única linha; quebras de linha no exemplo acima estão presentes apenas para torná-lo mais fácil de ler. Além disso, o exemplo pressupõe que o caminho para a ferramenta ADT está na definição do caminho do shell de linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308 para obter ajuda.)

Você deve executar o comando do diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, e um diretório de ícones.

Quando você executa o comando, como demonstrado, a ADT solicitará a senha do armazenamento de chaves. (Os caracteres da senha digitada não são exibidos; basta pressionar Enter quando terminar de digitar.)

Nota: *Por padrão, todos os aplicativos AIR Android têm o* `air.` *prefixo no nome do pacote. Para cancelar este comportamento padrão, defina a variável de ambiente,* `AIR_NOANDROIDFLAIR`*, como true no computador.*

Criação de um pacote APK para um aplicativo que utilize extensões nativas.

Para criar um pacote APK para um aplicativo que use extensões nativas, use a opção `-extdir` além das opções normais de empacotamento. Caso haja vários ANEs que compartilham recursos/bibliotecas, o ADT escolhe apenas um único recurso/biblioteca e ignora outras entradas duplicadas antes de emitir um aviso. Esse opção especifica o diretório que contém os arquivos ANE que o aplicativo utiliza. Por exemplo:

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

Criação de um pacote APK que inclua sua própria versão do runtime do AIR

Para criar um pacote APK que contenha tanto o aplicativo quanto a versão cativa do runtime do AIR, use o destino `apk-captive-runtime`. Esse opção especifica o diretório que contém os arquivos ANE que o aplicativo utiliza. Por exemplo:

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

As possíveis desvantagens dessa técnica incluem:

- Correções importantes de segurança não estão automaticamente disponíveis para os usuários quando a Adobe publica um patch de segurança
- Tamanho de memória RAM maior para os aplicativos

***Nota:** Ao agrupar o runtime, o ADT adiciona as permissões `INTERNET` e `BROADCAST_STICKY` ao seu aplicativo. Essas permissões são requeridas pelo runtime do AIR.*

Criação de um pacote de depuração APK

Para criar uma versão do aplicativo que você pode usar com um depurador, use apk-debug como o destino e especifique opções de conexão:

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

O sinalizador -connect diz ao runtime do AIR no dispositivo onde se conectar a um depurador remoto através da rede. Para depurar através de USB, você deve especificar o sinalizador `-listen`, especificando a porta TCP para ser usada para a conexão de depuração:

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

Para a maioria dos recursos de depuração funcionar, você também deve compilar os SWFs e SWCs do aplicativo com a depuração ativada. Consulte "Opções de conexão do depurador" na página 186 para obter uma descrição completa dos sinalizadores `-connect` e `-listen`.

***Nota:** Por padrão, o ADT cria um pacote de uma cópia cativa do tempo de execução do AIR com seu aplicativo Android e, ao mesmo tempo, cria um pacote do aplicativo com o destino apk-debug. Para forçar o ADT a criar um APK que use o tempo de execução externo, defina a variável de ambiente `AIR_ANDROID_SHARED_RUNTIME` como `true`.*

No Android, o aplicativo também deve ter permissão para acessar a Internet para que ele se conecte ao computador que está executando o depurador na rede. Consulte "Permissões do Android" na página 77.

Criação de um pacote APK para uso em um emulador do Android

Você pode usar um pacote de depuração APK em um emulador do Android, mas não um pacote de modo de versão. Para criar um pacote de modo de versão APK para uso em um emulador, use o comando package do ADT definindo o tipo de destino para *apk-emulator* :

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

O exemplo pressupõe que o caminho para a ferramenta ADT está na definição do caminho do shell de linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308 para obter ajuda.)

Criação de um pacote APK a partir de um arquivo AIR ou AIRI

Você pode criar um pacote APK diretamente de um arquivo AIR ou AIRI existente:

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

O arquivo do AIR deve usar o namespace AIR 2.5 (ou superior) no arquivo descritor do aplicativo.

Criar um pacote APK para a plataforma Android x86

Começando pelo AIR 14, o argumento `-arch` pode ser usado para criar um pacote de APK para a plataforma Android x86. Por exemplo:

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

Pacotes iOS

No iOS, o ADT converte o código de bites do arquivo SWF e outros arquivos de origem em um aplicativo iOS nativo.

1. Crie o arquivo SWF usando o Flash Builder, o Flash Professional ou um compilador de linha de comando.
2. Abra o comando shell ou um terminal e navegue para a pasta de projetos do aplicativo para iPhone.
3. Em seguida, utilize a ferramenta ADT para criar um arquivo IPA utilizando a sintaxe a seguir:

```
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
```

   Altere a referência `adt` para incluir o caminho completo para o aplicativo adt. O aplicativo adt está instalado no subdiretório bin do AIR SDK.

Selecione a opção `-target` correspondente ao tipo de aplicativo para iPhone que deseja criar:

- `-target ipa-test`: escolha essa opção para compilar rapidamente a versão do aplicativo para testar em seu iPhone do desenvolvedor. Você também pode usar o `ipa-test-interpreter` para obter uma compilação ainda mais rápida ou `ipa-test-interpreter-simulator` para executar o simulador do iOS.
- `-target ipa-debug`: Escolha essa opção para compilar a versão de depuração para testar em seu iPhone de desenvolvimento. Com essa opção, você pode utilizar uma sessão de depuração para receber a saída de `trace()` do aplicativo iPhone.

  É possível incluir uma das seguintes opções `-connect` (`CONNECT_OPTIONS`) para especificar o endereço IP do computador de desenvolvimento que está executando o depurador:

  - `-connect`— o aplicativo tentará se conectar a uma sessão de depuração por wifi no computador de desenvolvimento utilizado para compilar o aplicativo.
  - `-connect IP_ADDRESS`— o aplicativo tentará se conectar a uma sessão de depuração por wifi no computador com o endereço IP especificado. Por exemplo:

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

  - `-connect HOST_NAME`— o aplicativo tentará se conectar a uma sessão de depuração por wifi no computador com o nome de host especificado. Por exemplo:

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

  A opção `-connect` é facultativa. Caso não especificado, o aplicativo de depuração resultante não tentará se conectar a um depurador hospedado. Como alternativa, você pode especificar `-listen` em vez de `-connect` para ativar a depuração USB descrita em "Depuração remota com FDB através de USB" na página 107.

  Se houver falha de uma tentativa de conexão de depuração, o aplicativo apresentará uma caixa de diálogo que solicitará que o usuário insira o endereço IP da máquina de host da depuração. Uma tentativa de conexão pode falhar se o dispositivo não estiver conectado ao wifi. Isto também pode ocorrer se o dispositivo estiver conectado mas não estiver protegido por um firewall da máquina de host de depuração.

  Você também pode usar o `ipa-debug-interpreter` para obter uma compilação mais rápida ou `ipa-debug-interpreter-simulator` para executar o simulador iOS.

  Para obter mais informações, consulte "Depuração de um aplicativo do AIR móvel" na página 101.
- `-target ipa-ad-hoc`: Escolha essa opção para criar um aplicativo para implantação ad hoc. Consulte o centro de desenvolvedores da Apple iPhone
- `-target ipa-app-store`: Escolha essa opção para criar a versão final do arquivo IPA para implantação na Apple App Store.

Substitua *`PROFILE_PATH`* pelo caminho para o arquivo de perfil de aprovisionamento de seu aplicativo. Para obter mais informações sobre perfis de aprovisionamento, consulte "Configuração do iOS" na página 66.

Use a opção `-platformsdk` para apontar para o SDK do simulador iOS quando estiver criando para executar seu aplicativo no simulador iOS.

Substitua *`SIGNING_OPTIONS`* para fazer referência ao certificado e senha de desenvolvedor de iPhone. Use a seguinte sintaxe:

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

Substitua *P12_FILE_PATH* pelo caminho para o arquivo de certificado P12. Substitua *PASSWORD* pela senha do certificado. (Veja o exemplo abaixo). Para obter mais informações sobre o arquivo de certificado P12, consulte "Converter um certificado de desenvolvedor em um arquivo de armazenamento de chave P12" na página 201.

Nota: *Você pode usar um certificado autoassinado ao empacotar para o simulador iOS.*

Substitua o *APP_DESCRIPTOR* para fazer referência ao arquivo do descritor do aplicativo.

Substitua *SOURCE_FILES* para fazer referência ao arquivo principal SWF de seu projeto seguido de outros ativos para inclusão. Inclua os caminhos para todos os arquivos de ícones definidos na caixa de diálogo de configurações do aplicativo no Flash Professional ou em um arquivo personalizado do arquivo do descritor do aplicativo. Além disso, inclua o arquivo Default.png da arte de tela inicial.

Use a opção `-extdir` *`extension-directory`* para especificar o diretório que contém os arquivos ANE (extensões nativas) que o aplicativo utiliza. Caso o aplicativo não utilizar extensões nativas, não inclua essa opção.

Importante: *Não crie um subdiretório chamado* `Recursos` *no diretório do seu aplicativo. O runtime cria automaticamente uma pasta com esse nome para se adequar à estrutura do pacote de IPA. A criação de uma pasta Recursos resulta em um conflito fatal.*

Criação de um pacote do iOS para depuração

Para criar um pacote do iOS para instalar em dispositivos de teste, use o comando package do ADT, definindo o tipo de origem para *ios-debug*. Antes de executar este comando, você já deve ter obtido um certificado de assinatura de código de desenvolvimento e perfil de aprovisionamento da 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: *Você também pode usar o* `ipa-debug-interpreter` *para obter uma compilação mais rápida ou* `ipa-debug-interpreter-simulator` *para executar o simulador iOS*

Digite o comando inteiro em uma única linha; quebras de linha no exemplo acima estão presentes apenas para torná-lo mais fácil de ler. Além disso, o exemplo pressupõe que o caminho para a ferramenta ADT está na definição do caminho do shell de linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308 para obter ajuda.)

Você deve executar o comando do diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, um diretório de ícones e o arquivo Default.png.

Você deve assinar o aplicativo usando o certificado de distribuição correto to emitido pela Apple; outros certificados de assinatura de código não podem ser usados.

Use a opção `-connect` para depuração wifi. O aplicativo tenta iniciar uma sessão de depuração com o Flash Debugger (FDB) em execução no IP ou nome de host especificado. Use a opção `-listen` para depuração USB. Você inicia o aplicativo primeiro e então o FDB que inicia uma sessão de depuração para o aplicativo em execução. Consulte "Conexão ao depurador do Flash" na página 105 para obter mais informações.

Criação de um pacote do iOS para envio à App Store da Apple

Para criar um pacote do iOS para a envio à App Store da Apple, use o comando package do ADT, definindo o tipo de destino para *ios-app-store* . Antes de executar este comando, você já deve ter obtido um certificado de assinatura de código de distribuição e perfil de aprovisionamento da 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
```

Digite o comando inteiro em uma única linha; quebras de linha no exemplo acima estão presentes apenas para torná-lo mais fácil de ler. Além disso, o exemplo pressupõe que o caminho para a ferramenta ADT está na definição do caminho do shell de linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308 para obter ajuda.)

Você deve executar o comando do diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, um diretório de ícones e o arquivo Default.png.

Você deve assinar o aplicativo usando o certificado de distribuição correto to emitido pela Apple; outros certificados de assinatura de código não podem ser usados.

Importante: *Apple exige que você use o programa Application Loader para carregar aplicativos para a App Store. A Apple somente publica o Application Loader para Mac OS X. Assim, enquanto você pode desenvolver um aplicativo do AIR para o iPhone usando um computador com Windows, você deve ter acesso a um computador com Mac OS X (versão 10.5.3 ou posterior) para enviar o aplicativo para a App Store. Você pode obter o programa "Application Loader" do Centro do desenvolvedor do iOS da Apple.*

Criação de um pacote do iOS para distribuição ad hoc

Para criar um pacote do iOS para distribuição ad hoc, use o comando package do ADT, definindo o tipo de origem para *ios-ad-hoc*. Antes de executar este comando, você já deve ter obtido o certificado de assinatura de código de distribuição ad hoc adequado e perfil de aprovisionamento da 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
```

Digite o comando inteiro em uma única linha; quebras de linha no exemplo acima estão presentes apenas para torná-lo mais fácil de ler. Além disso, o exemplo pressupõe que o caminho para a ferramenta ADT está na definição do caminho do shell de linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308 para obter ajuda.)

Você deve executar o comando do diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, um diretório de ícones e o arquivo Default.png.

Você deve assinar o aplicativo usando o certificado de distribuição correto to emitido pela Apple; outros certificados de assinatura de código não podem ser usados.

Criação de um pacote do iOS para um aplicativo que utilize extensões nativas.

Para criar um pacote do iOS para um aplicativo que utilize extensões nativas, use o comando do pacote ADT com a opção `-extdir` . Use o comando ADT como apropriado para o destino (`ipa-app-store`, `ipa-debug`, `ipa-ad-hoc`, `ipa-test`). Por exemplo:

```
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
```

Digite o comando inteiro em uma única linha; quebras de linha no exemplo acima estão presentes apenas para torná-lo mais fácil de ler.

Em relação às extensões nativas, o exemplo supõe que o diretório denominado `extensionsDir` está no diretório no qual o comando é executado. O diretório `extensionsDir` contém os arquivos ANE que o aplicativo utiliza.

Depuração de um aplicativo do AIR móvel

Você pode depurar seu aplicativo do AIR móvel de várias maneiras. A maneira mais simples de descobrir problemas de lógica de aplicativo é a depuração no computador de desenvolvimento utilizando ADL ou o simulador iOS. Você também pode instalar o aplicativo em um dispositivo e depurar remotamente com o depurador do Flash em um computador desktop.

Simulação de dispositivos utilizando ADL

A forma mais rápida e fácil de testar e depurar a maioria dos recursos de aplicativo móvel é executar o aplicativo no computador de desenvolvimento usando o utilitário ADL (Adobe Debug Launcher). A ADL usa o elemento `supportedProfiles` no descritor do aplicativo para determinar qual perfil será usado. Se mais de um perfil estiver listado, o ADL usa o primeiro na lista. Você também pode usar o parâmetro `-profile` do ADL para selecionar um dos outros perfis na lista supportedProfiles. (Se você não incluir um elemento supportedProfiles no descritor do aplicativo, qualquer perfil poderá ser especificado para o `argumento` -profile.) Por exemplo, use o seguinte comando para iniciar um aplicativo para simular o perfil de dispositivo móvel:

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

Ao simular o perfil móvel no desktop desta forma, o aplicativo é executado em um ambiente que mais se aproxima de um dispositivo móvel de destino. As APIs do ActionScript que não fazem parte do perfil móvel não estão disponíveis. No entanto, o ADL não faz distinção entre os recursos de diferentes dispositivos móveis. Por exemplo, você pode enviar pressões simuladas de teclas de função para seu aplicativo, apesar de seu dispositivo destino na verdade não utilizar teclas de função.

ADL é compatível com simulações de mudanças de orientação de dispositivo e entrada de teclas de função através de comandos de menu. Ao executar o ADL no perfil de dispositivo móvel, ele exibe um menu (em qualquer janela do aplicativo ou na barra de menu do desktop) que permite que você insira a rotação do dispositivo ou a entrada da tecla de função.

Entrada de tecla de função

O ADL simula os botões de tecla de função para botões de Voltar, Menu e Pesquisar em um dispositivo móvel. Você pode enviar estas teclas para o dispositivo simulado usando o menu exibido quando o ADL é ativado através do perfil móvel.

Rotação do dispositivo

O ADL permite simular a rotação do dispositivo através do menu exibido quando o ADL é ativado usando o perfil móvel. Você pode girar o dispositivo simulado para a esquerda ou para a direita.

A simulação de rotação afeta apenas um aplicativo que permita a orientação automática. Você pode ativar este recurso definindo o elemento `autoOrients` para `true` no descritor do aplicativo.

Tamanho da tela

Você pode testar seu aplicativo em telas de tamanho diferente, definindo o parâmetro `-screensize` do ADL. Você pode passar o código para um dos tipos de tela predefinidos ou uma sequência de caracteres que contenha os quatro valores que representam as dimensões em pixel das telas normal e maximizada.

Sempre especifique as dimensões de pixel para o layout retrato, o que significa especificar a largura como um valor menor do que o valor da altura. Por exemplo, o comando a seguir abre o ADL para simular a tela usada no Droid da Motorola:

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Para uma lista de tipos de tela predefinidos, consulte "Uso do ADL" na página 162.

Limitações

Algumas APIs que não são compatíveis no perfil desktop não podem ser simuladas pelo ADL. As APIs que não são simuladas incluem:

- Acelerômetro
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocalização
- Multitoque e gestos em sistemas operacionais de desktop que não são compatíveis com estes recursos
- SystemIdleMode

Se seu aplicativo usa essas classes, você deve testar os recursos em um dispositivo de verdade ou em um emulador.

De forma semelhante, existem APIs que funcionam ao serem executadas sob a ADL no computador pessoal, mas que não funcionam em todos os tipos de dispositivos móveis. Isso inclui:

- Codec de áudio Speex e AAC
- Suporte para leitor de tela e acessibilidade
- RTMPE
- Carregando arquivos SWF que contêm bytecode ActionScript
- Sombreadores PixelBender

Certifique-se de testar os aplicativos que usam recursos nos dispositivos de destino, uma vez que a ADL não replica completamente o ambiente de execução.

Simulação de dispositivo usando o simulador iOS

O simulador iOS (somente Mac) oferece uma forma mais rápida de executar e depurar aplicativos iOS. Ao testar com o simulador iOS, você não precisa de um certificado de desenvolvedor ou de um perfil de provisionamento. Você ainda precisa criar um certificado p12 ainda que ele possa ser autoassinado.

Por padrão, o ADT sempre inicia o simulador do iPhone. Para alterar o dispositivo do simulador, realize o procedimento a seguir:

- Use o comando abaixo para ver os simuladores disponíveis.

  ```
  xcrun simctl list devices
  ```

  A saída exibida é similar à saída exibida abaixo.

  ```
  == 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)
  ....
  ```

- Configure a variável de ambiente `AIR_IOS_SIMULATOR_DEVICE` para escolher um simulador específico:

  ```
  export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
  ```

Reinicie o processo após definir a variável de ambiente e execute o aplicativo no dispositivo simulador de sua escolha.

***Nota:** Ao usar o ADT com o simulador iOS, você precisa sempre incluir a opção `-platformsdk` especificando o caminho do SDK do simulador iOS.*

Para usar um aplicativo com o simulador iOS:

1. Use o comando adt -package com `-target ipa-test-interpreter-simulator` ou `-target ipa-debug-interpreter-simulator`, como mostrado no exemplo a seguir:

   ```
   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:** Agora, as opções de assinatura não são mais necessárias no caso de simuladores. Por isso, qualquer valor pode ser fornecido no sinalizador `-keystore`, já que não será considerado pelo ADT.*

2. Use o comando adt -installApp para instalar o aplicativo no simulador do iOS, conforme mostrado no exemplo a seguir:

   ```
   adt -installApp
                                  -platform ios
                                  -platformsdk
   /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                  -device ios-simulator
                                  -package sample_ipa_name.ipa
   ```

3. Use o comando adt -launchApp para instalar o aplicativo no simulador do iOS, conforme mostrado no exemplo a seguir:

Nota: *Por padrão, o comando* `adt -launchApp` *executa o aplicativo no simulador do iPhone. Para executar o aplicativo no simulador do iPad, exporte a variável de ambiente,* `AIR_IOS_SIMULATOR_DEVICE` *= "iPad" e use o 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 testar uma extensão nativa no simulador iOS, use o nome de plataforma `iPhone-x86` no arquivo extension.xml e especifique `library.a` (biblioteca estática) no elemento `nativeLibrary` como o exemplo extension.xml a seguir mostra:

```
<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: *Ao testar uma extensão nativa no simulador iOS, não use a biblioteca estática (arquivo* `.a`*) compilado para o dispositivo. Em vez disso, certifique-se de usar a biblioteca estática compilada para o simulador.*

Instruções de rastreio

Ao executar o aplicativo de móvel no desktop, a saída de rastreio é impressa para o depurador ou para a janela de terminal usada para ativar o ADL. Ao executar seu aplicativo em um dispositivo ou emulador, você pode configurar uma sessão de depuração remota para ver a saída de rastreio. Quando possível, você também pode ver a saída de rastreio usando as ferramentas de desenvolvimento de software fornecidas pelo fabricante do dispositivo ou do sistema operacional.

Em todos os casos, os arquivos SWF do aplicativo devem ser compilado com a depuração ativada para o runtime para a saída de todas as instruções de rastreio.

Instruções de rastreio remoto no Android

Quando executado em um dispositivo ou emulador do Android, é possível exibir a saída de instrução de rastreio no log do sistema do Android usando o utilitário Android Debug Bridge (ADB) incluído no Android SDK. Para visualizar a saída de sua aplicação, execute o seguinte comando a partir de uma janela de terminal ou do prompt de comando no computador de desenvolvimento:

```
tools/adb logcat air.MyApp:I *:S
```

onde *MyApp* é o ID do aplicativo do AIR de seu aplicativo. O argumento `*:S` omite a saída de todos os outros processos. Para exibir informações do sistema sobre seu aplicativo além da saída de rastreio, você pode incluir o ActivityManager na especificação do filtro logcat:

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

Estes exemplos de comando presumem que você está executando o ADB da pasta Android SDK, ou que você adicionou a pasta `SDK` à variável do ambiente de caminho.

Nota: *Do AIR 2.6 em diante, o utilitário ADB está incluído no AIR SDK e pode ser encontrado na pasta lib/android/bin.*

Instruções de rastreio remoto no iOS
Para exibir a saída de instruções de rastreio de um aplicativo em execução em um dispositivo do iOS, você deve estabelecer uma sessão de depuração remota usando o depurador do Flash (FDB).

Mais tópicos da Ajuda

Android Debug Bridge: ativar registro de logcat

"Variáveis de ambiente do caminho" na página 308

Conexão ao depurador do Flash

Para depurar um aplicativo rodando em um dispositivo móvel, você pode executar o depurador do Flash no seu computador de desenvolvimento e se conectar a ele através da rede. Para ativar a depuração remota, você deve fazer o seguinte:

- No Android, especifique a permissão android:permission.INTERNET no descritor do aplicativo.
- Compile os SWFs do aplicativo com a depuração ativada.
- Empacote o aplicativo com o sinalizador `-target apk-debug, para Android,` ou `-target ipa-debug`, para iOS e o `-connect` (depuração wifi) ou `-listen` (depuração USB).

Para depuração remota por wifi, o dispositivo deve conseguir acessar a porta TCP 7935 do computador que executa o depurador do Flash por endereço IP ou pelo nome de domínio totalmente qualificado. Para depuração remota por USB, o dispositivo precisa poder acessar a porta TCP 7936 ou a porta especificada no sinalizador `-listen`.

No iOS, você também pode especificar `-target ipa-debug-interpreter` ou `-target ipa-debug-interpreter-simulator`.

Depuração remota com o Flash Professional

Uma vez que seu aplicativo esteja pronto para depurar e as permissões sejam definidas no descritor do aplicativo, faça o seguinte:

1. Abra a caixa de diálogo Configurações do AIR for Android.
2. Na aba Implementação:
   - Selecione "Depuração do dispositivo" para o tipo de implementação
   - Selecione "Instalar aplicativo no dispositivo Android conectado" para Após publicação
   - Desmarque a seleção "Ativar aplicativo no dispositivo Android conectado" para Após publicação
   - Defina o caminho para o Android SDK, se necessário.
3. Clique em Publicar.

   Seu aplicativo é instalado e ativado no dispositivo.
4. Feche a caixa de diálogo Configurações do AIR for Android.
5. Selecione Depurar > Começar sessão de depuração remota > ActionScript 3 do menu do Flash Professional.

   O Flash Professional exibe "Aguardando conexão do Player" no painel de saída.

6 Inicie o aplicativo no dispositivo.

7 Digite o endereço IP ou nome de host do computador que está executando o depurador do Flash na janela de conexão do Adobe AIR e clique em OK.

Depuração remota com FDB através de uma conexão de rede

Para depurar um aplicativo em execução em um dispositivo com a linha de comando do depurador do Flash (FDB), execute o depurador no computador de desenvolvimento e inicie o aplicativo no dispositivo. O procedimento a seguir usa as ferramentas AMXMLC, FDB e ADT para compilar, compactar e depurar um aplicativo no dispositivo. Os exemplos assumem que você está usando Flex e AIR SDK combinados e que o diretório bin está incluído na variável de ambiente do caminho. (Esta suposição é feita apenas para simplificar os exemplos de comando.)

1 Abra um terminal ou janela de comando do prompt e vá ao diretório que contém o código-fonte para o aplicativo.

2 Compile o aplicativo com amxmlc, ativando a depuração:

```
amxmlc -debug DebugExample.as
```

3 Compacte o aplicativo usando os destinos `apk-debug` ou `ipa-debug`:

```
Android
                                    adt -package -target apk-debug -connect -storetype
pkcs12 -keystore ../../AndroidCert.p12 DebugExample.apk DebugExample-app.xml
DebugExample.swf
                                    iOS
                                    adt -package -target ipa-debug -connect -storetype
pkcs12 -keystore ../../AppleDeveloperCert.p12 -provisioning-profile test.mobileprovision
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

Se você sempre usa o mesmo nome de host ou endereço IP para a depuração, pode colocar esse valor depois do sinalizador `-connect`. O aplicativo tentará se conectar com esse endereço IP ou com o nome de host automaticamente. Caso contrário, você deve digitar as informações no dispositivo cada vez que iniciar a depuração.

4 Instalar o aplicativo.

No Android, você pode usar o comando `-installApp` do ADT:

```
adt -installApp -platform android -package DebugExample.apk
```

No iOS, você pode instalar o aplicativo usando o comando `-installApp` do ADT ou o iTunes.

5 Abra um segundo terminal ou janela de comandos e execute FDB:

```
fdb
```

6 Na janela FDB, digite o 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 o aplicativo no dispositivo.

8 Assim que o aplicativo for ativado no dispositivo ou no emulador, o Adobe AIR abre a caixa de diálogo de conexão. (Se você especificou um nome de host ou endereço IP com a opção -connect ao compactar o aplicativo, ele tentará se conectar automaticamente usando esse endereço.) Digite o endereço correto e toque em OK.

Para se conectar com o depurador neste modo, o dispositivo deve ser capaz de resolver o endereço ou nome de host e conectar a porta TCP 7935. É necessária uma conexão de rede.

9 Quando o runtime remota se conecta com o depurador, você pode definir pontos de interrupção com o comando `break` do FDB e iniciar a execução com o 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
```

Depuração remota com FDB através de USB

AIR 2.6 (Android) AIR 3.3 (iOS)

Para depurar um aplicativo por uma conexão USB, você empacota o aplicativo usando a opção `-listen` em vez da opção `-connect`. Quando você especifica a opção `-listen`, o runtime monitora uma conexão do depurador Flash (FDB) na porta TCP 7936 quando iniciar o aplicativo. Você então executa o FDB com a opção `-p` e o FDB inicia a conexão.

Procedimento de depuração USB para Android

Para que o depurador Flash em execução no computador se conecte ao runtime do AIR em execução no dispositivo ou emulador, você precisa usar o Android Debug Bridge (ADB - utilitário do Android SDK) ou o iOS Debug Bridge (IDB - utilitário do AIR SDK) para encaminhar a porta do dispositivo para a porta do computador.

1 Abra um terminal ou janela de comando do prompt e vá ao diretório que contém o código-fonte para o aplicativo.

2 Compile o aplicativo com amxmlc, ativando a depuração:

```
amxmlc -debug DebugExample.as
```

3 Empacote o aplicativo usando o destino de deputação apropriado (como `apk-debug`) e especifique a opção `-listen`:

```
adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

4 Conecte o dispositivo ao computador de depuração com um cabo USB. (Você também pode usar esse procedimento para depurar um aplicativo em execução em um emulador; neste caso a conexão USB não é necessária - ou possível.)

5 Instalar o aplicativo.

Você também pode usar o comando ADT `-installApp`.

```
adt -installApp -platform android -package DebugExample.apk
```

6 Encaminhe a porta TCP 7936 do dispositivo ou do emulador para o computador desktop usando o utilitário ADB do Android:

```
adb forward tcp:7936 tcp:7936
```

7 Inicie o aplicativo no dispositivo.

8 Em um terminal ou janela de comando execute o FDB usando a opção -p:

```
fdb -p 7936
```

9 Na janela FDB, digite o comando `run`:

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 O utilitário FDB tenta se conectar com o aplicativo.

11 Quando a conexão remota é estabelecida, você pode definir pontos de interrupção com o comando `break` do FDB e iniciar a execução com o 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:** O número de porta 7936 é usado como padrão para depuração por USB pelo runtime do AIR e pelo FDB. Você pode especificar portas diferentes para usar com o parâmetro de porta -listen do ADT o -p do FDB. Neste caso, você deve usar o utilitário Android Debug Bridge para encaminhar o número da porta especificada em ADT para a porta especificada em FDB: `adb forward tcp:adt_listen_port#:tcp:fdb_port#`*

Procedimento de depuração para USB para iOS

Para que o depurador do Flash em execução no computador se conecte ao runtime do AIR em execução no dispositivo ou no emulador, você deve usar o utilitário iOS Debug Bridge (IDB - utilitário do AIR SDK) para encaminhar a porta do dispositivo para a porta do desktop.

1 Abra um terminal ou janela de comando do prompt e vá ao diretório que contém o código-fonte para o aplicativo.

2 Compile o aplicativo com amxmlc, ativando a depuração:

```
amxmlc -debug DebugExample.as
```

3 Empacote o aplicativo usando o destino de depuração apropriado (como `ipa-debug` ou `ipa-debug-interpreter` e especificar a opção `-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 o dispositivo ao computador de depuração com um cabo USB. (Você também pode usar esse procedimento para depurar um aplicativo em execução em um emulador; neste caso a conexão USB não é necessária - ou possível.)

5 Instale e inicie o aplicativo no dispositivo iOS. No AIR 3.4 e superior você pode usar`adt -installApp` para instalar o aplicativo por USB.

6 Determine o identificador do dispositivo usando o comando `idb -devices` (IDB está localizado em *`air_sdk_root`*`/lib/aot/bin/iOSBin/idb`):

```
./idb -devices
                                    List of attached devices
                                    Handle    UUID
                                        1     91770d8381d12644df91fbcee1c5bbdacb735500
```

***Nota:** (AIR 3.4 e superior) Você pode usar `adt -devices` no lugar de `idb -devices` para determinar o manipulador do dispositivo.*

7 Encaminhe uma porta em seu desktop para a porta especificada no parâmetro `adt -listen` (neste caso, 16000; o padrão é 7936) usando o utilitário IDB e a ID de dispositivo encontrada na etapa anterior:

```
idb -forward 7936 16000 1
```

Neste exemplo, 7936 é a porta do desktop, 16000 é a porta pela qual o dispositivo conectado ouve e 1 é a ID do dispositivo do dispositivo conectado.

8 Em um terminal ou janela de comando execute o FDB usando a opção -p:

```
fdb -p 7936
```

9 Na janela FDB, digite o comando `run`:

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 O utilitário FDB tenta se conectar com o aplicativo.

11 Quando a conexão remota é estabelecida, você pode definir pontos de interrupção com o comando `break` do FDB e iniciar a execução com o comando `continue`:

***Nota:** O número de porta 7936 é usado como padrão para depuração por USB pelo runtime do AIR e pelo FDB. Você pode especificar portas diferentes para usar com o parâmetro de porta -listen do IDB o -p do FDB.*

Instalação de aplicativos AIR AIRI nos dispositivos móveis

Os usuários finais do seu aplicativo podem instalar o runtime do AIR e os aplicativos do AIR usando o aplicativo normal e o mecanismo de distribuição para seus dispositivos.

No Android, por exemplo, os usuários podem instalar aplicativos no Android Market. Ou se tiverem permissão para a instalação de aplicativos a partir de fontes desconhecidas, nas configurações do aplicativo, os usuários podem instalar um aplicativo clicando em um link em uma página da web, ou copiando o pacote de aplicativos para seu dispositivo e abrindo-o. Se um usuário tenta instalar um aplicativo do Android, mas não tem o AIR instalado ainda, será automaticamente direcionado para o Market, onde é possível instalar o runtime.

No iOS, há duas maneiras de distribuir aplicativos para usuários finais. O principal canal de distribuição é a App Store da Apple. Você também pode usar a distribuição ad hoc para permitir que um número limitado de usuários instalem o aplicativo sem ir até a App Store.

Instale o runtime do AIR e os aplicativos para desenvolvimento

Como os aplicativos do AIR em dispositivos móveis são instalados como pacotes nativo, você pode utilizar os recursos da plataforma normal de instalação de aplicativos para testes. Quando disponível, você pode usar comandos do ADT para instalar o runtime e os aplicativos do AIR. No momento esta abordagem é compatível com o Android.

No iOS, você pode instalar aplicativos para testes usando o iTunes. Os aplicativos de testes devem ser assinados com um certificado de assinatura por código da Apple, emitido especificamente para o desenvolvimento de aplicativos e compactados com um perfil de desenvolvimento de aprovisionamento. Um aplicativo do AIR é um pacote independente no iOS. Um runtime separado não é usado.

Instalação de aplicativos do AIR usando o ADT

Durante o desenvolvimento de aplicativos do AIR você pode usar o ADT para instalar e desinstalar o runtime e seus aplicativos. (O IDE também pode integrar estes comandos para que você não tenha que executar o ADT por si próprio).

Você pode instalar o runtime do AIR em um dispositivo ou emulador usando o utilitário AIR ADT. O SDK fornecido para o dispositivo deve ser instalado. Use o comando `-installRuntime`:

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

Se o parâmetro `-package` não for especificado, o pacote do runtime adequado para o dispositivo ou emulador é escolhido entre os disponíveis no seu AIR SDK instalado.

Para instalar um aplicativo AIR no Android ou no iOS (AIR 3.4 e superior), use o comando similar `-installApp`:

```
adt -installApp -platform android -device deviceID -package path-to-app
```

O valor definido para o argumento `-platform` deve corresponder ao dispositivo no qual você está instalando.

Nota: *Versões existentes do runtime ou do aplicativo do AIR devem ser removidas antes de serem reinstaladas.*

Instalando aplicativos AIR em dispositivos iOS com o iTunes

Para instalar um aplicativo do AIR num dispositivo iOS para fim de teste:

1. Abra o aplicativo iTunes.
2. Se você ainda não fez isso, adicione o perfil de aprovisionamento desse aplicativo ao iTunes. No iTunes, selecione Arquivo > Adicionar à Biblioteca. Selecione o arquivo de perfil de aprovisionamento (que tem mobileprovision como tipo de arquivo).
3. Algumas versões do iTunes não substituem o aplicativo, se a mesma versão do aplicativo estiver instalada. Nesse caso, exclua o aplicativo do seu dispositivo e da lista de aplicativos no iTunes.
4. Clique duas vezes no arquivo IPA do aplicativo. O arquivo deve aparecer na lista de aplicativos no iTunes.
5. Conecte seu dispositivo à porta USB em seu computador.
6. No iTunes, selecione a guia Aplicativos do dispositivo e certifique-se de que o aplicativo está selecionado na lista de aplicativos para instalação.
7. Selecione o dispositivo na lista à esquerda do aplicativo iTunes. Em seguida, clique no botão Sync. Quando a sincronização terminar, o aplicativo Hello World aparecerá em seu iPhone.

Se a nova versão não estiver instalada, exclua-a do dispositivo e da lista de aplicativos no iTunes e, em seguida, execute novamente esse procedimento. Talvez esse seja o caso, se a versão instalada atualmente usa o mesmo ID do aplicativo e versão.

Mais tópicos da Ajuda

"Comando installRuntime do ADT" na página 180

"Comando installApp do ADT" na página 177

Execução de aplicativos do AIR em um dispositivo

Você pode iniciar aplicativos do AIR instalados usando a interface de usuário do dispositivo. Quando disponível, você também pode iniciar aplicativos remotamente usando o utilitário AIR ADT:

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

O valor do argumento `-appid` deve ser o ID do aplicativo do AIR para ativação. Use o valor especificado no descritor do aplicativo do AIR (sem o prefixo *air.*incluído durante a compactação).

Se apenas um único dispositivo ou emulador estiver anexado e em execução, você pode omitir o sinalizador `-device`. O valor definido para o argumento `-platform` deve corresponder ao dispositivo no qual você está instalando. No momento apenas o valor *android* está disponível.

Remoção do runtime e aplicativos do AIR

Você pode usar os meios normais de remoção de aplicativos fornecidos pelo sistema operacional do dispositivo. Quando disponível, você também pode usar o utilitário AIR ADT para remover os aplicativos e o runtime do AIR. Para remover o runtime use o comando `-uninstallRuntime`:

```
adt -uninstallRuntime -platform android -device deviceID
```

Para desinstalar um aplicativo use o comando `-uninstallApp`:

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Se apenas um único dispositivo ou emulador estiver anexado e em execução, você pode omitir o sinalizador `-device`. O valor definido para o argumento `-platform` deve corresponder ao dispositivo no qual você está instalando. No momento apenas o valor *android* está disponível.

Configuração de um emulador

Para executar o aplicativo do AIR em um emulador de dispositivo, você deve geralmente usar o SDK para o dispositivo, para criar e executar uma ocorrência do emulador em seu computador de desenvolvimento. Você pode instalar a versão do emulador do aplicativo e do runtime do AIR no emulador. Observe que os aplicativos em um emulador funcionam geralmente muito mais lentos do que em um dispositivo real.

Crie um emulador do Android

1. Ative o Android SDK e o aplicativo AVD Manager:
   - No Windows, execute o arquivo Setup.exe no SDK, na raiz do diretório do Android SDK.
   - No Mac OS, execute o aplicativo android, no subdiretório de ferramenta do diretório Android SDK
2. Selecione a opção Configurações e depois "Forçar https://".
3. Selecione a opção Pacotes disponíveis. Você verá uma lista de Android SDKs disponíveis.
4. Selecione um SDK do Android compatível (Android 2.3 ou posterior) e clique no botão Instalar selecionado.
5. Selecione a opção Dispositivos virtuais e clique no botão Novo.
6. Faça as seguintes configurações:
   - Um nome para seu dispositivo virtual
   - A API de destino, como Android 2.3, nível 8 de API
   - Um tamanho para o cartão SD (como 1024)
   - Uma capa (como HVGA Padrão)
7. Clique no botão Criar AVD.

Observe que a criação de dispositivo virtual pode levar algum tempo dependendo da configuração do sistema.

Agora você pode iniciar o novo dispositivo virtual.

1. Selecione o dispositivo virtual no aplicativo AVD Manager. O dispositivo virtual criado acima deve ser listado.

2 Selecione o dispositivo virtual e clique no botão Iniciar.

3 Clique no botão Iniciar na próxima tela.

Você deverá ver a janela de um emulador aberta em seu desktop. Isto pode demorar alguns segundos. Também pode levar algum tempo para o sistema operacional Android inicializar. Você pode instalar aplicativos compactados com *apk-debug* e *apk-emulator* em um emulador. Aplicativos compactados com o destino *apk* não funcionam em um emulador.

Mais tópicos da Ajuda

http://developer.android.com/guide/developing/tools/othertools.html#android

http://developer.android.com/guide/developing/tools/emulator.html

Atualizando aplicativos AIR móveis

Os aplicativos AIR móveis são distribuídos como pacotes nativos e, portanto, usam os mecanismos padrão de outros aplicativos na plataforma. Geralmente, isso envolve o envio ao mesmo marketplace ou armazenamento de aplicativo usado para distribuir o aplicativo original.

Os aplicativos AIR móveis não podem usar a classe ou estrutura AIR Updater.

Atualização de aplicativos do AIR no Android

Para aplicativos distribuídos no Android Market, você pode atualizar um aplicativo colocando uma nova versão no mercado, desde que os seguintes itens sejam verdadeiros (essas políticas são impostas pelo Market, não pelo AIR):

- O pacote APK é assinado pelo mesmo certificado.
- O ID do AIR é igual.
- O valor `versionNumber` no descritor do aplicativo é maior. (Você também deve incrementar o valor `versionLabel`, se usado.)

Os usuários que baixaram o aplicativo do Android Market são notificados pelo seu software de dispositivo de que uma atualização está disponível.

Mais tópicos da Ajuda

Desenvolvedores para Android: Publicação de atualizações no Android Market

Atualizando aplicativos do AIR no iOS

Nos aplicativos AIR distribuídos por meio do armazenamento de aplicativo iTunes, você pode atualizar um aplicativo enviando a atualização para o armazenamento quando todas as condições seguintes forem verdadeiras (essas políticas são impostas pelo armazenamento de aplicativos da Apple, não pelo AIR):

- O certificado de assinatura do código e os perfis de aprovisionamento são emitidos para o mesmo ID da Apple ID
- O pacote IPA usa o mesmo ID de Bundle da Apple
- A atualização não diminui a rede de dispositivos suportados (em outras palavras, se o seu aplicativo original suportar dispositivos que executem o iOS 3, então você não poderá criar uma atualização que descarte o suporte ao iOS 3).

Importante: *Uma vez que as versões 2.6 e posteriores do AIR SDK não suportam iOS 3 e o que o AIR 2 suporta, você não pode atualizar aplicativos iOS publicados que foram desenvolvidos usando o AIR 2 com uma atualização desenvolvida com o uso do AIR 2.6+.*

Use as notificações por push

As notificações por push permitem que os provedores de notificação remota enviem notificações para os aplicativos em execução em um dispositivo móvel. O AIR 3.4 tem suporte para notificações por push em dispositivos que usam o Apple Push Notification service (APNs).

Nota: *Para ativar as notificações por push em um AIR para aplicativos Android use uma extensão nativa, como as3c2dm, desenvolvida pelo divulgador da Adobe, Piotr Walczyszyn.*

O restante desta seção descreve como ativar as notificações por push no AIR para aplicativos iOS.

Nota: *Esta discussão assume que você tenha uma ID de desenvolvedor Apple, esteja familiarizado com o fluxo de trabalho de desenvolvimento iOS e tenha implantado pelo menos um aplicativo em um dispositivo iOS.*

Visão geral de notificações por push

O Apple Push Notification service (APNs) permite que os provedores de notificação remota enviem notificações para aplicativos em execução em dispositivos iOS. O APNs tem suporte para os seguintes tipos de notificação:

- Alertas
- Distintivos
- Sons

Para obter informações completas sobre o APNs, consulte developer.apple.com.

Usar notificações por push em seu aplicativo envolve vários aspectos:

- **Aplicativo cliente** - É registrado para notificações por push, se comunica com o provedor de notificações remotas e recebe notificações por push.
- **iOS** - Gerencia a interação entre o aplicativo cliente e o APNs.
- **APNs** - Fornece uma tokenID durante o registro do cliente e envia a notificação do provedor de notificação remota para o iOS.
- **Provedor de notificação remota** - Armazena as informações de tokenID do aplicativo cliente e envia notificações para o APNs.

Fluxo de trabalho de registro

O fluxo de trabalho para registrar notificações por push em um serviço no lado do servidor é:

1. O aplicativo cliente solicita que o iOS ative as notificações por push.
2. O iOS encaminha a solicitação para o ANPs.
3. O servidor APNs devolve a tokenID para o iOS.
4. O iOS devolve a tokenID para o aplicativo cliente.
5. O aplicativo cliente, usando um mecanismo específico do aplicativo, oferece a tokenID para o servidor de notificação remota, que armazena a tokenID para usar para notificações por push.

Fluxo de trabalho de notificações

O fluxo de trabalho de notificações é o seguinte:

1. O provedor de notificação remota gera uma notificação e transmite a carga da notificação para o APNs junto com a tokenID.
2. O APNs encaminha a notificação para o iOS no dispositivo.
3. O iOS encaminha a carga da notificação para o aplicativo.

API de notificação por push

O AIR 3.4 apresenta um conjunto de APIs que tem suporte para notificações por push no iOS. Essas APIs estão no pacote `flash.notifications` e incluem as seguintes classes:

- `NotificationStyle` - Define as constantes por tipos de notificação: `ALERTA`, `MEDALHA` e `SOM`.C
- `RemoteNotifier` - Permite que você assine e deixe de assinar notificações por push.
- `RemoteNotifierSubscribeOptions` - Permite que você selecione quais tipos de notificação quer receber. Use a propriedade `notificationStyles` para definir um vetor de sequências de caracteres que registram vários tipos de notificação.

O AIR 3.4 também inclui `flash.events.RemoteNotificationEvent` que são enviados pelo `RemoteNotifier`, de acordo com o seguinte:

- Quando a assinatura de um aplicativo é concluída com sucesso e uma nova tokenID é recebida do APNs.
- Após receber uma nova notificação remota.

Além disso, o `RemoteNotifier` envia `flash.events.StatusEvent` caso encontre algum erro durante o processo de assinatura.

Gerenciar as notificações por push no aplicativo.

Para registrar seu aplicativo para notificações por push, você deve realizar uma das seguintes etapas:

- Criar um código que assine notificações por push em seu aplicativo.
- Ativar as notificações por push no XML do aplicativo.
- Criar um certificado e um perfil de provisionamento para ativar o iOS Push Services.

O exemplo de código anotado a seguir assina notificações por push e manipula eventos de notificação por 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 notfications at runtime.
                        public function unSubsButtonHandler(e:MouseEvent):void{
                        remoteNot.unsubscribe();
                        tt.text +="\n UNSUBSCRIBED";
                        }
                        public function clearButtonHandler(e:MouseEvent):void{
                        tt.text = " ";
                        }
                        // Receive notification payload data and use it in your app
                        public function notificationHandler(e:RemoteNotificationEvent):void{
                        tt.appendText("\nRemoteNotificationEvent type: " + e.type +
```

```
                        "\nbubbles: "+ e.bubbles + "\ncancelable " +e.cancelable);
                        for (var x:String in e.data) {
                        tt.text += "\n"+ x + ":  " + e.data[x];
                        }
                        }
                        // If the subscribe() request succeeds, a RemoteNotificationEvent of
                        // type TOKEN is received, from which you retrieve e.tokenId,
                        // which you use to register with the server provider (urbanairship, in
                        // this example.
                        public function tokenHandler(e:RemoteNotificationEvent):void
                        {
                        tt.appendText("\nRemoteNotificationEvent type: "+e.type +"\nBubbles:
"+ e.bubbles + "\ncancelable " +e.cancelable +"\ntokenID:\n"+ e.tokenId +"\n");
                        urlString = new
String("https://go.urbanairship.com/api/device_tokens/" +
                        e.tokenId);
                        urlreq = new URLRequest(urlString);
                        urlreq.authenticate = true;
                        urlreq.method = URLRequestMethod.PUT;
                        URLRequestDefaults.setLoginCredentialsForHost
                        ("go.urbanairship.com",
                        "1ssB2iV_RL6_UBLiYMQVfg","t-kZlzXGQ6-yU8T3iHiSyQ");
                        urlLoad.load(urlreq);
                        urlLoad.addEventListener(IOErrorEvent.IO_ERROR,iohandler);
                        urlLoad.addEventListener(Event.COMPLETE,compHandler);
                        urlLoad.addEventListener(HTTPStatusEvent.HTTP_STATUS,httpHandler);
                        }
                        private function iohandler(e:IOErrorEvent):void
                        {
                        tt.appendText("\n In IOError handler" + e.errorID +" " +e.type);
                        }
                        private function compHandler(e:Event):void{
                        tt.appendText("\n In Complete handler,"+"status: " +e.type + "\n");
                        }
                        private function httpHandler(e:HTTPStatusEvent):void{
                        tt.appendText("\n in httpstatus handler,"+ "Status: " + e.status);
                        }
                        // If the subscription request fails, StatusEvent is dispatched with
                        // error level and code.
                        public function statusHandler(e:StatusEvent):void{
                        tt.appendText("\n statusHandler");
                        tt.appendText("event Level" + e.level +"\nevent code " +
                        e.code + "\ne.currentTarget: " + e.currentTarget.toString());
                        }
                        }
                        }
```

Ativar notificações por push no arquivo XML do aplicativo.

Para utilizar notificações por push em seu aplicativo, forneça o seguinte na tag `Entitlements`, na tag `iphone`:

```
<iphone>
                                ...
                                   <Entitlements>
                                      <![CDATA[
                                         <key>aps-environment</key>
                                         <string>development</string>
                                      ]]>
                                   </Entitlements>
                                </iphone>
```

Quando estiver pronto para enviar o aplicativo para a App Store, um elemento `<string>` para desenvolvimento para produção:

```
<string>production</string>
```

Se seu aplicativo suporta sequências de caracteres localizadas, especifique os idiomas na tag `supportedLanguages` embaixo da tag `initialWindow` conforme o exemplo a seguir:

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

Criar um certificado e um perfil de provisionamento para ativar o iOS Push Services

Para ativar a comunicação entre aplicativos e o APNs, você deve empacotar o aplicativo com um certificado e um perfil de provisionamento que ativem os iOS Push Services, conforme a seguir:

1. Faça o login em sua conta de desenvolvedor Apple.
2. Navegue até o Provisioning Portal (Portal de provisionamento).
3. Clique na aba de App IDs (IDs do aplicativo).
4. Clique no botão New App ID (Nova ID do aplicativo).
5. Especifique uma descrição e um identificador de pacote, você não deve usar * no identificador de pacote.
6. Clique em Submit (Enviar). O Portal de provisionamento gera sua ID do aplicativo e exibe novamente a página das ID dos aplicativos.
7. Clique em Configure (Configurar), à direita de sua ID do aplicativo. A página Configure App ID (Configurar ID do aplicativo) é exibida.
8. Marque a caixa de seleção Enable for Apple Push Notification service (Habilitar para Apple Push Notification service). Observe que existem dois tipos de certificados SSL por push: um para desenvolvimento/teste e outro para produção.
9. Clique no botão Configurar, à direita do Certificado do desenvolvimento por push SSL. A página Gerar solicitação de assinatura de certificado (CSR) é exibida.
10. Gere uma CSR utilizando um utilitário Acesso Porta-chaves conforme as instruções da página.
11. Gerar o certificado SSL.
12. Faça o download e instale o certificado SSL.
13. Opcional: repita os passos 9 a 12 para produzir o Certificado por SSL por push.
14. Clique em Concluído. A página Configure App ID (Configurar ID do aplicativo) é exibida.
15. Clique em Concluído. A página de ID do aplicativo é exibida. Observe o círculo verde ao lado da Notificação push de sua ID do aplicativo.
16. Certifique-se de salvar seus certificados SSL, eles serão usados posteriormente para comunicação entre o aplicativo e o provedor.
17. Clique na aba Provisionamento para exibir a página Perfis de provisionamento.

18 Crie um perfil de provisionamento para a nova ID do aplicativo e faça o download deste perfil.

19 Clique na aba Certificados e faça o download do novo certificado para o novo perfil de provisionamento.

Use avisos sonoros para notificações por push.

Para ativar as notificações sonoras para seu aplicativo, agrupe os arquivos de áudio da mesma forma como faria com qualquer outro ativo, mas no mesmo diretório, como os arquivos SWF e app-xml. Por exemplo:

```
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
```

A Apple suporta os seguinte formatos de arquivos de áudio (arquivos aiff, wav ou caf):

- PCM Linear
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Utilize notificações de alerta localizadas

Para usar notificações de alerta localizadas em seu aplicativo agrupe as sequências de caracteres localizadas no formulário da pasta lproj. Por exemplo, você suporta alertas em espanhol:

1. Crie uma pasta es.lproj no projeto no mesmo nível que o arquivo app-xml.
2. Na pasta es.lproj crie um arquivo de texto chamado LocalizableStrings.
3. Abra o Localizable.Strings em um editor de texto e adicione chaves de mensagem e as strings localizadas correspondentes. Por exemplo:

   ```
   "PokeMessageFormat" = "La notificación de alertas en español."
   ```

4. Salve o arquivo.
5. Quando o aplicativo receber um alerta de notificação com esta chave e o idioma do dispositivo estiver em espanhol o texto traduzido do alerta será exibido.

Configure um provedor de notificação remota.

Você precisa de um provedor de notificações remotas para enviar notificações por push para seu aplicativo. Este servidor de aplicativo age como um provedor, aceitando sua entrada por push e enviando a notificação e os dados de notificação para o APNs, que, por sua vez, envia a notificação por push para um aplicativo cliente.

Para obter mais informações sobre notificações por push de um provedor de notificações remotas, consulte Provedor de comunicação com o Apple Push Notification Service na biblioteca do desenvolvedor da Apple.

Opções de provedor de notificação por push remoto

As opções para provedor de notificação remota incluem:

- Criar seu próprio provedor baseado em um servidor APNS-php de código aberto. Você pode montar um servidor PHP usando http://code.google.com/p/apns-php/. Este projeto de Código do Google permite que você projete uma interface que combine com suas necessidades específicas.
- Use um provedor de serviços. Por exemplo, o http://urbanairship.com/ oferece um provedor APNs pronto para uso. Após se registrar neste serviço você começa a fornecer o token de seu dispositivo usando um código similar ao a seguir:

```
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);
                                    }
```

Você pode então enviar notificações de teste utilizando as ferramentas Urban Airship.

Certificados para provedor de notificação remota.

Você deve copiar o certificado SSL e a chave privada, gerada anteriormente, no local apropriado no servidor do provedor de notificação remota. Normalmente você combina estes dois arquivos em um único arquivo `.pem`. Para isso, execute as etapas a seguir:

1. Abra uma janela de terminal.
2. Crie um arquivo `.pem` com base no certificado SSL digitando o comando:

   ```
   openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
   ```

3. Crie um arquivo .pem da chave privada (`.p12`) digitando o comando:

   ```
   openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
   ```

4. Combine os dois arquivos .pem em um único arquivo digitando o comando:

   ```
   cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
   ```

5. Forneça o arquivo `.pem` combinado ao provedor do servidor quando criar seu aplicativo por push no lado do servidor.

Para obter mais informações consulte Instalando o certificado SSL e a chave no servidor no Guia de programação de notificação por push e local da Apple.

Manipular as notificações por push em um aplicativo

A manipulação de notificações por push em um aplicativo envolve:

- Configuração global do usuário e aceitação de notificações por push
- A aprovação do usuário de notificações por push individuais
- Manipulação de notificações por push e cargas de dados de notificação

Configuração e aprovação de configurações por push

Na primeira vez que um usuário abre um aplicativo com notificações por push ativada, o iOS exibe um diálogo ***appname*** **Gostaria de lhe enviar notificações por push** com os botões Não permitir e OK. Se o usuário selecionar OK o aplicativo poderá receber todos os tipos de notificações para os quais foi assinado. Se o usuário selecionar Não permitir, não receberá notificações.

Nota: *O usuário também pode ir até Configurações > Notificações para controlar tipos específicos de notificação que pode receber de cada aplicativo ativado para push.*

A Apple recomenda que sempre que um aplicativo estiver ativado, ele deve assinar notificações por push. Quando um aplicativo chama`RemoteNotifier.subscriber()` recebe um `RemoteNotificationEvent` de tipo `token`, que contém uma tokenId numérica de 32 bytes exclusiva que identifica especificamente o aplicativo no dispositivo.

Quando o dispositivo recebe uma notificação por push um pop-up é exibido com os botões de Fechar e Abrir. Se o usuário tocar em Fechar, nada acontece. Se tocar em Abrir o iOS executa o aplicativo e ele recebe uma `flash.events.RemoteNotificationEvent` do tipo `notification` conforme descrito abaixo.

A manipulação de notificações por push e dados de carga.

Quando um provedor de notificação remota envia uma notificação para um dispositivo usando uma tokenID seu aplicativo recebe um `flash.events.RemoteNotificationEvent`do tipo `notification` independentemente de estar em execução. Neste momento seu aplicativo realiza um processamento específico de notificação. Se seu aplicativo manipula os dados de notificação, você acessa os dados pela propriedade formatada para JSON`RemoteNotificationEvent.data`.

Capítulo 8: Desenvolvendo aplicativos AIR para aparelhos de televisão

Recursos do AIR for TVs

Você poderá criar aplicativos Adobe® AIR® para aparelhos de TV tais como televisores, gravadores de vídeo digital e reprodutores Blu-ray, se o dispositivo tiver o Adobe AIR for TV. O AIR for TV é otimizado para aparelhos de TV, usando, por exemplo, os aceleradores de hardware de um dispositivo para gráficos e vídeo de alto desempenho.

Aplicativos AIR para aparelhos de TV têm por base SWF, não HTML. Seu aplicativo AIR for TV pode aproveitar a aceleração de hardware, assim como outros recursos do AIR que são adequados para o ambiente "sala de estar".

Perfis de dispositivo

O AIR usa perfis para definir um conjunto de destino de dispositivos com capacidades semelhantes. Use os seguintes perfis para os aplicativos AIR for TV:

- O perfil `tv`. Use esse perfil nos aplicativos AIR que têm o dispositivo AIR for TV como destino.
- O perfil `extendedTV`. Use esse perfil se o aplicativo do AIR for TV usar extensões nativas.

Os recursos do ActionScript definidos para estes perfis são abordados em "Perfis de dispositivo" na página 249. Diferenças específicas do ActionScript para aplicativos AIR for TV são observados na Referência do ActionScript® 3.0 para Adobe® Flash® Platform.

Para obter detalhes sobre os perfis do AIR for TV, consulte "Perfis disponíveis" na página 143.

Aceleração por hardware

Os aparelhos de televisão fornecem aceleradores de hardware que melhoram substancialmente o desempenho gráfico e de vídeo no aplicativo do AIR. Para tirar proveito desses aceleradores de hardware, consulte "Considerações a respeito do design do aplicativo para AIR for TV" na página 124.

Proteção de conteúdo

O AIR for TV permite a criação de experiências enriquecedoras para o consumidor em torno de um conteúdo de vídeo de alta qualidade, de superproduções de Hollywood a filmes independentes e episódios de TV. Os provedores de conteúdo podem criar aplicativos interativos usando ferramentas do Adobe. Eles podem integrar os produtos de servidor Adobe em sua infraestrutura de distribuição de conteúdo ou trabalhar com um dos parceiros em ecossistemas da Adobe.

A proteção de conteúdo é um requisito essencial para a distribuição de vídeo de alta qualidade. O AIR for TV suporta Adobe® Flash® Access™, uma solução de monetização e proteção de conteúdo que atende aos mais rígidos requisitos de segurança dos proprietários de conteúdo, incluindo os maiores estúdios cinematográficos.

O Flash Access suporta o seguinte:

- Transmissão e download de vídeo
- Vários modelos de negócio, incluindo suporte a anúncios, assinatura, locação e venda por meio eletrônico.

- Diferentes tecnologias de fornecimento de conteúdo, incluindo HTTP Dynamic Streaming, transmissão sobre RTMP (Real Time Media Protocol) usando Flash® Media Server, e download progressivo com HTTP.

O AIR for TV também possui suporte integrado para RTMPE, a versão criptografada do RTMP, para soluções de transmissão com baixos requisitos de segurança. RTMPE e as respectivas tecnologias de verificação SWF são suportadas no Flash Media Server.

Para obter mais informações, consulte Adobe Flash Access.

Áudio multicanal

A partir do AIR 3, o AIR for TV suporta áudio multicanal para vídeos que são continuamente baixados de um servidor HTTP. O suporte inclui estes codecs:

- AC-3 (Dolby Digital)
- E-AC-3 (Dolby Digital Avançado)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

Nota: *O suporte para áudio multicanal em vídeos transmitidos pelo Adobe Flash Media Server ainda não está disponível.*

Entrada para jogos

A partir do AIR 3, o AIR for TV suporta APIs do ActionScript que permitem que os aplicativos se comuniquem com dispositivos de entrada para jogos conectados como controles, gamepads e bastões. Embora esses dispositivos sejam chamados de dispositivos de entrada para jogos, qualquer aplicativo do AIR for TV, não só para jogos, poderá usar os dispositivos.

Uma ampla gama de dispositivos de entrada para jogos com diferentes capacidades está disponível. Portanto, os dispositivos são generalizados na API para que um aplicativo possa funcionar corretamente com diferentes (e provavelmente desconhecidos) tipos de dispositivos de entrada para jogos.

A classe GameInput é o ponto de entrada da entrada para jogos das APIs do ActionScript. Para obter mais informações, consulte GameInput.

Renderização acelerada de gráficos Stage 3D

A partir do AIR 3, o AIR for TV suporta a renderização de gráficos Stage 3D acelerada. As APIs Stage3D do ActionScript são um conjunto de APIs aceleradas por GPU de baixo nível que ativam as capacidades de 2D e 3D avançadas. Essas APIs de baixo nível fornecem aos desenvolvedores a flexibilidade para aproveitar os significativos ganhos de desempenho. Você também pode usar os mecanismos para jogos que suportem as APIs Stage3D do ActionScript.

Para obter mais informações, consulte Mecanismos para jogos, 3D e Stage 3D.

Extensões nativas

Quando o aplicativo tem como destino o perfil `extendedTV` ele pode usar pacotes ANE (extensão nativa do AIR).

Geralmente, um fabricante do dispositivo fornece pacotes ANE para fornecer acesso aos recursos do dispositivo não compatíveis com o AIR. Por exemplo, uma extensão nativa poderia permitir que você altere os canais em uma televisão ou pause a reprodução em um player de vídeo.

AO empacotar um aplicativo do AIR for TV que usa pacotes ANE, você empacota o aplicativo em um arquivo do AIRN, em vez de um arquivo do AIR.

As extensões nativas para os dispositivos do AIR for TV são sempre extensões nativas *agrupadas no dispositivo*. Agrupado por dispositivo significa que as bibliotecas de extensão estão instaladas no dispositivo do AIR for TV. O pacote ANE que você inclui no pacote de seu aplicativo *nunca* inclui as bibliotecas nativas da extensão. Às vezes ele contém uma versão somente para ActionScript da extensão nativa. Essa versão somente para ActionScript é um fragmento ou simulador da extensão. O fabricante do dispositivo instala a extensão verdadeira, incluindo as bibliotecas nativas, no dispositivo.

Se você estiver desenvolvendo extensões nativas, observe o seguinte:

- Sempre consulte o fabricante do dispositivo se estiver criando uma extensão nativa do AIR for TV para os dispositivos deles.
- Em alguns dispositivos do AIR for TV, somente o fabricante do dispositivo cria as extensões nativas.
- Em todos os dispositivos do AIR for TV, o fabricante do dispositivo decide quais extensões nativas podem ser instaladas.
- As ferramentas de desenvolvimento para a criação de extensões nativas do AIR for TV variam para cada fabricante.

Para obter mais informações sobre a utilização de extensões nativas no aplicativo do AIR, consulte “Como utilizar extensões nativas para Adobe AIR” na página 151.

Para obter informações sobre a criação de extensões nativas, consulte Developing Native Extensions for Adobe AIR.

Considerações a respeito do design do aplicativo para AIR for TV

Considerações a respeito de vídeo

Diretrizes de codificação de vídeo

Ao transmitir vídeo para um aparelho de TV, a Adobe recomenda respeitar as seguintes diretrizes:

Codec de vídeo:	H.264, perfil Principal ou Alto, codificação progressiva
Resolução:	720i, 720p, 1080i ou 1080p
Taxa de quadros:	24 quadros por segundo ou 30 quadros por segundo
Codec de áudio:	O AAC-LC ou o AC-3, o 44.1 kHz, estéreo ou estes codecs de áudio multicanal: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio ou DTS-HD Master Audio
Taxa de bit composta:	até 8M bps, dependendo da largura de banda disponível
Taxa de bit de áudio:	até 192 Kbps
Proporção de pixels:	1 × 1

A Adobe recomenda usar o codec H.264 para vídeo entregue a dispositivos AIR for TV.

Nota: *O AIR for TV também suporta vídeo codificado com Sorenson Spark ou codecs On2 VP6. No entanto, o hardware não decodifica nem apresenta esses codecs. Em vez disso, o runtime decodifica esses codecs usando o software; portanto, o vídeo é exibido com uma taxa de quadros muito menor. Por isso, use o H.264 se for possível.*

A classe StageVideo

O AIR for TV suporta decodificação e apresentação de hardware de vídeo H.264 codificado. Use a classe StageVideo para habilitar esse recurso.

Consulte Uso da classe StageVideo para apresentação acelerada por hardware no *Guia do desenvolvedor do ActionScript 3.0* para obter detalhes sobre:

- a API da classe StageVideo e das classes relacionadas.
- limitações ao uso da classe StageVideo.

Para um melhor suporte aos aplicativos AIR existentes que usam o objeto Video para vídeo H.264 codificado, o AIR for TV usa um objeto StageVideo *internamente*. Fazer isso significa que a reprodução de vídeo tira proveito da decodificação e da apresentação do hardware. No entanto, o objeto Video está sujeito às mesmas restrições de um objeto StageVideo. Por exemplo, se o aplicativo tentar girar o vídeo, nenhum giro ocorrerá, uma vez que o hardware, não o runtime, estará apresentando o vídeo.

Contudo, ao escrever novos aplicativos, use o objeto StageVideo para vídeo H.264 codificado.

Para obter um exemplo do uso da classe StageVideo, consulte Fornecendo vídeo e conteúdo para a Plataforma Flash em TV.

Diretrizes de fornecimento de vídeo

Em um dispositivo AIR for TV, a largura de banda disponível da rede pode variar durante a reprodução de vídeo. Essas variações podem ocorrer, por exemplo, quando outro usuário começa a usar a mesma conexão com a Internet.

Portanto, a Adobe recomenda que o seu sistema de fornecimento de vídeo use capacidades de taxa de bits adaptáveis. Por exemplo, no lado do servidor, o Flash Media Server suporta capacidades de taxa de bits adaptáveis. No lado do cliente, você pode usar a estrutura de mídia de fonte aberta Open Source Media Framework (OSMF).

Os seguintes protocolos estão disponíveis para o fornecimento de conteúdo de vídeo em uma rede para um aplicativo do AIR for TV:

- Transmissão Dinâmica em HTTP e HTTPS (formato F4F)
- Transmissão em RTMP, RTMPE, RTMFP, RTMPT e RTMPTE
- Download Progressivo em HTTP e HTTPS

Para obter mais informações, consulte o seguinte:

- Guia do desenvolvedor do Adobe Flash Media Server
- Open Source Media Framework

Considerações a respeito de áudio

O ActionScript para reprodução de som não é diferente nos aplicativos AIR for TV em comparação com outros aplicativos AIR. Para mais informações, consulte Trabalho com som no *Guia do desenvolvedor do ActionScript 3.0*.

Quanto ao suporte de áudio multicanal no AIR for TV, considere o seguinte:

- O AIR for TV suporta áudio multicanal para vídeos que são continuamente baixados de um servidor HTTP. O suporte para áudio multicanal em vídeos transmitidos pelo Adobe Flash Media Server ainda não está disponível.
- Embora o AIR for TV seja compatível com muitos codecs de áudio, nem todos os dispositivos para *o AIR for TV* são compatíveis com o conjunto inteiro. Use o método flash.system.Capabilities`hasMultiChannelAudio()` para verificar se o dispositivo do AIR for TV é compatível com o codec de áudio específico de multicanais como o AC-3.

  Por exemplo, considere um aplicativo que progressivamente baixa um arquivo de vídeo de um servidor. O servidor possui diferentes arquivos de vídeo H.264 que são compatíveis com diferentes codecs de áudio de multi-canais. O aplicativo poderá utilizar `hasMultiChannelAudio()` para determinar qual arquivo de vídeo deve ser solicitado do servidor. Como alternativa, o aplicativo poderá enviar ao servidor uma string contida em `Capabilities.serverString`. A string indica quais codecs de áudio multicanais estão disponíveis, permitindo ao servidor selecionar o arquivo de vídeo correto.
- Ao usar um dos codecs de áudio DTS, existem situações em que `hasMultiChannelAudio()` retorna `true`, mas o áudio DTS não é reproduzido.

  Por exemplo, considere um player de Blu-raio com uma saída S/PDIF, conectada a um velho ampliador. O velho ampliador não apoia DTS, mas S/PDIF não tem nenhum protocolo para notificar o player de Blu-raio. Se o player de Blu-raio enviar a corrente DTS ao velho ampliador, o usuário não ouve nada. Por isso, como uma prática recomendada usando DTS, forneça uma interface do usuário para que o usuário possa indicar se nenhum som está jogando. Então, o seu aplicativo pode reverter a um codec diferente.

A seguinte tabela resume quando usar diferentes codecs de áudio nos aplicativos do AIR for TV. A tabela também indica quando os dispositivos do AIR for TV utilizam aceleradores de hardware para decodificar um codec de áudio. A decodificação de hardware melhora o desempenho e evita a sobrecarga da CPU.

Codec de áudio	**Disponibilidade para dispositivo do AIR for TV**	**Decodificação de hardware**	**Quando usar este codec de áudio**	**Mais informações**
AAC	Sempre	Sempre	Em vídeos codificados com H.264. Em transmissão de áudio tal como um serviço de transmissão de música na Internet.	Ao usar um fluxo contínuo AAC exclusivamente de áudio, encapsule o fluxo contínuo de áudio num recipiente MP4.
mp3	Sempre	Não	Para sons contidos nos arquivos SWF do aplicativo. Em vídeos codificados com Sorenson Spark ou On2 VP6.	Um vídeo H.264 que utilize mp3 para áudio não é reproduzido em dispositivos do AIR for TV.

Codec de áudio	Disponibilidade para dispositivo do AIR for TV	Decodificação de hardware	Quando usar este codec de áudio	Mais informações
AC-3 (Dolby Digital) E-AC-3 (Dolby Digital Avançado) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Verificar	Sim	Em vídeos codificados com H.264.	Geralmente, o AIR for TV envia um fluxo de áudio multicanal para um receptor externo de áudio/vídeo que decodifica e reproduz o áudio.
Speex	Sempre	Não	Recebendo uma transmissão de voz ao vivo.	Um vídeo H.264 que utilize Speex para o áudio não é reproduzido em dispositivos do AIR for TV. Use Speex somente com vídeos codificados com Sorenson Spark ou On2 VP6.
NellyMoser	Sempre	Não	Recebendo uma transmissão de voz ao vivo.	Um vídeo H.264 que utilize NellyMoser para o áudio não é reproduzido em dispositivos do AIR for TV. Use NellyMoser somente com vídeos codificados com Sorenson Spark ou On2 VP6.

Nota: *Alguns arquivos de vídeo contêm dois fluxos de áudio. Por exemplo, um arquivo de vídeo poderá conter um fluxo AAC e um fluxo AC3. O AIR for TV não é compatível com tais arquivos de vídeo, e o uso desse arquivo poderá causar a ausência de som do vídeo.*

Aceleração por hardware de gráfico

Usando aceleração gráfica de hardware

Os dispositivos AIR for TV fornecem aceleração de hardware para operações com imagens 2D. Os aceleradores de imagem de hardware do dispositivo liberam a CPU para realizar as seguintes operações:

- Renderização de bitmaps
- Escala de bitmap
- Mesclagem de bitmap
- Preenchimento sólido do retângulo

Esta aceleração gráfica de hardware significa que muitas operações com imagens no aplicativo do AIR for TV podem estar executando com alto desempenho. Algumas dessas operações incluem:

- Transições de deslocamento
- Transições de escala
- Intensificação e esmaecimento gradual da imagem (fade in e fade out)
- Imagens de composição múltipla com alfa

Para obter os benefícios de desempenho da aceleração gráfica de hardware para estes tipos de operações, use uma das seguintes técnicas:

- Defina a propriedade `cacheAsBitmap` como `true` nos objetos MovieClip e em outros objetos de exibição que possuem conteúdo na maioria das vezes inalterável. Em seguida, execute as transições laterais, as transições de intensidade e as mesclagens de alfa nesses objetos.
- Use a propriedade `cacheAsBitmapMatrix` nos objetos de exibição que você deseja escalar ou converter (aplicar reposicionamento de x e y).

  Usando operações da classe Matrix para escala e conversão, os aceleradores de hardware do dispositivo executam as operações. Alternativamente, considere o cenário no qual você altera as dimensões de um objeto de exibição que tem sua propriedade `cacheAsBitmap` definida como `true`. Quando as dimensões são alteradas, o software do runtime redesenha o bitmap. O redesenho com o software produz um desempenho pior do que a escala com aceleração de hardware usando uma operação Matrix.

  Por exemplo, considere um aplicativo que exibe uma imagem que se expande quando um usuário a seleciona. Use os tempos múltiplos de operação de escala Matrix para criar a ilusão de uma imagem em expansão. Contudo, dependendo da imagem original e da imagem final, a qualidade da imagem final poderá ser inaceitável. Portanto, redefina as dimensões do objeto de exibição depois que as operações de expansão forem concluídas. Uma vez que `cacheAsBitmap` seja `true`, o software do runtime redesenhará o objeto de exibição, mas somente uma vez, e proporcionará uma imagem de alta qualidade.

  Nota: *Geralmente, os dispositivos AIR for TV não suportam rotação e inclinação de hardware acelerado. Portanto, se você especificar a rotação e a inclinação na classe Matrix, o AIR for TV executará todas as operações de Matrix no software. Essas operações de software podem ter um impacto prejudicial ao desempenho.*
- Use a classe BitmapData para criar o comportamento de armazenamento de bitmap personalizado em cache.

Para obter mais informações sobre cache de bitmap, consulte o seguinte:

- Cache de objetos de exibição
- Cache de bitmap
- Cache manual de bitmap

Gerenciando a memória gráfica

Para executar operações de imagem acelerada, os aceleradores de hardware usam memória gráfica especial. Se o seu aplicativo usar toda a memória gráfica, executará lentamente porque o AIR for TV reverterá para o uso do software das operações de imagem.

Para gerenciar o uso de memória gráfico pelo seu aplicativo:

- Quando você estiver usando uma imagem ou outro dado de bitmap, libere sua memória gráfica associada. Para tanto, chame o método `dispose()` da propriedade `bitmapData` do objeto Bitmap. Por exemplo:

  ```
  myBitmap.bitmapData.dispose();
  ```

  Nota: *A liberação da referência para o objeto BitmapData não libera a memória gráfica imediatamente. O coletor de lixo do runtime eventualmente libera a memória gráfica, mas a chamada de* `dispose()` *oferece maior controle para o seu aplicativo.*
- Use o PerfMaster Deluxe, um aplicativo do AIR fornecido pela Adobe, para entender melhor a aceleração gráfica de hardware no seu dispositivo de destino. Este aplicativo mostra os quadros por segundo para executar várias operações. Use o PerfMaster Deluxe para comparar diferentes implementações da mesma operação. Por exemplo, compare a movimentação de uma imagem em bitmap com a movimentação de uma imagem em vetor. O PerfMaster Deluxe está disponível em Flash Platform para TV.

Gerenciando a lista de exibição

Para tornar um objeto de exibição invisível, defina a propriedade `visible` do objeto como `false`. Em seguida, o objeto continuará na lista de exibição, mas o AIR for TV não irá renderizá-lo nem exibi-lo. Essa técnica é útil para objetos que frequentemente entram e saem de visualização, porque demanda apenas um pouco mais de processamento. No entanto, a definição da propriedade `visible` em `false` não libera nenhum dos recursos do objeto. Portanto, ao concluir a exibição de um objeto ou, ao menos, concluí-la por algum tempo, remova o objeto da lista de exibição. Do mesmo modo, defina todas as referências ao objeto como `null`. Essas ações permitem que o coletor de lixo libere os recursos do objeto.

uso de imagem PNG e JPEG

PNG e JPEG são dois formatos de imagem comuns em aplicativos. Quanto a esses formatos de imagem no AIR for TV, considere o seguinte:

- O AIR for TV geralmente usa aceleração de hardware para decodificar arquivos JPEG.
- O AIR for TV geralmente usa software para decodificar arquivos PNG. A decodificação de arquivos PNG no software é rápida.
- PNG é o único formato de plataforma cruzada que suporta transparência (um canal alfa).

Portanto, use esses formatos de imagem da seguinte maneira em seus aplicativos:

- Use arquivos JPEG para as fotografias tirarem proveito da decodificação por aceleração de hardware.
- Use os arquivos de imagem PNG para os elementos de interface do usuário Os elementos de interface do usuário pode ter uma configuração alfa, e a decodificação com software proporciona um desempenho suficientemente rápido para os elementos de interface do usuário.

O palco nos aplicativos AIR for TV

Ao criar um aplicativo do AIR for TV, considere o seguinte ao trabalhar com a classe Stage:

- Resolução de tela
- A área de visualização segura
- O modo de escala do palco
- O alinhamento do palco
- O estado de exibição do palco
- Desenhando para vários tamanhos de tela
- A configuração de qualidade do palco

Resolução de tela

Atualmente, os aparelhos de TV costumam ter uma das seguintes resoluções de tela: 540p, 720p e 1080p. Essas resoluções de tela resultam nos seguintes valores na classe ActionScript Capabilities:

Resolução de tela	`Capabilities.screenResolutionX`	`Capabilities.screenResolutionY`
540p	960	540
720p	1280	720
1080p	1920	1080

Para escrever um aplicativo do AIR for TV de tela inteira para um dispositivo específico, defina os códigos `Stage.stageWidth` and `Stage.stageHeight` para a resolução de tela do dispositivo. Contudo, para escrever um aplicativo de tela inteira que execute em vários dispositivos, use as propriedades `Capabilities.screenResolutionX` e `Capabilities.screenResolutionY` para definir as dimensões do palco.

Por exemplo:

```
stage.stageWidth = Capabilities.screenResolutionX;
stage.stageHeight = Capabilities.screenResolutionY;
```

A área de visualização segura

A *área de visualização segura* em um televisor é uma área da tela que é afastada das bordas da tela. Essa área é inserida suficientemente longe para que o usuário possa ver a área inteira, sem que o bisel do televisor oculte qualquer parte da área. Uma vez que o bisel, que é a moldura física em torno da tela, varia de acordo com o fabricante, o afastamento necessário também varia. A área de visualização segura tenta garantir que a área visível da tela. A área de visualização segura também é conhecida como *área de título segura*.

Overscan é a área da tela que não é visível porque está atrás do bisel.

A Adobe recomenda um afastamento de 7,5% em cada borda da tela. Por exemplo:

A área de visualização segura para uma resolução de tela de 1920 x 1080

Considere sempre a área de visualização segura ao projetar um aplicativo do AIR for TV de tela inteira:

- Use a tela inteira para os planos de fundo, tais como imagens de fundo ou cores de fundo.
- Use a área de visualização segura somente para elementos de aplicativo críticos tais como texto, imagens, vídeo e itens de interface do usuário como botões.

A tabela a seguir mostra as dimensões da área de visualização segura de cada resolução de tela típica, usando um afastamento de 7,5%.

Resolução de tela	Largura e altura de uma área de visualização segura	Largura do afastamento à direita e à esquerda	Altura do afastamento superior e inferior
960x 540	816 x 460	72	40

Resolução de tela	Largura e altura de uma área de visualização segura	Largura do afastamento à direita e à esquerda	Altura do afastamento superior e inferior
1280x 720	1088x 612	96	54
1920 x 1080	1632 x 918	144	81

Contudo, uma prática melhor é sempre calcular dinamicamente a área de visualização segura. Por exemplo:

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

O modo de escala do palco

Defina `Stage.scaleMode` como `StageScaleMode.NO_SCALE` e fique na escuta de eventos de redimensionamento de palco.

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

Essa configuração torna as coordenadas do palco iguais às coordenadas do pixel. Junto com o estado de exibição `FULL_SCREEN_INTERACTIVE` e o alinhamento de palco `TOP_LEFT`, essa configuração permite que você use eficientemente a área de visualização segura.

Especificamente, nos aplicativos de tela inteira esse modo de escala significa que as propriedades `stageWidth` e `stageHeight` da classe Stage correspondem às propriedades `screenResolutionX` e `screenResolutionY` da classe Capabilities.

Além disso, quando a janela do aplicativo muda de tamanho, os conteúdos do palco mantêm seu tamanho definido. O runtime não executa automaticamente nenhum layout ou escala. Do mesmo modo, o runtime despacha o evento `resize` da classe Stage quando a janela muda de tamanho. Portanto, você tem controle completo sobre como ajustar os conteúdos do aplicativo quando o aplicativo inicial e quando a janela do aplicativo muda de tamanho.

***Nota:** O comportamento `NO_SCALE` é igual ao de qualquer aplicativo do AIR. Nos aplicativos AIR for TV, no entanto, essa configuração é crítica para usar a área de visualização segura.*

O alinhamento do palco

Defina `Stage.align` como `StageAlign.TOP_LEFT`:

```
stage.align = StageAlign.TOP_LEFT;
```

Esse alinhamento coloca a coordenada `0,0` no canto superior esquerdo da tela, o que é conveniente para a colocação de conteúdo usando ActionScript.

Junto com o modo de escala `NO_SCALE` e com o estado de exibição `FULL_SCREEN_INTERACTIVE`, essa configuração permite que você use eficientemente a área de visualização segura.

O estado de exibição do palco

Defina `Stage.displayState` num aplicativo do AIR for TV de tela inteira como `StageDisplayState.FULL_SCREEN_INTERACTIVE`:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Este valor define o aplicativo do AIR para expandir o palco sobre a tela inteira com a entrada de teclado ativada.

A Adobe recomenda o uso da configuração `FULL_SCREEN_INTERACTIVE`. Junto com o modo de escala `NO_SCALE` e com o estado de alinhamento `TOP_LEFT`, essa configuração permite que você use eficientemente a área de visualização segura.

Portanto, nos aplicativos de tela inteira, em um manipulador do evento `ADDED_TO_STAGE` na classe document principal, faça o seguinte:

```
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;
}
```

Em seguida, no manipulador do evento `RESIZE`:

- Compare os tamanhos da resolução da tela com a largura e a altura do palco. Se forem iguais, o evento `RESIZE` ocorreu, porque o estado de exibição do palco mudou para `FULL_SCREEN_INTERACTIVE`.
- Calcule e salve as dimensões da área de visualização segura e dos afastamentos correspondentes.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {

        // Calculate and save safe viewing area dimensions.
    }
}
```

Quando as dimensões do palco forem iguais a `Capabilities.screenResolutionX` e `screenResolutionY`, o AIR for TV fará com que o hardware libere a melhor fidelidade possível para os vídeos e imagens.

Nota: *A fidelidade com a qual as imagens e o vídeo são exibidos numa tela de TV pode diferir dos valores `Capabilities.screenResolutionX` e `screenResolutionY`, que dependem do dispositivo que está executando o AIR for TV. Por exemplo, uma caixa set-top que executa o AIR for TV pode ter uma resolução de tela de 1280 x 720, e a TC conectada pode ter uma resolução de tela de 1920 x 1080. Contudo, o AIR for TV faz com que o hardware libere a melhor fidelidade possível. Portanto, nesse exemplo, o hardware exibe um vídeo de 1080p usando uma resolução de tela de 1920 x 1080.*

Desenhando para vários tamanhos de tela

Você pode desenvolver o mesmo aplicativo do AIR for TV de tela inteira para trabalhar e bem e ter uma boa aparência em vários dispositivos AIR for TV. Faça o seguinte:

1. Defina as propriedades de palco `scaleMode`, `align` e `displayState` com os valores recomendados: `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` e `StageDisplayState.FULL_SCREEN_INTERACTIVE`, respectivamente.
2. Configure a área de visualização segura com base em `Capabilities.screenResolutionX` e `Capabilities.screenResolutionY`.
3. Ajuste o tamanho e o layout do conteúdo de acordo com a largura e a altura da área de visualização segura.

Embora os objetos do conteúdo sejam grandes, especialmente quando comparados com aplicativos de dispositivos móveis, os conceitos tais como layout dinâmico, posicionamento relativo e conteúdo adaptável são iguais. Para mais informações sobre o ActionScript em apoio a esses contextos, consulte Criação de conteúdo do Flash móvel para vários tamanhos de tela.

A qualidade do palco

A propriedade `Stage.quality` do aplicativo do AIR for TV é sempre `StageQuality.High`. Ela não pode ser alterada.

Essa propriedade especifica a qualidade de renderização de todos os objetos Stage.

Tratamento de entrada do controle remoto

Os usuários geralmente interagem com o seu aplicativo do AIR for TV usando um controle remoto. No entanto, manipule a entrada de tecla da mesma forma que você manipula a entrada de tecla de um teclado num aplicativo para computador pessoal. Especificamente, manipulou o evento `KeyboardEvent.KEY_DOWN`. Para mais informações, consulte Captura da entrada do teclado no *Guia do desenvolvedor do ActionScript 3.0*.

As teclas do controle remoto mapeiam constantes do ActionScript. Por exemplo, as teclas do teclado direcional de um controle remoto mapeiam da seguinte maneira:

Tecla de teclado direcional do controle remoto	Constante do ActionScript 3.0
Up	`Keyboard.UP`
Para baixo	`Keyboard.DOWN`
À Esquerda	`Keyboard.LEFT`
À Direita	`Keyboard.RIGHT`
OK ou Selecionar	`Keyboard.ENTER`

O AIR 2.5 adicionou muitas outras constantes de Teclado para suportar entrada de controle remoto. Para mais informações, consulte Classe keyboard no livro *Referência do ActionScript® 3.0 para Adobe® Flash® Platform*.

Para garantir que seu aplicativo funcione no máximo possível de dispositivos, a Adobe recomenda o seguinte:

- Use somente teclas de teclado direcional, se possível.

  Diferentes dispositivos de controle remoto possuem diferentes conjuntos de teclas. No entanto, eles geralmente possuem sempre teclas de teclado direcional.

  Por exemplo, um controle remoto de um reprodutor Blu-ray geralmente não têm as teclas “canal acima” e “canal abaixo”. Mesmo as teclas para reproduzir, pausar ou parar não existem em todos os controles remotos.

- Use as teclas Menu e Info se o aplicativo precisar de algo mais do que as teclas de teclado direcional.

  As teclas Menu e Info são as próximas teclas mais comuns nos controles remotos.

- Considere o uso frequente de controles remotos universais.

  Mesmo que você esteja criando um aplicativo para um dispositivo específico, entenda que muitos usuários não usam um controle remoto que vem com o dispositivo. Em vez disso, eles usam um controle remoto universal. Do mesmo modo, os usuários nem sempre programam seus controle remotos universais para que correspondam a todas as teclas do controle remoto do dispositivo. Portanto, é recomendável usar somente as teclas mais comuns.

- Certifique-se de que o usuário sempre escape de uma situação usando uma das teclas do teclado direcional.

Ás vezes o seu aplicativo tem um bom motivo para usar uma tecla que não seja das mais comuns nos controles remotos. Proporcionar uma rota de escape com teclas de teclado direcional faz o seu aplicativo ter um comportamento agradável em todos os dispositivos.

- Não exija entrada para ponteiro exceto se você conhecer o destino para o qual o dispositivo do AIR for TV tenha uma capacidade de entrada para ponteiro.

  Embora muitos aplicativos de desktop prevejam entrada para mouse, a maioria dos televisores não suporta entrada para mouse. Portanto, se você estiver convertendo aplicativos de desktop para serem executados em televisores, certifique-se de alterar o aplicativo para não prever entrada para mouse. Essas modificações incluem mudanças na manipulação de eventos e alterações nas instruções ao usuário. Por exemplo, quando a tela de inicialização de um aplicativo é exibida, não exiba um texto que diga “Clique para iniciar”.

Gerenciamento do foco

Quando um elemento de interface do usuário mantém o foco em um aplicativo de computador pessoal, é objetivo do usuário introduzir eventos tais como eventos de teclado e de mouse. Além disso, um aplicativo destaca o elemento de interface do usuário com o foco. O gerenciamento do foco em um aplicativo do AIR for TV é diferente do gerenciamento do foco em um aplicativo de computador pessoal, porque:

- Os aplicativos de computador pessoal frequentemente usam a tecla tab para mudar o foco para o próximo elemento da interface do usuário. O uso da tecla tab não se aplica aos aplicativos AIR for TV. Os dispositivos de controle remoto geralmente possuem uma tecla tab. Portanto, o gerenciamento do foco com a propriedade `tabEnabled` de um DisplayObject como o de um computador pessoal não é aplicável.
- Os aplicativos de computador pessoal frequentemente esperam que o usuário use o mouse para focalizar um elemento da interface do usuário.

Portanto, no seu aplicativo, faça o seguinte:

- Adicione ao Palco um evento do escutador que escute eventos do teclado tais como `KeyboardEvent.KEY_DOWN`.
- Providencie uma lógica de aplicação para determinar qual elemento da interface do usuário será destacado para o usuário final. Certifique-se de destacar um elemento da interface do usuário quando o aplicativo iniciar.
- Com base em sua lógica de aplicação, despache o evento Keyboard que o Palco recebeu para o objeto apropriado do elemento da interface do usuário.

  Você também pode usar `Stage.focus` ou `Stage.assignFocus()` para dirigir o foco para um elemento da interface do usuário. Também é possível adicionar um escutador de eventos ao DisplayObject de forma que ele receba eventos do teclado.

Design de interface de usuário

Faça a interface do usuário de um aplicativo do AIR for TV funcionar bem em televisores incorporando essas recomendações sobre:

- a responsividade do aplicativo
- a usabilidade do aplicativo
- a personalidade e as expectativas do usuário

Responsividade

Use as seguintes dicas para tornar um aplicativo do AIR for TV o mais responsivo possível.

- Faça com que o arquivo SWF inicial do aplicativo seja o menor possível.

  No arquivo SWF inicial, carregue apenas os recursos necessários para iniciar o aplicativo. Por exemplo, carregue somente a imagem da tela de inicialização do aplicativo.

  Embora essas recomendações sejam válidas para os aplicativos AIR para computador pessoal, são ainda mais importantes para os dispositivos AIR for TV. Por exemplo, os dispositivos AIR for TV não têm potência de processamento igual à dos computadores pessoais. Do mesmo modo, eles armazenam o aplicativo na memória flash, cujo acesso não é tão rápido como nos discos rígidos dos computadores pessoais.

- Certifique-se de que o aplicativo execute a uma taxa de quadros de ao menos 20 quadros por segundo.

  Projete seus gráficos para atingir essa meta. A complexidade de suas operações gráficas pode afetar a taxa de quadros por segundo. Para obter dicas sobre como melhorar o desempenho de renderização, consulte Otimizando o desempenho para a plataforma Adobe Flash.

  Nota: *O hardware gráfico nos dispositivos AIR for TV geralmente atualizam a tela à taxa de 60 Hz ou 120 Hz (60 ou 120 vezes por segundo). O hardware varre o palco em busca de atualizações a, por exemplo, 30 quadros por segundo ou 60 quadros por segundo para exibição em tela de 60 Hz ou 120 Hz. Contudo, se o usuário experimentará essas altas taxas de quadro dependerá da complexidade das operações gráficas do aplicativo.*

- Atualiza a tela dentro de 100 - 200 milésimos de segundo a partir da entrada do usuário.

  Os usuários ficam impacientes quando as atualizações demoram demais, o que geralmente resulta em vários toques nas teclas.

Usabilidade

Os usuários dos aplicativos AIR for TV estão em um ambiente de "sala de estar". Eles estão sentados na sala vendo TV a cerca de 3 a 5 metros de distância. A sala às vezes é escura. Eles geralmente usam um aparelho de controle remoto para seus comandos. Mais de uma pessoa pode usar o aplicativo; às vezes usam juntos e, às vezes, em sequência.

Portanto, ao projetar a interface do usuário para melhor usabilidade num televisor, considere o seguinte:

- Faça com que os elementos da interface do usuário sejam grandes.

  Ao conceber texto, botões ou qualquer outro elemento de interface, considere que o usuário está sentado em uma sala. Faça tudo fácil de ver e ouvir, por exemplo, a 5 metros de distância. Não caia na tentação de abarrotar a tela só porque ela é grande.

- Use um bom contraste para tornar o conteúdo fácil de ver e ler de qualquer parte da sala.
- Faça com que fique óbvio qual elemento da interface do usuário está em foco, tornando-o brilhante.
- Use movimento apenas quando necessário. Por exemplo, passar de uma tela para outra para obter continuidade pode funcionar bem. Contudo, o movimento pode causar distração se não ajuda o usuário a navegar ou se não é intrínseco ao aplicativo.
- Forneça sempre um meio óbvio para o usuário retornar na interface.

Para mais informações sobre o uso do controle remoto, consulte "Tratamento de entrada do controle remoto" na página 133.

Personalidade e expectativas do usuário

Considere que os usuários dos aplicativos AIR for TV estão geralmente procurando entretenimento de qualidade na TV em um ambiente divertido e descontraído. Eles não são necessariamente especialistas em computadores ou em tecnologia.

Portanto, projete os seus aplicativos AIR for TV com as seguintes características:

- Não use termos técnicos.
- Evite diálogos modais.
- Use instruções informais e fáceis, apropriadas a um ambiente de sala de estar, não a um ambiente de trabalho ou técnico.
- Use imagens que tenham a alta qualidade de produção que os telespectadores esperam.
- Crie uma interface de usuário que trabalhe de maneira fácil com um dispositivo de controle remoto. Não use interface de usuário ou elementos de design que se adaptem melhor a um aplicativo de desktop ou móvel. Por exemplo, as interfaces de usuário em dispositivos de desktop e móveis geralmente envolvem apontar ou clicar botões com um mouse ou com o dedo.

Fontes e texto

Você pode usar fontes do dispositivo ou fontes incorporadas no seu aplicativo AIT para TV.

Fontes de dispositivo são as fontes que são instaladas em um dispositivo. Todos os dispositivos AIR for TV têm as seguintes fontes de dispositivo:

Nome da fonte	Descrição
`_sans`	A fonte de dispositivo `_sans` é uma fonte sans-serif. A fonte de dispositivo `_sans` . instalada nos dispositivos AIR for TV é a Myriad Pro. Normalmente, uma fonte sem serifa terá uma melhor aparência na TV do que fontes com serifa, devido à distância de visualização.
`_serif`	A fonte de dispositivo `_serif` é uma fonte serif. A fonte de dispositivo `serif`instaladas em todos os dispositivos AIR for TV é a Minion Pro.
`_typewriter`	A fonte de dispositivo `_typewriter` é uma fonte monospace. A fonte de dispositivo `_typewriter`instalada em todos os dispositivos AIR for TV é a Courier Std.

Todos os dispositivos AIR for TV têm as seguintes fontes de dispositivo asiáticas:

Nome da fonte	Idioma	Categoria de fonte	código do local
RyoGothicPlusN-Regular	Japonês	sans	ja
RyoTextPlusN-Regular	Japonês	serif	ja
AdobeGothicStd-Light	Coreano	sans	ko
AdobeHeitiStd-Regular	Chinês Simplificado	sans	zh_CN
AdobeSongStd-Light	Chinês Simplificado	serif	zh_CN
AdobeMingStd-Light	Chinês Tradicional	serif	zh_TW e zh_HK

Essas fontes de dispositivo do AIR for TV são:

- Da biblioteca de fontes Adobe® Type Library

- Assistir a bons programas de televisão
- Projetado para titulação de vídeo
- São fontes de contorno, não fontes bitmap

Nota: *Os fabricantes de dispositivos incluem outras fontes de dispositivo no dispositivo. Essas fontes de dispositivo fornecidas pelos fabricantes são instaladas em adição às fontes do dispositivo AIR for TV.*

A Adobe fornece um aplicativo chamado FontMaster Deluxe, que exibe todas as fontes de dispositivo presentes no dispositivo. O aplicativo está disponível em Flash Platform para TV.

Você também pode usar fontes de dispositivo no seu aplicativo AIT para TV. Para mais informações, consulte Renderização avançada de texto no *Guia do desenvolvedor do ActionScript 3.0*.

A Adobe recomenda o seguinte a respeito dos campos de texto TLF:

- Use os campos de texto TLF para textos em idiomas asiáticos para tirar proveito do local em que o aplicativo está sendo executado. Defina a propriedade `locale` no objeto TextLayoutFormat associado ao objeto TLFTextField. Para mais informações, consulte Seleção de código de idiomas no *Guia do desenvolvedor do ActionScript 3.0*.
- Especifique o nome da fonte na propriedade `fontFamily` no objeto TextLayoutFormat se a fonte não for uma das fontes do dispositivo AIR for TV. O AIR for TV usará a fonte se ela estiver disponível no dispositivo. Se a fonte que você deseja não estiver no dispositivo, baseada na configuração de `locale`, o AIR for TV substituirá a fonte de dispositivo apropriada do AIR for TV.
- Especifique `_sans`, `_serif`, ou `_typewriter` para a propriedade `fontFamily`, juntamente com a configuração da propriedade `locale` para fazer com que o AIR for TV escolha uma fonte de dispositivo correta para o AIR for TV. Dependendo do local, o AIR for TV escolhe seu conjunto de fontes de dispositivo asiáticas ou seu conjunto de fontes de dispositivo não asiáticas. Essas configurações proporcionam um modo fácil de usar automaticamente a fonte correta para a maior parte das variações idiomáticas asiáticas ou do inglês.

Nota: *Se você estiver usando campos de texto clássicos para texto em idioma asiático, especifique um nome de fonte de uma fonte de dispositivo do AIR for TV para garantir uma renderização adequada. Se você souber que outra fonte está instalada no dispositivo de destino, também poderá especificá-la.*

Quanto ao desempenho do aplicativo, considere o seguinte:

- Os campos de texto clássicos proporcionam um desempenho mais rápido do que os campos de texto TLF.
- Um campo de texto clássico que usa fontes bitmap proporciona um desempenho melhor.

  As fontes bitmap proporcionam um bitmap para cada caractere, ao contrário de fontes geométricas, que fornecem apenas dados de contorno de cada caractere. As duas fontes de dispositivo podem ser fontes bitmap.
- Se você especificar uma fonte de dispositivo, certifique-se de que a fonte de dispositivo esteja instalada no dispositivo de destino. Se a fonte não estiver instalada no dispositivo, o AIR for TV buscará e usará outra fonte que esteja instalada no dispositivo. Contudo, esse comportamento torna mais lento o desempenho do aplicativo.
- Como ocorre com qualquer objeto de exibição, se um objeto TextField for na maioria das vezes inalterável, defina a propriedade `cacheAsBitmap` do objeto como `true`. Essa configuração melhora o desempenho em transições como mudança de intensidade (fading), deslocamento e mesclagem de alfa. Use `cacheAsBitmapMatrix` para escala e conversão. Para obter mais informações, consulte “Aceleração por hardware de gráfico” na página 127.

Segurança do sistema de arquivos

Os aplicativos AIR for TV são aplicativos AIR e, portanto, podem acessar o sistema de arquivos do dispositivo. No entanto, em um dispositivo de "sala de estar" é muito importante que um aplicativo não possa acessar os arquivos de sistema do dispositivo ou os arquivos de outros aplicativos. Os usuários de TVs e dispositivos associados não esperam nem toleram qualquer falha no dispositivo — acima de tudo, eles estão assistindo à TV.

Portanto, o aplicativo do AIR for TV tem uma visualização limitada do sistema de arquivos do dispositivo. Usando o ActionScript 3.0, seu aplicativo pode acessar somente diretórios específicos (e seus subdiretórios). Além disso, os nomes dos diretórios que você usa no ActionScript não são nomes de diretórios reais no dispositivo. Esta camada extra protege os aplicativos AIR for TV contra o acesso mal-intencionado ou inadvertido aos arquivos locais que não pertençam a eles.

Para obter mais detalhes, consulte Visualização de diretórios para aplicativos AIR for TV.

A caixa de proteção do aplicativo do AIR

Os aplicativos AIR for TV são executados em uma caixa de segurança de aplicativo do AIR, descrita em A caixa de proteção do aplicativo do AIR.

A única diferença com os aplicativos AIR for TV é que esses têm acesso limitado ao sistema de arquivos, conforme descrito em "Segurança do sistema de arquivos" na página 138.

Ciclo de vida do aplicativo

Ao contrário de um ambiente de computação pessoal, o usuário final não pode fechar a janela na qual o aplicativo do AIR for TV está executando. Portanto, forneça um mecanismo de interface do usuário para saída do aplicativo.

Geralmente, um dispositivo permite que o usuário saia incondicionalmente de um aplicativo com a tecla de saída do controle remoto. No entanto, o AIR for TV não despacha o evento `flash.events.Event.EXITING` para o aplicativo. Portanto, salve o estado do aplicativo frequentemente para que o aplicativo possa recuperar-se para um estado razoável quando iniciar na próxima vez.

Cookies HTTP

O AIR for TV é compatível com os cookies contínuos de HTTP e com o cookies de sessão. O AIR for TV armazena cada cookie do aplicativo do AIR em um diretório específico do aplicativo:

`/app-storage/<app id>/Local Store`

O arquivo de cookie é nomeado `cookies`.

***Nota:** O AIR em outros dispositivos, como dispositivos pessoais, não armazenam cookies separadamente para cada aplicativo. O armazenamento de cookie específico de aplicativo é compatível com o modelo de aplicativo e de segurança do sistema do AIR for TV.*

Use a propriedade `URLRequest.manageCookies` do ActionScript da seguinte maneira:

- Defina `manageCookies` como `true`. Este valor é o padrão. Isso significa que o AIR for TV adiciona cookies automaticamente às solicitações ao HTTP e memoriza os cookies na resposta do HTTP.

  ***Nota:** Mesmo quando `manageCookies` é `true`, o aplicativo pode adicionar um cookie manualmente a uma solicitação ao HTTP usando `URLRequest.requestHeaders`. Se esse cookie tiver o mesmo nomde de um cookie que o AIR for TV esteja gerenciando, a solicitação terá dois cookies com o mesmo nome. Os valores dos dois cookies podem ser diferentes.*

- Defina `manageCookies` como `false`. Esse valor significa que o aplicativo é responsável por enviar cookies automaticamente às solicitações ao HTTP e por memorizar os cookies na resposta do HTTP.

Para obter mais informações, consulte URLRequest.

Fluxo de trabalho para desenvolver um aplicativo do AIR for TV

Você pode desenvolver aplicativos AIR for TV com as seguintes ferramentas de desenvolvimento da plataforma Adobe Flash:

- Adobe Flash Professional

  O Adobe Flash Professional CS5.5 suporta o AIR 2.5 para TV, a primeira versão do AIR a suportar aplicativos do AIR for TV.

- Adobe Flash Builder®

  O Flash Builder 4.5 suporta o AIR 2.5 para TV.

- O AIR SDK

  A partir do AIR 2.5, é possível desenvolver os aplicativos usando as ferramentas da linha de comando fornecidas pelo AIR SDK. Para baixar o AIR SDK, consulte http://www.adobe.com/br/products/air/sdk/.

Utilização do Flash Professional

O uso do Flash Professional para desenvolver, testar e publicar aplicativos do AIR for TV é semelhante ao uso da ferramenta dos aplicativos AIR para aplicações de desktop.

No entanto, ao escrever o seu código ActionScript 3.0, use somente classes e métodos que sejam suportadas pelos perfis `tv` e `extendedTV` do AIR. Para obter detalhes, consulte “Perfis de dispositivo” na página 249.

Configurações do projeto

Faça o seguinte para configurar seu projeto para um aplicativo do AIR for TV:

- Na guia Flash da caixa de diálogo Configurações de Publicação, defina o valor Player em pelo menos AIR 2.5.
- Na guia Geral da caixa de diálogo Configurações do Adobe AIR (Configurações do Aplicativo e do Programa de Instalação), defina o perfil como `TV` ou TV `expandida`.

Depuração

Você pode executar seu aplicativo usando o AIR Debug Launcher dentro do Flash Professional. Faça o seguinte:

- Para executar o aplicativo no modo de depuração, selecione:

  Escolha Depurar > Depurar filme > No AIR Debug Launcher (Desktop)

  Depois de fazer essa seleção, nas operações de depuração seguintes você poderá selecionar:

  Selecione Depurar > Depurar filme > Depurar

- Para executar o aplicativo sem os recursos do modo de depuração, selecione:

  Selecione Controle > Testar filme > No AIR Debug Launcher (Desktop)

  Depois de fazer essas seleção, você poderá selecionar Controle > Testar Filme > Testar para as execuções seguintes.

Depois de definir o perfil do AIR como TV ou TV expandida, o AIR Debug Launcher apresentará um menu chamado Botões do Controle Remoto. Você pode usar este menu para simular a digitação de teclas em um dispositivo com controle remoto.

Para obter mais informações, consulte "Depuração remota com o Flash Professional" na página 148.

Utilização de extensões nativas

Se o seu aplicativo usa uma extensão nativa, siga as instruções em "Lista de tarefas que utilizam uma extensão nativa" na página 153.

No entanto, quando um aplicativo usa extensões nativas:

- Não é possível publicar o aplicativo usando o Flash Professional. Para publicar o aplicativo, use ADT. Consulte "Compactação com ADT" na página 145.
- Não é possível executar ou depurar o aplicativo usando o Flash Professional. Para depurar o aplicativo na máquina de desenvolvimento, use ADL. Consulte "Simulação de dispositivos utilizando ADL" na página 146.

Usando o Flash Builder

A partir do Flash Builder 4.5, o Flash Builder suporta o desenvolvimento do AIR for TV. O uso do Flash Builder para desenvolver, testar e publicar aplicativos AIR for TV é semelhante ao uso da ferramenta dos aplicativos AIR para computador pessoal.

Configurando o aplicativo

Certifique-se de que o seu aplicativo:

- Utilize o elemento `Application` como a classe container no arquivo MXML, se você estiver usando um arquivo 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: *Os aplicativos AIR for TV não suportam o elemento* `WindowedApplication`.

  Nota: *Você não precisa usar um arquivo MXML em nenhuma hipótese. Em vez disso, pode criar um projeto do ActionScript 3.0.*

- Utilize somente classes e métodos do ActionScript 3.0 que sejam suportados pelos perfis `tv` e `extendedTV` do AIR. Para obter detalhes, consulte "Perfis de dispositivo" na página 249.

Além disso, no arquivo XML do seu aplicativo, certifique-se de que:

- O atributo `xmlns` do elemento `application` esteja definido como AIR 2.5:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

- O elemento `supportedProfiles` inclua `tv` ou `extendedTV`:

```
<supportedProfiles>tv</supportedProfiles>
```

Depuração do aplicativo para

Você pode executar seu aplicativo usando o AIR Debug Launcher dentro do Flash Builder. Faça o seguinte:

1. Selecione Executar > Configurações de Depuração.
2. Certifique-se de que o campo Perfil esteja definido como Desktop (computador pessoal).
3. Selecione Executar > Depurar para executar o modo de depuração ou selecione Executar > Executar para executar sem os recursos do modo de depuração.

Uma vez que você definiu o elemento `supportedProfiles` como TV ou TV expandida, o AIR Debug Launcher apresenta um menu chamado Botões do Controle Remoto. Você pode usar este menu para simular a digitação de teclas em um dispositivo com controle remoto.

Para obter mais informações, consulte "Depuração remota com o Flash Builder" na página 149.

Utilização de extensões nativas

Se o seu aplicativo usa uma extensão nativa, siga as instruções em "Lista de tarefas que utilizam uma extensão nativa" na página 153.

No entanto, quando um aplicativo usa extensões nativas:

- Não é possível publicar o aplicativo usando o Flash Builder. Para publicar o aplicativo, use ADT. Consulte "Compactação com ADT" na página 145.
- Não é possível executar ou depurar o aplicativo usando o Flash Builder. Para depurar o aplicativo na máquina de desenvolvimento, use ADL. Consulte "Simulação de dispositivos utilizando ADL" na página 146.

Propriedades do descritor do aplicativo AIR for TV

Tal como acontece com outros aplicativos AIR, você define as propriedades básicas do aplicativo no arquivo descritor do aplicativo. Os aplicativos de perfil de TV ignoram algumas das propriedades específicas para ambiente de trabalho, como tamanho da janela e transparência. Os dispositivos de destino do aplicativo no perfil `extendedTV` podem usar extensões nativas. Esses aplicativos identificam as extensões nativas usadas em um elemento `extensions`.

Configurações comuns

Várias configurações de descritor de aplicativo são importantes para todos os aplicativos de perfil de TV.

Versão de runtime exigida pelo AIR

Especifique a versão do runtime do AIR exigido pelo seu aplicativo usando o namespace do arquivo do descritor do aplicativo.

O namespace, atribuído no elemento do `application`, determina, em grande parte, que recursos seu aplicativo pode usar. Por exemplo, considere um aplicativo que use o namespace AIR 2.5, mas o usuário tenha alguma versão posterior instalada. Nesse caso, o aplicativo ainda verá o comportamento do AIR 2.5, mesmo que o comportamento seja diferente na versão posterior do AIR. Somente quando mudar o namespace e publicar uma atualização seu aplicativo terá acesso aos novos comportamentos e recursos. As correções de segurança são uma importante exceção a essa regra.

Especifique o namespace usando o atributo`xmlns` da raiz do elemento `application`:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

O AIR 2.5 é a primeira versão do AIR disponível para aplicativos de TV.

Identidade do aplicativo

Diversas configurações devem ser exclusivas para cada aplicativo que você publicar. Essas configurações incluem os elementos `id`, `name` e `filename`.

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

Versão do aplicativo

Especifique a versão do aplicativo no elemento `versionNumber`. Ao especificar um valor para versionNumber, você pode usar uma sequência de até três números separados por pontos, como: "0.1.2". Cada segmento do número de versão pode ter até três dígitos. (Em outras palavras, "999.999.999" é o maior número de versão autorizada.) Você não precisa incluir todos os três segmentos do número; "1" e "1.0" são números de versão legal.

Você também pode especificar um rótulo para a versão com o elemento `versionLabel`. Quando você adiciona um rótulo da versão este é exibido em vez do número da versão.

```
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

SWF do aplicativo principal

Especifique o arquivo SWF do aplicativo principal `versionLabel` no filho do elemento `initialWindow`. Ao direcionar dispositivos no perfil da tevê, você deve usar um arquivo SWF (aplicativos com base HTML não estão disponíveis).

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

Você deve incluir o arquivo no pacote AIR (usando ADT ou sua IDE). Simplesmente fazer referência do nome no descritor do aplicativo não faz com que o arquivo seja incluído no pacote de maneira automática.

Propriedades da tela principal

Vários elementos filho do elemento `initialWindow controlam a aparência inicial e o comportamento da tela principal do aplicativo`. Enquanto a maioria dessas propriedades são ignoradas em dispositivos nos perfis de TV, você pode usar o elemento `fullScreen`:

- **`fullScreen`** — Especifica se o aplicativo deve tomar a tela completa do dispositivo, ou se deve compartilhar a tela com o cromo normal do sistema operacional.

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

O elemento visible

O elemento `visible` é um elemento filho do elemento `initialWindow`initialWindow O AIR for TV ignora o elemento `visible` porque o conteúdo de seu aplicativo está sempre visível nos dispositivos do AIR for TV.

No entanto, defina o elemento `visible` como `true` se seu aplicativo também for dirigido aos dispositivos de desktop.

Nos dispositivos de desktop, o valor do elemento adota o padrão `false`. Portanto, se você não incluir o elemento `visible`, o conteúdo do aplicativo não ficará visível nos dispositivos de desktop. Embora você possa usar a classe NativeWindow do ActionScript para tornar o conteúdo visível nos dispositivos de desktop, os perfis de aparelhos de TV não suportam a classe NativeWindow. Se você tentar usar a classe NativeWindow em um aplicativo a ser executado em um dispositivo do AIR for TV, ocorrerá uma falha no carregamento do aplicativo. Não importa se você chamar ou não um método da classe NativeWindow; um aplicativo que utilize a classe não é carregado em um dispositivo do AIR for TV.

Perfis disponíveis

Se o seu aplicativo só fizer sentido em um aparelho de televisão, então você poderá impedir sua instalação em outros tipos de dispositivos de computação. Exclua os outros perfis da lista suportada no elemento `supportedProfiles`:

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Se um aplicativo usar uma extensão nativa, inclua somente o perfil `extendedTV` na lista de perfis suportados:

```
<supportedProfiles>extendedTV</supportedProfiles>
```

Se você omitir o elemento `supportedProfiles` , pressupõe-se que o aplicativo seja compatível com todos os perfis.

Não inclua *somente* o perfil de `tv` na lista `supportedProfiles`. Alguns aparelhos de TV sempre executam o AIR for TV em um modo que corresponde ao perfil `extendedTV`. Esse comportamento ocorre para que o AIR for TV possa executar aplicativos que utilizam extensões nativas. Se seu elemento `supportedProfiles` especificar somente `tv`, ele está declarando que seu conteúdo é incompatível com o modo `extendedTV` do AIR for TV. Portanto, alguns dispositivos para TV não carregam um aplicativo que especifica somente o perfil `tv`.

Para uma lista de classes ActionScript disponíveis nos perfis `tv` e `extendedTV`, consulte "Capacidades de perfis diferentes" na página 250.

Extensões nativas necessárias

Os aplicativos que suportam o perfil `extendedTV` podem utilizar extensões nativas.

Declare todas as extensões nativas que o aplicativo AIR usas no descritor de aplicativo usando os elementos `extensions` e `extensionID`. O seguinte exemplo ilustra a sintaxe para especificar duas extensões nativas necessárias:

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

Se uma extensão não estiver listada, o aplicativo não poderá usá-la.

O elemento `extensionID` tem o mesmo valor que o elemento `id` no arquivo descritor de extensão. O arquivo descritor da extensão é um arquivo XML chamado extension.xml. Ele é empacotado no arquivo ANE que você recebe do fabricante do dispositivo.

Se você listar uma extensão no elemento `extensions`, mas o dispositivo do AIR for TV não tiver uma extensão instalada, o aplicativo não poderá ser executado. A exceção a essa regra é ocorre quando o arquivo ANE que você empacotou com o aplicativo do AIR for TV tem uma versão do componente da extensão. Em caso afirmativo, o aplicativo poderá ser executado e usará a versão do componente da extensão. A versão do componente tem o código do ActionScript, mas nenhum código nativo.

Ícones de aplicativos

Os requisitos relativos aos ícones do aplicativo nos aparelhos de televisão dependem de cada aparelho. Por exemplo, o fabricante do aparelho especifica:

- Ícones e tamanhos de ícones necessários.
- Tipos de arquivos e convenções de nomenclatura necessárias.
- Como fornecer ícones para o seu aplicativo, como se os ícones serão empacotados com o seu aplicativo.
- Se é possível especificar os ícones em um elemento `icon` no arquivo descritor do aplicativo.
- O comportamento, se o aplicativo não fornecer ícones.

Consulte o fabricante do aparelho para obter detalhes.

Configurações ignoradas

Aplicativos em aparelhos de televisão ignoram as configurações de aplicativo que se aplicam à janela nativa e móvel, ou aos recursos do sistema operacional de área de trabalho. As configurações ignoradas são:

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

Compactação de um aplicativo do AIR for TV

Compactação com ADT

Você pode usar a ferramenta de linha de comando ADT AIR para compactar um aplicativo do AIR for TV. A partir da versão 2.5 do AIR SDK, o ADT suporta o empacotamento para aparelhos de TV. Antes de empacotar, compile todos os seus códigos ActionScript e MXML. Você deve também ter um certificado de assinatura de código. Você pode criar um certificado usando o comando certificado ADT.

Para uma referência pormenorizada sobre as opções e os comandos ADT, consulte "AIR Developer Tool (ADT)" na página 168.

Criação de um pacote AIR

Para criar um pacote AIR, use o comando de pacote ADT:

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

Este exemplo considera que:

- O caminho da ferramenta ADT está na definição de caminho do shell da linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308.)
- O certificado codesign.p12 está no diretório pai do qual você executa o comando ADT.

Execute o comando a partir de um diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, e um diretório de ícones.

Quando você executa o comando, como demonstrado, a ADT solicitará a senha do armazenamento de chaves. Nem todos os programas de shell exibem os caracteres de senha que você digita; simplesmente pressione Enter quando estiver digitando. Alternativamente, você pode usar o parâmetro `storepass` para incluir a senha no comando ADT.

Criação de um pacote AIRN

Se o aplicativo do AIR for TV usar uma extensão nativa, crie um pacote AIRN em vez de um pacote AIR. Para criar um pacote AIRN, use o comando de pacote ADT, definindo o tipo de destino como `airn`.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml
myApp.swf icons -extdir C:\extensions
```

Este exemplo considera que:

- O caminho da ferramenta ADT está na definição de caminho do shell da linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308.)
- O certificado codesign.p12 está no diretório pai do qual você executa o comando ADT.
- O parâmetro `-extdir` denomina um diretório que contém os arquivos ANE que o aplicativo usa.

  Esses arquivos ANE contêm um fragmento somente para ou uma versão de simulador da extensão ActionScript. A versão da extensão que contém o código nativo está instalada no dispositivo AIR for TV.

Execute o comando a partir de um diretório que contém os arquivos do aplicativo. Os arquivos do aplicativo no exemplo são myApp-app.xml (o arquivo descritor do aplicativo), myApp.swf, e um diretório de ícones.

Quando você executa o comando, como demonstrado, a ADT solicitará a senha do armazenamento de chaves. Nem todos os programas de shell exibem os caracteres de senha que você digita; simplesmente pressione Enter quando estiver digitando. Alternativamente, você pode usar o parâmetro `storepass` para incluir a senha no comando.

Também é possível criar um arquivo AIRI para um aplicativo do AIR for TV que use extensões nativas. O arquivo AIRI é semelhante ao arquivo AIRN, exceto por não ser assinado. Por exemplo:

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Você pode, então, criar um arquivo AIRN a partir de um arquivo AIRI quando estiver pronto para assinar o aplicativo:

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Para obter mais informações, consulte Desenvolvimento de extensões nativas para Adobe AIR.

Compactação com o Flash Builder ou Professional Flash

O Flash Professional e o Flash Builder permitem publicar ou exportar os pacotes AIR se ter que executar por si próprio a ADT. O procedimento para criar um pacote AIR para um aplicativo do AIR é abordado na documentação para esses programas.

Atualmente, no entanto, somente a ADT pode criar pacotes do AIRN, os pacotes de aplicativos do AIR for TV que usam extensões nativas.

Para obter mais informações, consulte o seguinte:

- Empacotar aplicativos AIR com o Flash Builder
- Publicação no Adobe AIR usando Flash Professional

Depuração de aplicativos AIR for TV

Simulação de dispositivos utilizando ADL

A forma mais rápida e fácil de testar e depurar a maioria dos recursos de aplicativo é executar o aplicativo no computador de desenvolvimento usando o utilitário ADL (Adobe Debug Launcher).

O ADL usa o elemento `supportedProfiles` no descritor do aplicativo para escolher que perfil usar. Especificamente:

- Se mais de um perfil estiver listado, o ADL usa o primeiro na lista.
- Você também pode usar o parâmetro `-profile` da ADL para selecionar um dos outros perfis na lista `supportedProfiles`.
- (Se você não incluir um elemento supportedProfiles no descritor do aplicativo, qualquer perfil poderá ser especificado para o `argumento` -profile.)

Por exemplo, use o seguinte comando para iniciar um aplicativo para simular o perfil `tv`:

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

Ao simular o perfil `tv` ou `extendedTV` na área de trabalho com o ADL, o aplicativo é executado em um ambiente que mais se aproxima de um dispositivo de destino. Por exemplo:

- As APIs do ActionScript que não fazem parte do perfil no argumento `-profile` não estão disponíveis.
- A ADL permite a entrada de controles de entrada de dispositivo, tais como controles remotos, por meio dos comandos de menu.
- A especificação de `tv` ou extendedTV`no argumento` -profile permite que o ADL simule a classe StageVideo na área de trabalho.

- A especificação `extendedTV` no argumento `-profile` permite que o aplicativo use fragmentos ou simuladores de extensões nativas empacotados com o arquivo AIRN do aplicativo.

No entanto, como ADL executa o aplicativo no ambiente de trabalho, os testes dos aplicativos AIR for TV usando o ADL tem limitações:

- Não refletem o desempenho do aplicativo no dispositivo. Executam testes de desempenho no dispositivo de destino.
- Não simula as limitações do objeto StageVideo. Normalmente você usa a classe StageVideo e não a classe Video para reproduzir um vídeo ao direcionar dispositivos AIR for TV. A classe StageVideo aproveita os benefícios de desempenho de hardware do dispositivo, mas tem limitações de exibição. O ADL reproduz o vídeo na área de trabalho sem essas limitações. Por isso, teste a reprodução de vídeo no dispositivo de destino.
- Não se pode simular o código nativo de uma extensão nativa. Você pode, no entanto, especificar o perfil `extendedTV`, que suporta extensões nativas, no argumento `-profile` da ADL. A ADL permite realizar testes com a versão do fragmento ou simulador somente para da extensão ActionScript incluída no pacote ANE. No entanto, de modo geral a extensão correspondente, que é instalada no dispositivo, também inclui código nativo. Para realizar testes usando a extensão com seu código nativo, execute o aplicativo no dispositivo de destino.

Para obter mais informações, consulte "AIR Debug Launcher (ADL)" na página 162.

Utilização de extensões nativas

Se o seu aplicativo usar extensões nativas, o comando ADL terá a aparência semelhante ao seguinte exemplo:

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

Este exemplo considera que:

- O caminho da ferramenta ADL está na definição de caminho do shell da linha de comando. (Consulte "Variáveis de ambiente do caminho" na página 308.)
- O diretório atual contém os arquivos do aplicativo. Esses arquivos incluem os arquivos SWF e o arquivo descritor do aplicativo, que é myApp-app.xml neste exemplo.
- O parâmetro `-extdir` nomeia um diretório que contém um diretório para cada extensão nativa que o aplicativo utiliza. Cada um desses diretórios contém o arquivo ANE *desempacotado* da extensão nativa. Por exemplo:

```
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
```

Esses arquivos ANE desempacotados contêm somente um fragmento ou uma versão de simulador da extensão ActionScript. A versão da extensão que contém o código nativo está instalada no dispositivo AIR for TV.

Para obter mais informações, consulte Desenvolvimento de extensões nativas para Adobe AIR.

Entrada de controle

O ADL simula as teclas do controle remoto de um aparelho de TV. Você pode enviar as entradas destes botões para o dispositivo simulado usando o menu exibido quando o ADL é ativado através de um dos perfis de TV.

Tamanho da tela

Você pode testar seu aplicativo em telas de tamanho diferente, definindo o parâmetro -screensize do ADL. Você pode especificar uma string contendo os quatro valores que representam as larguras e alturas de telas normais e maximizadas.

Sempre especifique as dimensões de pixel para o layout retrato, o que significa especificar a largura como um valor menor do que o valor da altura. Por exemplo:

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

Instruções de rastreio

Ao executar o aplicativo de TV na área de trabalho, a saída de rastreio é impressa para o depurador ou para a janela de terminal usada para ativar o ADL.

Depuração remota com o Flash Professional

Você pode usar o Flash Professional para depurar remotamente o seu aplicativo do AIR for TV enquanto ele é executado em um dispositivo de destino. No entanto, as etapas para configurar a depuração remota dependem do dispositivo. Por exemplo, o Kit de Desenvolvimento de Hardware Adobe® AIR® para TV MAX 2010 contém a documentação para as etapas detalhadas desse dispositivo.

Independentemente do dispositivo de destino, execute as seguintes etapas para preparar a depuração remota:

1 Na caixa de diálogo Configurações de Publicação, na guia Flash, selecione Permitir Depuração.

   Esta opção faz com que o Flash Professional inclua informações de depuração em todos os arquivos SWF que cria a partir do seu arquivo FLA.

2 Na guia Assinatura da caixa de diálogo Configurações do Adobe AIR (Configurações do Aplicativo e do Programa de Instalação), selecione a opção para preparar um arquivo intermediário do AIR (AIRI - AIR Intermediate).

   Quando você ainda estiver desenvolvendo seu aplicativo, a utilização de um arquivo AIRI, que não requer nenhuma assinatura, será suficiente.

3 Publique o seu aplicativo criando o arquivo AIRI.

As últimas etapas são instalar e executar o aplicativo no dispositivo de destino. No entanto, essas etapas dependem do dispositivo.

Depuração remota com o Flash Builder

Você pode usar o Flash Builder para depurar remotamente o seu aplicativo do AIR for TV enquanto ele é executado no dispositivo de destino. No entanto, as etapas para configurar a depuração remota dependem do dispositivo.

Independentemente do dispositivo de destino, execute as seguintes etapas para preparar a depuração remota:

1 Selecione Projeto > Exportar versão da compilação. Selecione a opção para preparar um arquivo intermediário do AIR (AIRI - AIR Intermediate).

   Quando você ainda estiver desenvolvendo seu aplicativo, a utilização de um arquivo AIRI, que não requer nenhuma assinatura, será suficiente.

2 Publique o seu aplicativo criando o arquivo AIRI.

3 Altere o pacote AIRI do aplicativo para que contenha arquivos SWF que contenham informações de depuração.

   Os arquivos SWF que contém informações de depuração do aplicativo em um diretório chamado bin-debug estão localizados no diretório de projeto do Flash Builder. Substitua os arquivos SWF do pacote AIRI por arquivos SWF do diretório bin-debug.

Em uma máquina de desenvolvimento em Windows, você pode fazer essa substituição da seguinte maneira:

1 Renomeie o pacote AIRI para que tenha a extensão de nome de arquivo .zip, em vez de .airi.

2 Extraia o conteúdo do arquivo ZIP.

3 Substitua os arquivos SWF que estão na estrutura do diretório de extração pelos arquivos da depuração de bin.

4 Compacte os arquivos novamente no diretório de extração.

5 Altere o arquivo compactado para que tenha novamente a extensão de nome de arquivo .airi.

Se você estiver usando uma máquina de desenvolvimento em Mac, as etapas dessa substituição dependerão do dispositivo No entanto, elas geralmente envolvem o seguinte:

1 Instalar o pacote AIRI no dispositivo de destino.

2 Substituir os arquivos SWF do diretório de instalação do pacote no dispositivo de destino por arquivos SWF do diretório bin-debug.

Por exemplo, considere o dispositivo incluído com o Kit de Desenvolvimento de Hardware MAX 2010 do Adobe AIR for TV. Instale o pacote AIRI como descrito na documentação do kit. Em seguida, faça telnet na linha de comando de sua máquina de desenvolvimento em Mac para acessar o dispositivo de destino. Substitua os arquivos SWF do diretório de instalação do aplicativo em /opt/adobe/stagecraft/apps/<*nome do aplicativo*>/ por arquivos SWF do diretório bin-debug.

As etapas a seguir são relativas à depuração remota com o Flash Builder e o dispositivo incluído com o Kit de Desenvolvimento de Hardware MAX 2010 do Adobe AIR for TV.

1. No computador que executa o Flash Builder, o seu computador de desenvolvimento, execute o Conector de Dispositivos do AIR for TV que vem com o Kit de Desenvolvimento de Hardware MAX 2010. Ele mostra o endereço IP do seu computador de desenvolvimento.
2. No dispositivo do kit de hardware, ative o aplicativo DevMaster, que também é fornecido com o kit de desenvolvimento.
3. No aplicativo DevMaster, insira o endereço IP do seu computador de desenvolvimento como demonstrado no Conector de Dispositivos do AIR for TV.
4. No aplicativo DevMaster, certifique-se de que a opção Habilitar Depuração Remota esteja selecionada.
5. Encerre o aplicativo DevMaster.
6. No computador de desenvolvimento, selecione Iniciar no Conector do AIR for TV.
7. No dispositivo de kit de hardware, inicie outro aplicativo. Verifique se a informação de rastreio é exibida no Conector do AIR for TV.

   Se a informação de rastreio não for exibida, o computador de desenvolvimento e o dispositivo do kitg de hardware não estarão conectados. Certifique-se de que a porta do computador de desenvolvimento que é usada para a informação de rastreio esteja disponível. Você pode escolher uma porta diferente no Conector do AIR for TV. Certifique-se também de que o seu firewall permita o acesso à porta escolhida.

Em seguida, inicie o depurador no Flash Builder. Faça o seguinte:

1. No Flash Builder, selecione Executar > Configurações de Depuração.
2. Na configuração de depuração existente, que se destina à depuração local, copie o nome do projeto.
3. Na caixa de diálogo Configurações de Depuração, selecione Aplicativo da Web. Em seguida, selecione o ícone Configuração do Novo Lançamento.
4. Cole o nome do projeto no campo Projeto.
5. Na seção URL Ou Caminho Para o Lançamento, remova a marca de Usar Padrão. Insira também **`about:blank`** no campo de texto.
6. Selecione Aplicar para salvar as alterações.
7. Selecione Depurar para iniciar o depurador do Flash Builder.
8. Abra o seu aplicativo no dispositivo do kit de hardware.

Agora você pode usar o depurador do Flash Builder para, por exemplo, definir pontos de quebra e examinar variáveis.

Capítulo 9: Como utilizar extensões nativas para Adobe AIR

As extensões nativas para Adobe AIR fornecem APIs do ActionScript que permitem o acesso à funcionalidade específica do dispositivo programada em código nativo. Os desenvolvedores de extensões nativas trabalham com os fabricantes de dispositivos e às vezes são desenvolvedores terceirizados.

Caso você esteja desenvolvendo uma extensão nativa, consulte Developing Native Extensions for Adobe AIR.

Uma extensão nativa é uma combinação de:

- Classes ActionScript.
- Código nativo.

No entanto, como um desenvolvedor de aplicativos do AIR que usa uma extensão nativa, você trabalha somente com classes ActionScript.

As extensões nativas são úteis nas seguintes situações:

- A implementação do código nativo fornece o acesso aos recursos específicos da plataforma. Esses recursos específicos da plataforma não estão disponíveis nas classes ActionScript embutidas, e não é possível os implementar nas classes ActionScript específicas do aplicativo. A implementação do código nativo pode fornecer essa funcionalidade porque tem acesso ao hardware e ao software específicos do dispositivo.
- Uma implementação do código nativo às vezes pode ser mais rápida do que uma implementação que utilize somente o ActionScript.
- A implementação do código nativo pode fornecer ao ActionScript o acesso ao código nativo legado.

Alguns exemplos de extensões nativas estão no Centro do Desenvolvedor da Adobe. Por exemplo, uma extensão nativa fornece aos aplicativos AIR o acesso ao recurso de vibração do Android. Consulte Native extensions for Adobe AIR.

Arquivos de Extensão Nativa do AIR (ANE)

Os desenvolvedores de extensões nativas fazem a compactação de uma extensão nativa para um arquivo ANE. Um arquivo ANE é um arquivo de arquivo morto que contém as bibliotecas e recursos necessários para a extensão nativa.

Observe que, em alguns dispositivos, o arquivo ANE contém a biblioteca do código nativo que a extensão nativa utiliza. Mas para outros dispositivos a biblioteca do código nativo é instalada no dispositivo. Em alguns casos, a extensão nativa não tem nenhum código nativo para um dispositivo em particular; ele é implementado somente com o ActionScript.

Como um desenvolvedor de aplicativos do AIR, você utiliza o arquivo ANE da seguinte maneira:

- Inclua o arquivo ANE no caminho da biblioteca do aplicativo da mesma maneira que você incluiu um arquivo SWC no caminho da biblioteca. Esse procedimento permite que o aplicativo busque referências das extensões das classes ActionScript.

  Nota: *Ao compilar seu aplicativo certifique-se de usar um vínculo dinâmico para o ANE. Se você usa o Flash Builder, especifique Externo no painel Propriedades do caminho no ActionScript Builder. Caso use a linha de comando, especifique -external-library-path.*

- Empacote o arquivo ANE com o aplicativo do AIR.

Extensões nativas em contraste com a classe ActionScript NativeProcess

O ActionScript 3.0 fornece uma classe NativeProcess. Essa classe permite que o aplicativo do AIR execute processos nativos no sistema operacional host. Essa capacidade é similar a extensões nativas, que fornecem acesso a recursos e bibliotecas específicas de plataforma. Ao decidir utilizar a classe NativeProcess em vez de utilizar uma extensão nativa, considere o seguinte:

- Somente o perfil `extendedDesktop` do AIR suporta a classe NativeProcess. Portanto, para aplicativos com os perfis `mobileDevice` e `extendedMobileDevice` do AIR, as extensões nativas são a única escolha.
- Os desenvolvedores de extensões nativas geralmente fornecem implementações nativas para várias plataformas, mas a API do ActionScript que eles fornecem são geralmente as mesmas plataformas cruzadas. Ao usar a classe NativeProcess, o código do ActionScript para iniciar o processo nativo pode variar entre as diferentes plataformas.
- A classe NativeProcess inicia um processo separado, enquanto que a extensão nativa é executada no mesmo processo do aplicativo do AIR. Portanto, se você está preocupado com o bloqueio de códigos, o uso da classe NativeProcess é mais seguro. No entanto, o processo separado significa que provavelmente você tem uma manipulação de comunicação de interprocessos para implementar.

Extensões nativas em contraste com as bibliotecas da classe ActionScript (arquivos SWC)

Um arquivo SWC file é uma biblioteca da classe ActionScript em um formato de arquivo morto. O arquivo SWC contém um arquivo SWF e outros arquivos de recurso. O arquivo SWC é uma maneira conveniente de compartilhar classes ActionScript em vez de compartilhar arquivos ActionScript de recurso e de código individuais.

Um pacote de extensão nativa é um arquivo ANE. Assim como um arquivo SWC, um arquivo ANE também é uma biblioteca da classe ActionScript, que contém um arquivo SWF e outros arquivos de recurso em formato de arquivo morto. No entanto, a diferença mais importante entre um arquivo ANE file e um arquivo SWC é que somente um arquivo ANE pode obter uma biblioteca de código nativa.

Nota: *Ao compilar seu aplicativo certifique-se de usar um vínculo dinâmico para o arquivo ANE. Se você usa o Flash Builder, especifique Externo no painel Propriedades do caminho no ActionScript Builder. Caso use a linha de comando, especifique -external-library-path.*

Mais tópicos da Ajuda

Sobre arquivos SWC

Dispositivos suportados

A partir do AIR 3, é possível utilizar extensões nativas em aplicativos para os seguintes dispositivos:

- Dispositivos do Android, a partir do Android 2.2

- Dispositivos do iOS, a partir do iOS 4.0
- Simulador iOS, a partir do AIR 3.3
- Blackberry PlayBook
- Dispositivos pessoais do Windows que suportam o AIR 3.0
- Dispositivos pessoais do Mac OS X que suportam o AIR 3.0

Geralmente, a mesma extensão nativa é direcionada para múltiplas plataformas. O arquivo ANE da extensão contém bibliotecas do ActionScript e nativas para cada plataforma suportada. Geralmente, as bibliotecas do ActionScript possuem as mesmas interfaces públicas para todas as plataformas. As bibliotecas nativas são basicamente diferentes.

Às vezes, uma extensão nativa é compatível com uma plataforma padrão. A implementação da plataforma padrão tem somente o código do ActionScript, mas nenhum código nativo. Se você empacotar um aplicativo para uma plataforma que a extensão não suporte especificamente, o aplicativo utilizará a implementação padrão ao ser executado. Por exemplo, considere uma extensão que fornece um recurso que se aplica somente a dispositivos móveis. A extensão também poderá fornecer uma implementação padrão que um aplicativo de desktop poderá utilizar para simular o recurso.

Perfis de dispositivo suportados

Os seguintes perfis do AIR suportam extensões nativas:

- `extendedDesktop`, a partir do AIR 3.0
- `mobileDevice`, a partir do AIR 3.0
- `extendedMobileDevice`, a partir do AIR 3.0

Mais tópicos da Ajuda

Suporte ao perfil do AIR

Lista de tarefas que utilizam uma extensão nativa

Para usar uma extensão nativa no aplicativo, realize as seguintes tarefas:

1. Indique a extensão no arquivo de indexação do aplicativo.
2. Inclua o arquivo ANE no caminho da biblioteca do aplicativo.
3. Empacotar o aplicativo.

Indicação da extensão no arquivo de indexação do aplicativo

Todos os aplicativos do AIR possuem um arquivo de indexação do aplicativo Quando um aplicativo utiliza uma extensão nativa, o arquivo de indexação do aplicativo inclui um elemento `<extensions>`. Por exemplo:

```
<extensions>
    <extensionID>com.example.Extension1</extensionID>
    <extensionID>com.example.Extension2</extensionID>
</extensions>
```

O elemento `extensionID` tem o mesmo valor que o elemento `id` no arquivo descritor de extensão. O arquivo descritor da extensão é um arquivo XML chamado extension.xml. Ele é empacotado em um arquivo ANE. É possível utilizar uma ferramenta de extração de arquivo morto para visualizar o arquivo extension.xml.

Inclusão do arquivo ANE no caminho da biblioteca do aplicativo

Para compilar um aplicativo que utilize uma extensão nativa, inclua o arquivo ANE no caminho da biblioteca.

Uso do arquivo ANE com o Flash Builder

Se o aplicativo utilizar uma extensão nativa, inclua o arquivo ANE para a extensão nativa no caminho da biblioteca. Em seguida, você pode usar o Flash Builder para compilar o seu código ActionScript.

Execute as seguintes etapas, que usam o Flash Builder 4.5.1:

1. Altere a extensão do nome do arquivo ANE de .ane para .swc. Essa etapa é necessária para que o Flash Builder possa localizar o arquivo.
2. Selecione Projeto > Propriedades no seu projeto do Flash Builder.
3. Selecione Caminho de Criação do Flex na caixa de diálogo Propriedades.
4. Na guia Caminho da Biblioteca, selecione Adicionar SWC....
5. Navegue para o arquivo SWC e selecione Abrir.
6. Selecione OK na caixa de diálogo Adicionar SWC...

   O arquivo ANE agora aparece na guia Caminho da Biblioteca, na caixa de diálogo Propriedades.
7. Expanda a entrada do arquivo SWC. Clique duas vezes em Tipo de Link para abrir a caixa de diálogo Opções de Itens de Caminho da Biblioteca.
8. Na caixa de diálogo Opções do Item do Caminho da Biblioteca, altere o tipo de link para Externo.

Agora você pode compilar o seu aplicativo usando, por exemplo, Projeto > Criar Projeto.

Uso do arquivo ANE com o Flash Professional

Se o aplicativo utilizar uma extensão nativa, inclua o arquivo ANE para a extensão nativa no caminho da biblioteca. Em seguida, você poderá usar o Flash Professional CS5.5 para compilar o seu código ActionScript. Faça o seguinte:

1. Altere a extensão do nome do arquivo ANE de .ane para .swc. Essa etapa é necessária para que o Flash Professional possa localizar o arquivo.
2. Selecione Arquivo > Configurações do ActionScript no seu arquivo FLA.
3. Selecione a guia Caminho da Biblioteca na caixa de diálogo Configurações Avançadas do ActionScript 3.0.
4. Selecione o botão Procurar arquivo SWC.
5. Navegue para o arquivo SWC e selecione Abrir.

O arquivo SWC agora aparece na guia Caminho da Biblioteca na caixa de diálogo Configurações Avançadas do ActionScript 3.0.

6 Com o arquivo SWC selecionado, selecione o botão Selecionar Opções de Ligação com uma Biblioteca.

7 Na caixa de diálogo Opções do Item do Caminho da Biblioteca, altere o tipo de link para Externo.

Compactando um aplicativo que utiliza extensões nativas

Use a ADT para empacotar um aplicativo que utilize extensões nativas. Não é possível empacotar o aplicativo usando o Flash Professional CS5.5 ou o Flash Builder 4.5.1.

Detalhes sobre o uso de ADT podem ser encontrados em Ferramenta do Desenvolvedor do AIR (ADT).

Por exemplo, o seguinte comando de ADT cria um arquivo DMG (um arquivo de instalador nativo para o Mac OS X) para um aplicativo que utiliza extensões nativas:

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

O seguinte comando cria um pacote APK para um dispositivo do Android:

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

O seguinte comando cria um pacote iOS para um aplicativo 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
```

Observe o seguinte:

- Utilize um tipo de pacote de instalador nativo.
- Especifique o diretório da extensão.
- Certifique-se de que o arquivo ANE suporte o dispositivo de destino do aplicativo.

Utilize um tipo de pacote de instalador nativo.

O pacote do aplicativo deverá ser um instalador nativo. Não é possível criar um pacote do AIR de plataformas cruzadas (um pacote .air) para um aplicativo que utilize uma extensão nativa, já que as extensões nativas geralmente contêm o código nativo. No entanto, uma extensão nativa geralmente suporta múltiplas plataformas nativas com as mesmas APIs do ActionScript. Nesses casos, é possível utilizar o mesmo arquivo ANE em diferentes pacotes do instalador nativo.

A seguinte tabela resume o valor de uso para a opção `-target` do comando ADT:

Plataforma de destino do aplicativo	**-target**
Dispositivos pessoais do Mac OS X ou Windows	-target nativo -target conjunto
Android	-target apk ou outros destinos do pacote do Android.
iOS	-target ipa-ad-hoc ou outros destinos do pacote do iOS
Simulador iOS	-target ipa-test-interpreter-simulator -target ipa-debug-interpreter-simulator

Especifique o diretório de extensão

Utilize a opção do ADT `-extdir` para reportar ao ADT o diretório que contém as extensões nativas (arquivos ANE).

Para obter detalhes sobre esta opção, consulte "Opções de caminho e arquivo" na página 185.

Certifique-se de que o arquivo ANE suporte o dispositivo de destino do aplicativo

Ao fornecer um arquivo ANE, o desenvolvedor da extensão nativa informa quais plataformas a extensão suporta. Também é possível usar uma ferramenta de extração de arquivo morto para visualizar os conteúdos do arquivo ANE. Os arquivos extraídos incluem um diretório para cada plataforma suportada.

Saber quais plataformas a extensão suporta é importante ao compactar o aplicativo que utiliza o arquivo ANE. Considere as seguintes regras:

- Para criar um pacote de aplicativo do Android, o arquivo ANE deverá incluir a plataforma `Android-ARM`. Alternativamente, o arquivo ANE deverá incluir a plataforma padrão e pelo menos uma outra plataforma.
- Para criar um pacote de aplicativo do iOS, o arquivo ANE deverá incluir a plataforma `iPhone-ARM`. Alternativamente, o arquivo ANE deverá incluir a plataforma padrão e pelo menos uma outra plataforma.
- Para criar um pacote de aplicativo do simulador iOS, o arquivo ANE deverá incluir a plataforma `iPhone-x86`.
- Para criar um pacote de aplicativo do Mac OS X, o arquivo ANE deverá incluir a plataforma `MacOS-x86` . Alternativamente, o arquivo ANE deverá incluir a plataforma padrão e pelo menos uma outra plataforma.
- Para criar um pacote de aplicativo do Windows, o arquivo ANE deverá incluir a plataforma `Windows-x86`. Alternativamente, o arquivo ANE deverá incluir a plataforma padrão e pelo menos uma outra plataforma.

Capítulo 10: Compiladores ActionScript

Antes que o código MXML e ActionScript possa ser incluído em um aplicativo do AIR, ele deve ser compilado. Se você usar um IDE (ambiente de desenvolvimento integrado), como Adobe Flash Builder ou Adobe Flash Professional, o IDE manipula a compilação nos bastidores. No entanto, você também pode invocar os compiladores ActionScript na linha de comando para criar arquivos SWF quando não estiver usando um IDE ou quando usar um script de construção.

Sobre as ferramentas de linha de comando do AIR no Flex SDK

Cada uma das ferramentas de linha de comando que você usar para criar um aplicativo Adobe AIR chama a ferramenta correspondente usada para criar aplicativos

- amxmlc chama mxmlc para compilar classes de aplicativos
- acompc chama compc para compilar bibliotecas e classes de componentes
- aasdoc chama asdoc para gerar arquivos de documentação a partir dos comentários do código-fonte

A única diferença entre as versões Flex e AIR os utilitários é que as versões AIR carregam as opções de configuração do arquivo air-config.xml, em vez do arquivo flex-config.xml.

As ferramentas Flex SDK e as suas opções de linha de comando são descritas em detalhes na Documentação do Flex. As ferramentas Flex SDK são descritas aqui em nível básico para ajudá-lo a iniciar e para apontar as diferenças entre criar aplicativos Flex e aplicativos AIR.

Mais tópicos da Ajuda

"Criando seu primeiro aplicativo desktop do AIR com o Flex SDK" na página 35

Configuração do compilador

Geralmente você deve especificar as opções de compilação tanto na linha de comando quanto com um ou mais arquivos de configuração. O arquivo de configuração GlobalFlex SDK contém valores padrão que são usados sempre que os compiladores estão executando. Você pode editar este arquivo para que se adapte ao seu ambiente de desenvolvimento. Existem dois arquivos de configuração global do Flex, situados no diretório da estrutura da sua instalação do Flex SDK. O arquivo air-config.xml é usado ao executar o compilador amxmlc. Este arquivo configura o compilador do AIR incluindo bibliotecas do AIR. O arquivo flex-config.xml é usado ao executar mxmlc.

Os valores de configuração padrão são adequados para descobrir como o Flex e o AIR funcionam, mas quando você embarcar em um projeto de larga escala examine as opções disponíveis mais atentamente. Você pode fornecer valores específicos ao projeto para as opções do compilador m um arquivo de configuração local que tem precedência sobre os valores globais para um determinado projeto.

Nota: *Nenhuma opção de compilação é usada para aplicativos AIR, mas você deve consultar as biblioteca do AIR ao compilar um aplicativo do AIR. Geralmente, estas bibliotecas referem-se a um arquivo de configuração em nível de projeto, em um arquivo para uma ferramenta de criação como Ant ou diretamente na linha de comando.*

Compilação de arquivos de origem do MXML e ActionScript para AIR

Você pode compilar os ativos do Adobe® ActionScript® 3.0 e MXML do aplicativo do AIR com o compilador MXML de linha de comando (amxmlc). (Você não precisa compilar os aplicativos baseados em HTML. Para compilar um SWF no Flash Professional, basta publicar o filme em um arquivo SWF.)

O padrão básico de linha de comando para usar amxmlc é:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

em que *[compiler options]* especifica as opções de linha de comando usadas para compilar o aplicativo do AIR.

O comando amxmlc invoca o compilador mxmlc padrão do Flex com um parâmetro adicional, `+configname=air`. Esse parâmetro instrui o compilador a usar o arquivo air-config.xml em vez do arquivo flex-config.xml. Usar amxmlc é, de outro modo, idêntico a usar mxmlc.

O compilador carrega o arquivo de configuração air-config.xml especificando as bibliotecas do AIR e do Flex normalmente necessárias para compilar um aplicativo do AIR. Você também pode usar um arquivo de configuração de nível de projeto local para substituir ou adicionar opções adicionais à configuração global. Tipicamente, a maneira mais fácil de criar um arquivo de configuração local é editar uma cópia da versão global. Você pode carregar o arquivo local com a opção `-load-config`:

-load-config=project-config.xml Substitui opções globais.

-load-config+=project-config.xml Adiciona valores adicionais àquelas opções globais que levam mais de um valor, como a opção `-library-path`. Opções globais que levam apenas um único valor são substituídas.

Se você usar uma convenção de nomenclatura especial para o arquivo de configuração local, o compilador amxmlc carrega o arquivo local automaticamente. Por exemplo, se o arquivo MXML principal é `RunningMan.mxml`, nomeie o arquivo de configuração local: `RunningMan-config.xml`. Agora, para compilar o aplicativo, você apenas precisa digitar:

```
amxmlc RunningMan.mxml
```

`RunningMan-config.xml` é carregado automaticamente, uma vez que seu nome de arquivo corresponde àquele do arquivo MXML compilado.

exemplos de amxmlc

Os seguintes exemplos demonstram o uso do compilador amxmlc. (Apenas os ativos do ActionScript e MXML do seu aplicativo devem ser compilados.)

Compilar um arquivo MXML do AIR:

```
amxmlc myApp.mxml
```

Compilar e definir o nome de saída:

```
amxmlc –output anApp.swf -- myApp.mxml
```

Compilar um arquivo ActionScript do AIR:

```
amxmlc myApp.as
```

Especificar um arquivo de configuração do compilador:

```
amxmlc –load-config config.xml -- myApp.mxml
```

Adicionar opções adicionais de outro arquivo de configuração:

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

Adicionar bibliotecas na linha de comando (além das bibliotecas já no arquivo de configuração):

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

Compilar um arquivo MXML do AIR sem usar um arquivo de configuração (Win):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, ^
[AIR SDK]/frameworks/libs/air/airframework.swc, ^
-library-path [Flex SDK]/frameworks/libs/framework.swc ^
-- myApp.mxml
```

Compilar um arquivo MXLM do AIR sem usar um arquivo de configuração (Mac OS X ou Linux):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, \
[AIR SDK]/frameworks/libs/air/airframework.swc, \
-library-path [Flex 3 SDK]/frameworks/libs/framework.swc \
-- myApp.mxml
```

Compilar um arquivo MXML do AIR para usar uma biblioteca compartilhada de runtime:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --
myApp.mxml
```

Copie um arquivo AIR MXML para usar um ANE, certifique-se de usar `-external-library-path` para o ANE:

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

Compilação de Java (com o caminho de classe definido para incluir `mxmlc.jar`):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional
compiler options] -- myApp.mxml
```

A opção flexlib identifica a localização do seu diretório de estruturas do SDK do Flex, permitindo que o compilador localize o arquivo flex_config.xml.

Compilação de Java (sem o caminho de classe definido):

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air
[additional compiler options] -- myApp.mxml
```

Para invocar o compilador usando o Apach Ant (o exemplo usa uma tarefa Java para executar mxmlc.jar):

```
<property name="SDK_HOME" value="C:/Flex46SDK"/>
<property name="MAIN_SOURCE_FILE" value="src/myApp.mxml"/>
<property name="DEBUG" value="true"/>
<target name="compile">
    <java jar="${MXMLC.JAR}" fork="true" failonerror="true">
        <arg value="-debug=${DEBUG}"/>
        <arg value="+flexlib=${SDK_HOME}/frameworks"/>
        <arg value="+configname=air"/>
        <arg value="-file-specs=${MAIN_SOURCE_FILE}"/>
    </java>
</target>
```

Compilação de um componente ou de uma biblioteca de códigos do AIR (Flex)

Use o compilador de componentes, acompc, para compilar bibliotecas do AIR e componentes independentes. O compilador de componentes acompc se comporta como o compilador amxmlc, com as seguintes exceções:

- Você deve especificar que classes dentro da base do código para incluir na biblioteca ou no componente.
- O acompc não procura um arquivo de configuração local automaticamente. Para usar um arquivo de configuração de projeto, você deve usar a opção –load-config.

O comando acompc invoca o compilador de componentes compc padrão do Flex, mas carrega suas opções de configuração do arquivo `air-config.xml` em vez do arquivo `flex-config.xml`.

Arquivo de configuração do compilador de componentes

Use um arquivo de configuração local para evitar digitar (e talvez digitar incorretamente) o caminho de origem e os nomes de classes na linha de comando. Adicione a opção -load-config à linha de comando do acompc para carregar o arquivo de configuração local.

O exemplo a seguir ilustra uma configuração para criar uma biblioteca com duas classes, ParticleManager e Particle, ambas no pacote: `com.adobe.samples.particles`. Os arquivos de classe são localizados na pasta `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 a biblioteca usando o arquivo de configuração, chamado `ParticleLib-config.xml`, digite:

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Para executar o mesmo comando inteiramente na linha de comando, digite:

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Digite o comando inteiro em uma linha ou use o caractere de continuação de linha para seu shell de comando.)

Exemplos de acompc

Esses exemplos supõem que você está usando um arquivo de configuração chamado `myLib-config.xml`.

Compilar um componente ou uma biblioteca do AIR:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Compilar uma biblioteca compartilhada de runtime:

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Observe, a pasta lib deve existir e estar vazia antes de executar o comando.)

Capítulo 11: AIR Debug Launcher (ADL)

Use o ADL (AIR Debug Launcher) para executar aplicativos com base em SWF e HTML durante o desenvolvimento. Usando o ADL, você pode executar um aplicativo sem compactá-lo e instalá-lo primeiro. Por padrão, o ADL usa um runtime incluído com o SDK, o que significa que você não precisa instalar o runtime separadamente para usar o ADL.

O ADL imprime instruções de rastreamento e erros de runtime para a saída padrão, mas não suporta pontos de interrupção ou outros recursos de depuração. Você pode usar o Flash Debugger (ou um ambiente de desenvolvimento integrado como o Flash Builder) para os problemas complexos de depuração.

Nota: *Se suas declarações* `trace()` *não forem exibidas no console, verifique se você não especificou* `ErrorReportingEnable` *ou* `TraceOutputFileEnable` *no arquivo mm.cfg. Para obter mais informações sobre o local específico de plataforma desse arquivo, consulte Edição do arquivo mm.cfg.*

O AIR suporta depuração direta, e assim você não precisa da versão de depuração do runtime (como precisaria com o Adobe® Flash® Player). Para proceder à depuração na linha de comando, você pode usar o Flash Debugger e o AIR Debug Launcher (ADL).

O Flash Debugger é distribuído no diretório Flex SDK. As versões nativas, tais como fdb.exe no Windows, estão no subdiretório bin. A versão do Java está no subdiretório lib. O AIR Debug Launcher, adl.exe, está no diretório bin da sua instalação do SDK. (Não existe uma versão separada do Java).

Nota: *Você não pode iniciar um aplicativo do AIR diretamente com fdb, porque fdb tenta lançá-lo com o Flash Player. Em vez disso, deixe o aplicativo do AIR se conectar a uma sessão fdb em execução.*

Uso do ADL

Para executar um aplicativo com o ADL, use o seguinte padrão:

```
adl application.xml
```

Em que *application.xml* é o arquivo do descritor do aplicativo para o aplicativo.

A sintaxe completa para o ADL é:

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(Itens entre colchetes, [], são opcionais.)

-runtime *runtime-directory* Especifica o diretório que contém o runtime a ser usado. Se não especificado, o diretório do runtime no mesmo SDK do programa ADL é usado. Se você mover o ADL para fora da pasta SDK, especifique o diretório de runtime. No Windows e no Linux, especifique o diretório que contém o diretório do `Adobe AIR` . No Mac OS X, especifique o diretório que contém a `estrutura do Adobe AIR`.

***-pubid publisher-id* Atribui o valor especificado como o ID do editor do aplicativo do AIR para essa execução.** Especificar um ID de editor temporária permite que você teste recursos de um aplicativo do AIR, como comunicação por uma conexão local, que usa o ID do editor para ajudar a identificar unicamente um aplicativo. A partir do AIR 1.5.3, você também pode especificar o ID do editor no arquivo descritor do aplicativo (e não deve utilizar este parâmetro).

***Nota:** A partir do AIR 1.5.3, o ID do editor não é mais automaticamente calculado e atribuído a um aplicativo do AIR. Você pode especificar um ID de editor durante a criação de uma atualização para um aplicativo do AIR, mas aplicativos novos não necessitam disso e você não deve especificar um ID de editor.*

-nodebug Desativa o suporte para depuração. Se usado, o processo de aplicativo não poderá se conectar ao depurador do Flash e as caixas de diálogo para exceções não manipuladas são suprimidas. (No entanto, instruções de rastreamento ainda serão impressas na janela do console.) Desativar a depuração permite que o seu aplicativo seja executado um pouco mais rapidamente e também emula mais rigorosamente o modo de execução de um aplicativo instalado.

-**atlogin** Simula a execução do aplicativo no login. Esta flag permite testar a lógica do aplicativo executada somente quando o aplicativo está configurado para iniciar no login do usuário. Quando `-atlogin` é utilizado, a propriedade `reason` do objeto InvokeEvent enviado para o aplicativo será `login`, em vez de `standard` (a não ser que o aplicativo já esteja sendo executado).

***-profile profileName* O ADL depura o aplicativo utilizando o perfil especificado.** O *`profileName`* pode ser um dos seguintes valores:

- desktop
- extendedDesktop
- mobileDevice

Se o descritor do aplicativo inclui um elemento supportedProfiles, o perfil que você especificar com -profile deve ser um membro da lista de suporte. Se o indicador `-profile` não for usado, o primeiro perfil no descritor do aplicativo é usado como o perfil ativo. Se o descritor do aplicativo não inclui o elemento supportedProfiles e você não usar o `indicador` -profile, o perfil *desktop* é usado.

Para obter mais informações, consulte os perfis "supportedProfiles" na página 243 e "Perfis de dispositivo" na página 249.

Valor -**screensize** O tamanho da tela simulada a ser usada quando aplicativos estiverem rodando no perfil MobileDevice no desktop. Especifique o tamanho da tela como um tipo de tela predefinido ou conforme as dimensões em pixels da largura e altura normais e da largura e da altura do layout retrato, além da largura e da altura da tela cheia. Para especificar o valor por tipo, use um dos seguintes tipos de tela predefinidos:

Tipo de tela	Largura x altura normal	Largura x altura tela cheia
480	720 x 480	720 x 480
720	1280x 720	1280x 720
1080	1920 x 1080	1920 x 1080
Droid	480 x 816	480x 854
FWQVGA	240 x 432	240 x 432
FWVGA	480x 854	480x 854
HVGA	320 x 480	320 x 480
iPad	768 x 1004	768 x 1024

Tipo de tela	Largura x altura normal	Largura x altura tela cheia
iPadRetina	1536 x 2008	1536 x 2048
iPhone	320 x 460	320 x 480
iPhoneRetina	640x 920	640 x 960
iPhone5Retina	640x1096	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	640x 920	640 x 960
iPod5Retina	640x1096	640 x 1136
NexusOne	480x 762	480 x 800
QVGA	240 x 320	240 x 320
SamsungGalaxyS	480x 762	480 x 800
SamsungGalaxyTab	600x 986	600 x 1024
WQVGA	240 x 400	240 x 400
WVGA	480 x 800	480 x 800

Para especificar as dimensões de pixels da tela diretamente, utilize o seguinte formato:

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Sempre especifique as dimensões de pixel para o layout retrato, o que significa especificar a largura como um valor menor do que o valor da altura. Por exemplo, a tela NexusOne pode ser especificada com:

```
-screensize 480x762:480x800
```

-extdir *extension-directory* O diretório no qual o runtime deve buscar extensões nativas. O diretório contém um subdiretório para cada extensão nativa que o aplicativo utilizar. Cada um desses subdiretórios contém o arquivo ANE *desempacotado* de uma extensão. Por exemplo:

```
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
```

Ao utilizar o parâmetro -extdir, considere o seguinte:

- O comando ADL exige que cada um dos diretórios especificados tenha a extensão do nome de arquivo .ane. No entanto, o trecho do nome de arquivo antes do sufixo ".ane" pode ser qualquer nome de arquivo válido. Ele*não* precisa corresponder ao valor do elemento `extensionID` do arquivo de indexação do aplicativo.
- É possível especificar o parâmetro `-extdir` mais de uma vez.
- O uso do parâmetro`-extdir` é diferente para a ferramenta ADT e a ferramenta ADL. Na ADT, o parâmetro especifica um diretório que contenha arquivos ANE.
- Também é possível utilizar a variável de extensão`AIR_EXTENSION_PATH` para especificar os diretórios de extensão. Consulte "Variáveis de ambiente ADT" na página 192.

application.xml O arquivo do descritor do aplicativo. Consulte "Arquivos descritores do aplicativo do AIR" na página 208. O descritor do aplicativo é o único parâmetro exigido pelo ADL e, na maioria dos casos, o único parâmetro necessário.

root-directory Especifica o diretório raiz do aplicativo a ser executado. Se não especificado, o diretório que contém o arquivo do descritor do aplicativo será usado.

-- ***arguments*** Qualquer sequência de caracteres que apareça após "--" é transmitida ao aplicativo como argumentos de linha de comando.

Nota: *Quando você inicia um aplicativo do AIR já em execução, uma nova instância desse aplicativo não é iniciada. Em vez disso, um evento* `invoke` *é despachado para a instância em execução.*

Exemplos de ADL

Execute um aplicativo no diretório atual:

```
adl myApp-app.xml
```

Execute um aplicativo em um subdiretório do diretório atual:

```
adl source/myApp-app.xml release
```

Execute um aplicativo e transmita dois argumentos de linha de comando, "tick" e "tock":

```
adl myApp-app.xml -- tick tock
```

Execute um aplicativo usando um runtime específico:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Execute um aplicativo sem o suporte de depuração:

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

Execute um aplicativo no perfil do dispositivo móvel e simular o tamanho da tela do Nexus One:

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Execute um aplicativo usando Apache Ant para executar o aplicativo (os caminhos mostrados no exemplo são 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 erro e saída do ADL

A tabela a seguir descreve os códigos de saída impressos pelo ADL:

Código de saída	Descrição
0	Inicialização bem-sucedida. O ADL é encerrado após o aplicativo do AIR ser encerrado.
1	Invocação bem-sucedida de um aplicativo do AIR já em execução. O ADL é encerrado imediatamente.
2	Erro de uso. Os argumentos fornecidos ao ADL estão incorretos.
3	O runtime não pode ser encontrado.
4	O runtime não pode ser iniciado. Muitas vezes, isso ocorre porque a versão especificada no aplicativo não corresponde à versão do runtime.
5	Ocorreu um erro de causa desconhecida.
6	O arquivo do descritor do aplicativo não pode ser encontrado.
7	O conteúdo do descritor do aplicativo não é válido. Esse erro normalmente indica que o XML não é bem formado.
8	O principal arquivo de conteúdo do aplicativo (especificado no elemento <content> do arquivo do descritor do aplicativo) não pode ser encontrado.

Código de saída	Descrição
9	O principal arquivo de conteúdo do aplicativo não é um arquivo SWF ou HTML válido.
10	Este aplicativo não possui suporte ao perfil especificado com a opção -profile.
11	O argumento -screensize não é compatível no perfil atual.

Capítulo 12: AIR Developer Tool (ADT)

O AIR Developer Tool (ADT) é uma ferramenta de linha de comando de várias utilidades para desenvolver aplicativos do AIR. Você pode usar ADT para realizar as seguintes tarefas:

- Compactar um aplicativo do AIR como um arquivo de instalação .air
- Configure um aplicativo do AIR como um instalador nativo - por exemplo, como um arquivo de instalação .exe no Windows, .ipa no iOS ou .apk no Android
- Empacote uma extensão nativa como um arquivo de Extensão Nativa do AIR (ANE)
- Assinar um aplicativo do AIR com um certificado digital
- Mudar (migrar) a assinatura digital utilizada para atualizações de aplicativos
- Determinar os dispositivos conectados a um computador
- Criar um certificado de assinatura do código digital autoassinado
- Instalar, lançar e desinstalar remotamente um aplicativo em um dispositivo móvel
- Instalar e desinstalar remotamente o runtime do AIR em um dispositivo móvel

ADT é um programa Java incluído no AIR SDK. Você deve ter o Java 1.5 ou superior para utilizá-lo. O SDK inclui um arquivo de script para chamar o ADT. Para usar este script, a localização do programa Java deve ser incluído na variável de ambiente do caminho. Se o diretório `bin` do AIR SDK também estiver listado na variável de ambiente do caminho, você pode digitar `adt` na linha de comando, com os argumentos adequados, para chamar o ADT. (Se você não sabe como definir sua variável de ambiente do caminho, consulte a documentação do seu sistema operacional. Como uma ajuda adicional, os procedimentos para definir o caminho na maioria dos sistemas de computador são descritos em "Variáveis de ambiente do caminho" na página 308).

São necessários pelo menos 2GB de memória de computador ao usar o ADT. Se você tem menos memória do que isso, o ADT pode ficar sem memória, especialmente quando ao compactar aplicativos para o iOS.

Pressupondo que o diretório bin de Java e AIR SDK estejam ambos incluídos na variável de caminho, você pode executar o ADT usando a seguinte sintaxe básica:

```
adt -command options
```

Nota: *A maioria dos ambientes de desenvolvimento integrados, incluindo Adobe Flash Builder e Adobe Flash Professional podem empacotar aplicativos do AIR para você. Normalmente, você não precisa usar o ADT para essas tarefas comuns quando já utiliza um ambiente de desenvolvimento. No entanto, você ainda pode precisar utilizar o ADT como ferramenta de linha de comando para funções que não disponíveis no seu ambiente de desenvolvimento integrado. Além disso, você pode usar ADT como ferramenta de linha de comando como parte de um processo de compilação automatizado.*

Comandos do ADT

O primeiro argumento passado para ADT especifica um dos seguintes comandos.

- -package — compacta um aplicativo do AIR ou AIR Native Extension (ANE).
- -prepare — compacta um aplicativo do AIR como arquivo intermediário (AIRI), mas não o assina. Arquivos do AIRI não podem ser instalados.

- -sign — assina um pacote do AIRI produzido com o comando -prepare. Os comandos -prepare e -sign permitem que a compactação e a assinatura sejam feitas em momentos diferentes. Você também pode usar o comando -sign para assinar ou reassinar um pacote ANE.
- -migrate — aplica uma assinatura de migração para um pacote do AIR assinado, o que permite a utilização de um certificado novo ou renovado de assinatura do código.
- -certificate — cria um certificado de assinatura de código digital autoassinado.
- -checkstore — verifica se um certificado digital em um armazenamento de chaves pode ser acessado.
- -installApp — instala um aplicativo do AIR em um dispositivo ou emulador de dispositivo.
- -launchApp — ativa um aplicativo do AIR em um dispositivo ou emulador de dispositivo.
- -appVersion — informa a versão de um aplicativo do AIR instalado atualmente em um dispositivo ou emulador de dispositivo.
- -uninstallApp — desinstala um aplicativo do AIR de um dispositivo ou emulador de dispositivo.
- -installRuntime — instala o runtime do AIR em um dispositivo ou emulador de dispositivo.
- -runtimeVersion — informa a versão de runtime do AIR instalado atualmente em um dispositivo ou emulador de dispositivo.
- -uninstallRuntime — desinstala o runtime do AIR instalado de um dispositivo ou emulador de dispositivo.
- -version — informa o número de versão do ADT.
- -devices — informa as informações do dispositivo a respeito de dispositivos móveis conectados ou emuladores.
- -help — exibe a lista de comandos e opções.

Muitos comandos do ADT compartilham conjuntos relacionadas de parâmetros e sinalizadores de opção. Estes conjuntos de opção são descritos em detalhe separadamente:

- "Opções de assinatura de código ADT" na página 182
- "Opções de caminho e arquivo" na página 185
- "Opções de conexão do depurador" na página 186
- "Opções de extensão nativas" na página 187

Comando package do ADT

O comando `-package` deve ser executado do diretório principal do aplicativo. O comando usa as seguintes sintaxes:

Crie um pacote do AIR a partir dos arquivos de aplicativo do componente:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Crie um pacote nativo a partir de arquivos do aplicativo do componente:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Crie um pacote nativo que inclua uma extensão nativa dos arquivos componentes do aplicativo:

```
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
```

Crie um pacote nativo de um arquivo AIR ou AIRI:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Crie um pacote de extensão nativa dos arquivos de extensão nativa do componente:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Nota: *Não é necessário assinar um arquivo ANE, pois os parâmetros* `AIR_SIGNING_OPTIONS` *são opcionais neste exemplo.*

AIR_SIGNING_OPTIONS As opções de assinatura do AIR identificam o certificado usado para assinar um arquivo de instalação do AIR. As opções de assinatura são totalmente descritas em "Opções de assinatura de código ADT" na página 182.

-migrate Este sinalizador especifica que o aplicativo é assinado com um certificado de migração, além do certificado especificado nos parâmetros `AIR_SIGNING_OPTIONS`. Este sinalizador é válido apenas se você estiver empacotando um aplicativo de desktop como um instalador nativo e o aplicativo usar uma extensão nativa. Em outros casos ocorre um erro. As opções de assinatura para o certificado de migração são especificados como os parâmetros **MIGRATION_SIGNING_OPTIONS**. Essas opções de assinatura são descritas por completo nas "Opções de assinatura de código ADT" na página 182. Usar o sinalizador `-migrate` permite que você crie uma atualização para um instalador de aplicativo de desktop nativo que use uma extensão nativa e altera o código do certificado de assinatura do aplicativo, como quando expira o certificado original. Para obter mais informações, consulte "Assinatura de uma versão atualizada de um aplicativo do AIR" na página 203.

O sinalizador `-migrate` do comando `-package` está disponível no AIR 3.6 e posterior.

-target O tipo de pacote a ser criado. Os tipos de pacotes disponíveis são:

- air — um pacote do AIR. "air" é o valor padrão e o sinalizador -target não precisa ser especificado ao criar arquivos do AIR ou AIRI.
- airn — um pacote de aplicativo nativo para dispositivos no perfil televisão estendida.
- ane —um pacote de extensão nativa do AIR
- Destinos dos pacotes Android:
  - apk — um pacote do Android. Um pacote produzido com este destino só pode ser instalado em um dispositivo Android e não em um emulador.
  - apk-captive-runtime — um pacote Android que inclui a versão aplicativo e uma versão cativa do AIR runtime. Um pacote produzido com este destino só pode ser instalado em um dispositivo Android e não em um emulador.
  - apk-debug — um pacote do Android com informações extras de depuração. (Os arquivos SWF no aplicativo também devem ser compilados com o suporte de depuração.)
  - apk-emulator — um pacote do Android para o uso em um emulador sem suporte de depuração. (Use o destino apk-debug para permitir a depuração dos emuladores e dos dispositivos.)
  - apk-profile — um pacote Android disponível para desempenho do aplicativo e determinação de perfis da memória.
- Destinos dos pacotes iOS:
  - ipa-ad-hoc — um pacote do iOS para distribuição ad hoc.
  - ipa-app-store — um pacote do iOS para distribuição à App Store da Apple
  - ipa-debug — um pacote do Android com informações extras de depuração. (Os arquivos SWF no aplicativo também devem ser compilados com o suporte de depuração.)
  - ipa-test — um pacote de IOS compilado sem otimização ou informações de depuração.
  - ipa-debug-interpreter — funcionalmente igual a um pacote de depuração, mas compila mais rapidamente. No entanto, o ActionScript bytecode é interpretado e não convertido para código de máquina. Como resultado, a execução do código é mais lenta num pacote do intérprete.
  - ipa-debug-interpreter-simulator — de funcionalidade equivalente ao ipa-debug-interpreter, mas empacotado para o simulador iOS. Somente Macintosh. Se você usar esta opção, poderá incluir também a opção -platformsdk, especificando o caminho para o SDK do simulador iOS.
  - ipa-debug-interpreter — funcionalmente igual a um pacote de teste, mas compila mais rapidamente. No entanto, o ActionScript bytecode é interpretado e não convertido para código de máquina. Como resultado, a execução do código é mais lenta num pacote do intérprete.
  - ipa-test-interpreter-simulator — de funcionalidade equivalente ao ipa-test-interpreter, mas empacotado para o simulador iOS. Somente Macintosh. Se você usar esta opção, poderá incluir também a opção -platformsdk, especificando o caminho para o SDK do simulador iOS.
- native — um instalador desktop nativo. O tipo de arquivo produzido é o formato de instalação nativo do sistema operacional no qual o comando é executado:
  - EXE — Windows
  - DMG — Mac
  - DEB — Ubuntu Linux (AIR 2.6 ou anterior)
  - RPM — Fedora ou Linux OpenSuse (AIR 2.6 ou anterior)

Para obter mais informações, consulte "Compactação de um instalador desktop nativo" na página 55.

-amostra (apenas iOS, AIR 3.4 ou superior) Ativa a amostra baseada em telemetria do ActionScript em aplicativos iOS. O uso desse sinalizador permite que você crie o perfil do aplicativo com o Adobe Scout. Embora o Scout possa criar o perfil de qualquer conteúdo da plataforma Flash, a ativação de telemetria detalhada oferece uma visão mais profunda da temporização de função do ActionScript, do DisplayList, da renderização do Stage3D e muito mais. Observe que o uso desse sinalizador terá um pequeno impacto no desempenho, portanto, não use em aplicativos de produção.

-hideAneLibSymbols (apenas iOS, AIR 3.4 ou superior). Os desenvolvedores de aplicativos podem usar várias extensões nativas de diferentes fontes e, se o ANE compartilhar um nome de símbolo comum, o ADT gerará um erro de "símbolo duplicado no arquivo do objeto". Em alguns casos, esse erro pode se manifestar até mesmo como um travamento no runtime. Você pode usar a opção `hideAneLibSymbols` para especificar se os símbolos da biblioteca ANE ficarão visíveis apenas para as fontes da biblioteca (yes) ou para todos (no):

- **yes** — Oculta os símbolos ANE, que resolvem qualquer conflito de símbolo indesejado.
- **no** — (Padrão) Não oculta os símbolos ANE. Este é o comportamento antes do AIR 3.4.

Os desenvolvedores de aplicativos **-embedBitcode** (apenas iOS, AIR 25 ou versões mais recentes) podem usar a opção `embedBitcode` para especificar se desejam ou não incorporar os bitcodes ao aplicativo iOS. O valor padrão deste switch, caso a opção não especificada seja não.

DEBUGGER_CONNECTION_OPTIONS As opções de conexão do depurador especificam se um pacote de depuração deverá tentar se conectar a um depurador remoto que esteja executando num computador ou se deverá ficar na escuta de uma conexão a partir de um depurador remoto. Este conjunto de opções é suportado somente para pacotes de depuração móveis (destinos apk-debug e ipa-debug). Essas opções são descritas em "Opções de conexão do depurador" na página 186.

-airDownloadURL Especifica um URL alternativo para baixar e instalar o runtime do AIR em dispositivos Android. Se não for especificado, um aplicativo do AIR redirecionará o usuário para um runtime do AIR no Android Market se o runtime ainda não estiver instalado.

Se o seu aplicativo for distribuído por meio de um marketplace alternativo (diferente do Android Market administrado pelo Google), então você poderá precisar especificar o URL para baixar o runtime do AIR do mercado. Alguns mercados alternativos não permitem que os aplicativos solicitem um download de fora do mercado. Essa opção só é suportada para pacotes Android.

NATIVE_SIGNING_OPTIONS As opções de assinatura nativa identificam o certificado usado para assinar um arquivo de pacote nativo. Estas opções de assinatura são usadas para aplicar uma assinatura utilizada pelo sistema operacional nativo, não o AIR. As opções são idênticas às AIR_SIGNING_OPTIONS e são completamente descritas em "Opções de assinatura de código ADT" na página 182.

Assinaturas nativas estão disponíveis no Windows e no Android. No Windows, as opções de assinatura do AIR e as nativas devem ser especificadas. No Android apenas as opções de assinatura nativa podem ser especificadas.

Em muitos casos, você pode usar o mesmo certificado de assinatura de código para aplicar tanto uma assinatura do AIR e uma nativa. Contudo, isso não é verdade em todos os casos. Por exemplo, a política do Google para aplicativos enviados ao Android Market determina que todos os aplicativos devam ser assinados com um certificado válido pelo menos até 2033. Isto significa que um certificado emitido por uma autoridade de certificação conhecida, recomendado quando se aplica uma assinatura do AIR, não deve ser usado para assinar um aplicativo do Android. (Nenhuma autoridade de certificação emitirá um certificado de assinatura de código com este longo período de validade).

output O nome do arquivo do pacote a ser criado. A especificação da extensão do arquivo é opcional. Se não especificada, é adicionada uma extensão adequada para o valor -target e para o sistema operacional atual.

app_descriptor O caminho para o arquivo do descritor do aplicativo. O caminho pode ser especificado em relação ao diretório atual ou como um caminho absoluto. *(O arquivo de descritor do aplicativo é renomeado como application.xml no arquivo do AIR.)*

-platformsdk O caminho para a plataforma SDK do dispositivo de destino:

- Android - O AIR 2.6+ SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)
- iOS - O AIR SDK é enviado com um iOS SDK cativo. A opção -platformsdk permite que você empacote aplicativos com um SDK externo para que não fique restrito ao iOS SDK cativo. Por exemplo, se você criou uma extensão com o iOS SDK mais recente, você pode especificar esse SDK ao empacotar seu aplicativo. Além disso, ao usar o ADT com o simulador iOS, você precisa sempre incluir a opção -platformsdk especificando o caminho do SDK do simulador iOS.

-arch Os desenvolvedores de aplicativos podem usar este argumento para criar um APK para plataformas x86. Ele usa os valores a seguir:

- armv7: o ADT cria pacotes APK para a plataforma Android armv7.
- x86: o ADT cria pacotes APK para a plataforma Android x86.

armv7 é o valor padrão quando nenhum valor é especificado

FILE_OPTIONS Identifica os arquivos do aplicativo a serem incluídos no pacote. As opções de arquivo estão descritas na íntegra em "Opções de caminho e arquivo" na página 185. Não especifique as opções de arquivo ao criar um pacote nativo de um arquivo do AIR ou AIRI.

input_airi Especifique ao criar um pacote nativo de um arquivo do AIRI. As AIR_SIGNING_OPTIONS são necessárias se o destino for *air* (ou se nenhum destino for especificado).

input_airi Especifique ao criar um pacote nativo de um arquivo do AIR. Não especifique AIR_SIGNING_OPTIONS.

ANE_OPTIONS Identifica as opções e arquivos para criar um pacote de extensão nativa. As opções de pacote de extensão são descritas em detalhes em "Opções de extensão nativas" na página 187.

Exemplos do comando ADT-package

Arquivos de aplicativo específico ao pacote no diretório atual para um aplicativo do AIR baseado em SWF:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Arquivos de aplicativo específico ao pacote no diretório atual para um aplicativo do AIR baseado em HTML:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js
image.gif
```

Empacote todos os arquivos e subdiretórios no diretório de trabalho atual:

```
adt –package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Nota: *O arquivo de armazenamento de chaves contém a chave privada usada para assinar seu aplicativo. Nunca inclua o certificado de assinatura no pacote do AIR! Se você usar caracteres curinga no comando do ADT, coloque o arquivo de armazenamento de chaves em um local diferente para que ele não seja incluído no pacote. Neste exemplo, o arquivo de armazenamento de chaves, cert.p12, reside no diretório pai.*

Empacote apenas os arquivos principais e um subdiretório de imagens:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Empacote um aplicativo baseado em HTML e todos os arquivos nos subdiretórios de HTML, scripts e imagens:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js
html scripts images
```

Empacote o arquivo application.xml e o SWF principal localizados em um diretório de trabalho (release/bin):

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml –C
release/bin myApp.swf
```

Empacote ativos de mais de um lugar no seu sistema de arquivo de criação. Neste exemplo, os ativos do aplicativo são localizados nas seguintes pastas antes do empacotamento:

```
/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
```

Executar o seguinte comando do ADT do diretório /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
```

Resulta na seguinte estrutura de pacote:

```
/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
```

Execute ADT como um programa Java para um aplicativo simples baseado em SWF (sem configurar o classpath):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Execute ADT como um programa Java para um aplicativo simples baseado em HTML (sem configurar o classpath):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Execute o ADT como um programa Java (com o caminho da classe Java definido para incluir o pacote ADT.jar):

```
java -com.adobe.air.ADT –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

Execute o ADT como uma tarefa Java no Apache Ant (embora seja melhor usar o comando diretamente nos scripts Ant). Os caminhos mostrados no exemplo são para o 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: *Em alguns sistemas de computadores, os caracteres de bytes duplos nos caminhos do sistema de arquivos pode ser mal-interpretado. Se isto acontecer, tente definir o JRE usado para executar o ADT para usar o conjunto de caracteres UTF-8. Isso é feito por padrão no script usado para iniciar o ADT no Mac e no Linux. No arquivo adt.bat do Windows, ou ao executar o ADT diretamente do Java, especifique a opção* `-Dfile.encoding=UTF-8` *na linha de comando Java.*

Comando prepare do ADT

O comando -prepare cria um pacote do AIRI não assinado. Um pacote do AIRI não pode ser usado por si próprio. Use o comando -sign para converter um arquivo do AIRI para um pacote do AIR assinado, ou o comando package para converter o arquivo AIRI para um pacote nativo.

O comando -prepare usa a seguinte sintaxe:

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output O nome do arquivo do AIRI criado.

app_descriptor O caminho para o arquivo do descritor do aplicativo. O caminho pode ser especificado em relação ao diretório atual ou como um caminho absoluto. *(O arquivo de descritor do aplicativo é renomeado como application.xml no arquivo do AIR.)*

FILE_OPTIONS Identifica os arquivos do aplicativo a serem incluídos no pacote. As opções de arquivo estão descritas na íntegra em "Opções de caminho e arquivo" na página 185.

Comando sign do ADT

O comando -sign assina arquivos do AIRI e ANE.

O comando -sign usa a seguinte sintaxe:

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS As opções de assinatura do AIR identificam o certificado usado para assinar um arquivo de pacote. As opções de assinatura são totalmente descritas em "Opções de assinatura de código ADT" na página 182.

input O nome do arquivo do AIRI ou ANE a ser criado.

output O nome do pacote assinado a ser criado.

Se um arquivo ANE já estiver assinado, a assinatura antiga é descartada. (Os arquivos AIR não podem ser assinados novamente — para usar uma nova assinatura para uma atualização do aplicativo, use o comando `-migrate`.)

Comando migrate do ADT

O comando -migrate aplica uma assinatura de migração para um arquivo do AIR. A assinatura de migração deve ser usada quando você renovar ou alterar seu certificado digital e precisar atualizar os aplicativos assinados com o certificado antigo.

Para obter mais informações sobre o empacotamento de aplicativos AIR com uma assinatura de migração, consulte "Assinatura de uma versão atualizada de um aplicativo do AIR" na página 203.

Nota: *O certificado de migração deve ser aplicado no prazo de 365 dias a partir da expiração do certificado. Assim que terminar este período de prorrogação, as atualizações de sua aplicação não podem mais ser assinadas com uma assinatura de migração. Os usuários podem primeiro atualizar para uma versão do seu aplicativo que foi assinado com uma assinatura de migração e instalar a atualização mais recente, ou podem desinstalar o aplicativo original e instalar o pacote novo do AIR.*

Para usar uma assinatura de migração, primeiro assine o aplicativo do AIR usando o certificado novo ou renovado (usando os comandos -package ou -sign), e depois aplique a assinatura de migração usando o certificado antigo e o comando -migrate.

O comando `-migrate` usa a seguinte sintaxe:

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS As opções de assinatura do AIR que identificam o certificado original usado para assinar as versões existentes do aplicativo do AIR. As opções de assinatura são totalmente descritas em "Opções de assinatura de código ADT" na página 182.

input O arquivo do AIR já assinado com o NOVO certificado do aplicativo.

output O nome das assinaturas de tolerância do pacote final dos certificados novos e antigos.

Os nomes de arquivos usados para os arquivos do AIR de entrada e de saída devem ser diferentes.

Nota: *O comando migrate do ADT não pode ser usado com aplicativos AIR para desktop que incluem extensões nativas, pois estes são empacotados como os instaladores nativos, não como arquivos .air. Para alterar certificados para um aplicativo AIR para desktop que inclua uma extensão nativa, empacote o aplicativo usando o "Comando package do ADT" na página 169 com o sinalizador -migrate.*

Comando checkstore do ADT

O comando -checkstore permite verificar a validade de um armazenamento de chaves. O comando usa a seguinte sintaxe:

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS As opções de assinatura que identificam o armazenamento de chaves para validação. As opções de assinatura são totalmente descritas em "Opções de assinatura de código ADT" na página 182.

Comando certificate do ADT

O comando -certificate permite criar um certificado de assinatura de código digital autoassinado. O comando usa a seguinte sintaxe:

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type
output password
```

-cn A sequência de caracteres atribuída como o nome comum do novo certificado.

-ou Uma sequência de caracteres atribuída como a unidade organizacional que emite o certificado. (Opcional.)

-o Uma sequência de caracteres atribuída como a organização que emite o certificado. (Opcional.)

-c Um código de país de duas letras ISO-3166. Um certificado não é gerado se um código inválido é fornecido. (Opcional.)

-validityPeriod O número de anos nos quais o certificado é válido. Se não for especificada, é atribuída uma validade de cinco anos. (Opcional.)

key_type: o tipo de chave a ser usada para o certificado é *2048-RSA*.

output O caminho e o nome do arquivo para o arquivo do certificado a ser gerado.

password A senha para acessar o novo certificado. A senha é necessária ao assinar arquivos do AIR com esse certificado.

Comando installApp do ADT

O comando -installApp instala um aplicativo em um dispositivo ou em um emulador.

Você deve desinstalar um aplicativo existente antes de reinstalar com esse comando.

O comando usa a seguinte sintaxe:

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package
fileName
```

-platform O nome da plataforma do dispositivo. Especifique *ios* ou *android*.

-platformsdk O caminho para a plataforma SDK para o dispositivo de destino (opcional):

- Android - O AIR 2.6+ SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)

- iOS - O AIR SDK é enviado com um iOS SDK cativo. A opção -platformsdk permite que você empacote aplicativos com um SDK externo para que não fique restrito ao iOS SDK cativo. Por exemplo, se você criou uma extensão com o iOS SDK mais recente, você pode especificar esse SDK ao empacotar seu aplicativo. Além disso, ao usar o ADT com o simulador iOS, você precisa sempre incluir a opção -platformsdk especificando o caminho do SDK do simulador iOS.

-device Especifica *ios_simulator*, o número de série (Android) ou identificador (iOS) do dispositivo conectado. No iOS, esse parâmetro é obrigatório. No Android, esse parâmetro precisa ser especificado somente quando mais de um dispositivo Android ou emulador é conectado ao seu computador e executado. Se o dispositivo especificado não estiver conectado, o ADT retorna o código de saída 14: Erro de dispositivo (Android) ou Dispositivo inválido especificado (iOS). Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

***Nota:** A instalação de um arquivo IPA diretamente em um dispositivo iOS está disponível no AIR 3.4 e superior e precisa do iTunes 10.5.0 ou superior.*

Use o comando `adt -devices` (disponível no AIR 3.4 e superior) para determinar o identificador ou o número de série dos dispositivos conectados. Observe que no iOS você usa o identificador, não o UUID do dispositivo. Para obter mais informações, consulte "Comando de dispositivos do ADT" na página 182.

Além disso, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

-package O nome do arquivo do pacote a ser instalado. No iOS, ele deve ser um arquivo IPA. No Android este deve ser um pacote APK. Se o pacote especificado já estiver instalado, o ADT retorna o código de erro 14: Erro do dispositivo.

Comando appVersion do ADT

O comando -appVersion informa a versão instalada de um aplicativo em um dispositivo ou emulador. O comando usa a seguinte sintaxe:

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform O nome da plataforma do dispositivo. Especifique *ios* ou *android*.

-platformsdk O caminho para a plataforma SDK do dispositivo de destino:

- Android - O AIR 2.6+ SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)
- iOS - O AIR SDK é enviado com um iOS SDK cativo. A opção -platformsdk permite que você empacote aplicativos com um SDK externo para que não fique restrito ao iOS SDK cativo. Por exemplo, se você criou uma extensão com o iOS SDK mais recente, você pode especificar esse SDK ao empacotar seu aplicativo. Além disso, ao usar o ADT com o simulador iOS, você precisa sempre incluir a opção -platformsdk especificando o caminho do SDK do simulador iOS.

-device especifique *ios_simulator* ou o número de série do dispositivo. O dispositivo só deve ser especificado quando mais de um dispositivo ou emulador Android estiver conectado ao computador e em funcionamento. Se o dispositivo especificado não estiver conectado, o ADT retorna código de saída 14: Erro do dispositivo. Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

No Android, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

-appid O ID do aplicativo do AIR do aplicativo instalado. Se nenhum aplicativo com o ID especificada estiver instalado no dispositivo, o ADT retorna o código de saída 14: Erro do dispositivo.

Comando launchApp do ADT

O comando -launchApp executa um aplicativo instalado em um dispositivo ou emulador. O comando usa a seguinte sintaxe:

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform O nome da plataforma do dispositivo. Especifique *ios* ou *android*.

-platformsdk O caminho para a plataforma SDK do dispositivo de destino:

- Android - O AIR 2.6+ SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)
- iOS - O AIR SDK é enviado com um iOS SDK cativo. A opção -platformsdk permite que você empacote aplicativos com um SDK externo para que não fique restrito ao iOS SDK cativo. Por exemplo, se você criou uma extensão com o iOS SDK mais recente, você pode especificar esse SDK ao empacotar seu aplicativo. Além disso, ao usar o ADT com o simulador iOS, você precisa sempre incluir a opção -platformsdk especificando o caminho do SDK do simulador iOS.

-device especifique *ios_simulator* ou o número de série do dispositivo. O dispositivo só deve ser especificado quando mais de um dispositivo ou emulador Android estiver conectado ao computador e em funcionamento. Se o dispositivo especificado não estiver conectado, o ADT retorna código de saída 14: Erro do dispositivo. Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

No Android, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

-appid O ID do aplicativo do AIR do aplicativo instalado. Se nenhum aplicativo com o ID especificada estiver instalado no dispositivo, o ADT retorna o código de saída 14: Erro do dispositivo.

Comando uninstallApp do ADT

O comando -uninstallApp remove completamente um aplicativo instalado em um dispositivo ou emulador remoto. O comando usa a seguinte sintaxe:

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform O nome da plataforma do dispositivo. Especifique *ios* ou *android*.

-platformsdk O caminho para a plataforma SDK do dispositivo de destino:

- Android - O AIR 2.6+ SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)
- iOS - O AIR SDK é enviado com um iOS SDK cativo. A opção -platformsdk permite que você empacote aplicativos com um SDK externo para que não fique restrito ao iOS SDK cativo. Por exemplo, se você criou uma extensão com o iOS SDK mais recente, você pode especificar esse SDK ao empacotar seu aplicativo. Além disso, ao usar o ADT com o simulador iOS, você precisa sempre incluir a opção -platformsdk especificando o caminho do SDK do simulador iOS.

-device especifique *ios_simulator* ou o número de série do dispositivo. O dispositivo só deve ser especificado quando mais de um dispositivo ou emulador Android estiver conectado ao computador e em funcionamento. Se o dispositivo especificado não estiver conectado, o ADT retorna código de saída 14: Erro do dispositivo. Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

No Android, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

-appid O ID do aplicativo do AIR do aplicativo instalado. Se nenhum aplicativo com o ID especificada estiver instalado no dispositivo, o ADT retorna o código de saída 14: Erro do dispositivo.

Comando installRuntime do ADT

O comando -installRuntime instala o runtime do AIR em um dispositivo.

Você deve desinstalar a versão existente do runtime AIR antes de reinstalar com esse comando.

O comando usa a seguinte sintaxe:

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package
fileName
```

-platform O nome da plataforma do dispositivo. No momento este comando está disponível apenas na plataforma do Android. Use o nome, *android*.

-platformsdk O caminho para a plataforma SDK para o dispositivo de destino. No momento a única plataforma SDK disponível é Android. O AIR 2.6 SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)

-device O número de série do dispositivo. O dispositivo só deve ser especificado quando mais de um dispositivo ou emulador estiver conectado ao computador e em funcionamento. Se o dispositivo especificado não estiver conectado, o ADT retorna código de saída 14: Erro do dispositivo. Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

No Android, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

-package O nome do arquivo do runtime a ser instalado. No Android este deve ser um pacote APK. Se nenhum pacote for especificado, o runtime adequado para o dispositivo ou emulador é escolhido entre os disponíveis no AIR SDK. Se o runtime já estiver instalado, o ADT retorna o código de erro 14: Erro do dispositivo.

Comando runtimeVersion do ADT

O comando -runtimeVersion informa a versão instalada do runtime do AIR em um dispositivo ou emulador. O comando usa a seguinte sintaxe:

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform O nome da plataforma do dispositivo. No momento este comando está disponível apenas na plataforma do Android. Use o nome, *android*.

-platformsdk O caminho para a plataforma SDK para o dispositivo de destino. No momento a única plataforma SDK disponível é Android. O AIR 2.6 SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)

-device O número de série do dispositivo. O dispositivo só deve ser especificado quando mais de um dispositivo ou emulador estiver conectado ao computador e em funcionamento. Se o runtime não estiver instalado ou o dispositivo especificado não estiver conectado, o ADT retorna o código de saída 14: Erro do dispositivo. Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

No Android, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

Comando uninstallRuntime do ADT

O comando -uninstallRuntime remove completamente o runtime do AIR de um dispositivo ou emulador. O comando usa a seguinte sintaxe:

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform O nome da plataforma do dispositivo. No momento este comando está disponível apenas na plataforma do Android. Use o nome, *android*.

-platformsdk O caminho para a plataforma SDK para o dispositivo de destino. No momento a única plataforma SDK disponível é Android. O AIR 2.6 SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Além disso, o caminho SDK da plataforma não precisa ser fornecido na linha do comando se a variável de ambiente AIR_ANDROID_SDK_HOME já estiver definida. (Se ambos estiverem definidos, o caminho fornecido na linha de comando é usado.)

-device O número de série do dispositivo. O dispositivo só deve ser especificado quando mais de um dispositivo ou emulador estiver conectado ao computador e em funcionamento. Se o dispositivo especificado não estiver conectado, o ADT retorna código de saída 14: Erro do dispositivo. Se mais de um dispositivo ou emulador estiver conectado e um dispositivo não for especificado, o ADT retorna o código de saída 2: Erro de uso.

No Android, use a ferramenta ADB do Android para listar os números de série dos dispositivos conectados e emuladores em execução:

```
adb devices
```

Comando de dispositivos do ADT

O comando -help do ADT exibe os IDs de dispositivo dos dispositivos móveis e emuladores conectados no momento:

```
adt -devices -platform iOS|android
```

-platform O nome da plataforma a ser verificada. Especifique `android` ou `iOS`.

Nota: *No iOS, esse comando precisa do iTunes 10.5.0 ou superior.*

Comando help do ADT

O comando -help do ADT exibe um lembrete resumido das opções da linha de comando:

```
adt -help
```

A saída help usa as seguintes convenções simbólicas:

- <> — itens entre colchetes angulares indicam as informações que você deve fornecer.
- () — itens entre parênteses indicam as opções tratadas como um grupo na saída do comando help.
- ALL_CAPS — itens escritos em letras maiúsculas indicam um conjunto de opções descrito separadamente.
- | — Ou. Por exemplo, (A | B), significa item A ou item B.
- ? — 0 ou 1. Um ponto de interrogação na sequência de um item indica que um item é opcional e que somente uma instância pode ocorrer, se utilizado.
- * — 0 ou mais. Um asterisco na sequência de um item indica que um item é opcional e que qualquer número de instâncias pode ocorrer.
- + — 1 ou mais. Um sinal de mais na sequência de um item indica que um item é obrigatório e que várias instâncias podem ocorrer.
- sem símbolo - Se um item não tiver nenhum símbolo de sufixo, esse item é obrigatório e somente uma instância pode ocorrer.

Conjuntos de opções do ADT

Vários dos comandos do ADT compartilham conjuntos comuns de opções.

Opções de assinatura de código ADT

O ADT usa o JCA (arquitetura de criptografia Java) para acessar chaves privadas e certificados para assinar aplicativos do AIR. As opções de assinatura identificam o armazenamento de chaves, a chave privada e o certificado dentro desse armazenamento de chaves.

O armazenamento de chaves deve incluir a chave privada e a cadeia de certificado associada. Se o certificado de assinatura estiver vinculado a um certificado confiável em um computador, o conteúdo do campo de nome comum do certificado será exibido como o nome do editor na caixa de diálogo de instalação do AIR.

O ADT requer que o certificado esteja em conformidade com o padrão x509v3 (RFC3280) e inclua a extensão de uso de chave estendida com os valores adequados para assinatura do código. As restrições definidas no certificado são respeitadas e poderiam impedir o uso de alguns certificados para assinar aplicativos do AIR.

***Nota:** O ADT usa as configurações de proxy do ambiente de runtime Java, quando apropriado, para conectar aos recursos de Internet para verificar listas de revogação de certificado e obter carimbos de data/hora. Se encontrar problemas para se conectar a estes recursos da Internet ao usar o ADT e sua rede exigir configurações de proxy específicas, você pode precisar configurar as configurações de proxy do JRE.*

Sintaxe de opções de assinatura do AIR

As opções de subscrição utilizam a seguinte sintaxe (numa linha de comando individual):

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias O alias de uma chave no armazenamento de chaves. Especificar um alias não é necessário quando um armazenamento de chaves contém apenas um único certificado. Se nenhum alias for especificado, o ADT usará a primeira chave do armazenamento de chaves.

Nem todos os aplicativos de gerenciamento do armazenamento de chaves permitem que um alias seja atribuído a certificados. Ao usar o armazenamento de chaves do sistema Windows, por exemplo, use o nome distinto do certificado como o alias. Você pode usar o utilitário Java Keytool para listar os certificados disponíveis para que possa determinar o alias. Por exemplo, executar o comando:

```
keytool -list -storetype Windows-MY
```

produz uma saída como a seguinte para um certificado:

```
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 se referir a esse certificado na linha de comando do ADT, defina o alias como:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

No Mac OS X, o alias de um certificado no Keychain é o nome exibido no aplicativo Keychain Access.

-storetype O tipo de armazenamento de chaves, determinado pela implementação do armazenamento de chaves. A implementação de armazenamento de chaves padrão incluída na maioria das instalações de Java suporta os tipos `JKS` e `PKCS12`. O Java 5.0 inclui suporte para o tipo `PKCS11`, para acessar armazenamentos de chaves em tokens de hardware e para o tipo `Keychain`, para acessar o chaveiro do Mac OS X. O Java 6.0 inclui suporte para o tipo `MSCAPI` (no Windows). Se outros provedores de JCA tiverem sido instalados e configurados, tipos adicionais de armazenamentos de chaves podem estar disponíveis. Se nenhum tipo de armazenamento de chave for especificado, o tipo padrão para o provedor de JCA padrão será usado.

Tipo de armazenamento	Formato de armazenamento de chave	Versão mínima de Java
JKS	Arquivo de armazenamento de chave Java (.keystore)	1.2
PKCS12	Arquivo PKCS12 (.p12 ou .pfx)	1.4

Tipo de armazenamento	Formato de armazenamento de chave	Versão mínima de Java
PKCS11	Token de hardware	1.5
KeychainStore	Chaveiro do Mac OS X	1.5
Windows-MY ou Windows-ROOT	MSCAPI	1.6

-keystore O caminho para o arquivo de armazenamento de chaves para tipos de armazenamento com base no arquivo.

-storepass A senha exigida para acessar o armazenamento de chaves. Se não especificada, o ADT solicita a senha.

-keypass A senha exigida para acessar a chave privada usada para assinar o aplicativo do AIR. Se não especificada, o ADT solicita a senha.

Nota: *Se você digitar uma senha como parte do comando do ADT, os caracteres da senha são gravados no histórico da linha de comando. Portanto, o uso de opções -keypass ou -storepass não é recomendado quando a segurança do certificado é importante. Observe também que ao omitir as opções de senha, os caracteres que você digita nos prompts de senha não são exibidos (por razões de segurança). Basta digitar a senha e pressionar a tecla Enter.*

-providerName O provedor de JCA para o tipo de armazenamento de chaves especificado. Se não especificado, o ADT usa o provedor padrão para esse tipo de armazenamento de chave.

-tsa Especifica a URL de um servidor compatível com carimbo de data e hora RFC3161 para carimbar com data e hora a assinatura digital. Se nenhuma URL for especificada, um servidor padrão com carimbo de data/hora fornecido pela Geotrust será usado. Quando a assinatura de um aplicativo do AIR receber uma marca de data/hora, o aplicativo poderá ainda ser instalado depois que o certificado de assinatura expirar, porque a marca de data/hora verifica se o certificado era válido no momento da assinatura.

Se o ADT não puder se conectar ao servidor de carimbo de data/hora, a assinatura será cancelada e nenhum empacotamento será produzido. Especifique `-tsa none` para desabilitar o carimbo de data/hora. No entanto, um aplicativo do AIR empacotado sem um carimbo de data/hora deixa de poder ser instalado depois que o certificado de assinatura expira.

Nota: *Muitas das opções de assinatura são equivalentes à mesma opção do utilitário Java Keytool. Você pode usar o utilitário Keytool para examinar e gerenciar armazenamentos de chaves no Windows. O utilitário de segurança da Apple® também pode ser usado para esse fim no Mac OS X.*

-provisioning-profile O arquivo de aprovisionamento iOS da Apple. (Obrigatório apenas para compactação de aplicativos iOS.)

Exemplos de opção de assinatura

Assinatura com um arquivo .p12:

```
-storetype pkcs12 -keystore cert.p12
```

Assinatura com o armazenamento de chaves Java padrão:

```
-alias AIRcert -storetype jks
```

Assinatura com o armazenamento de chaves Java específico:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Assinatura com o chaveiro do Mac OS X:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Assinatura com o armazenamento de chaves do sistema Windows:

```
-alias cn=AIRCert -storeype Windows-MY
```

Assinatura com um token de hardware (consulte as instruções do fabricante do token sobre como configurar Java para usar o token e para o valor correto de `providerName`):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Assinatura sem incorporar um carimbo de data/hora:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Opções de caminho e arquivo

As opções de arquivo e caminho especificam todos os arquivos incluídos no pacote. As opções de arquivo e caminho usam a seguinte sintaxe:

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

file_and_dirs Os arquivos e diretórios a empacotar no arquivo do AIR. Qualquer número de arquivos e diretórios pode ser especificado, delimitado por um espaço em branco. Se você listar um diretório, todos os arquivos e subdiretórios dentro dele, exceto arquivos ocultos, serão adicionados ao pacote. (Além disso, se o arquivo do descritor do aplicativo for especificado, diretamente ou por caractere curinga ou expansão de diretório, ele será ignorado e não adicionado ao pacote uma segunda vez.) Arquivos e diretórios especificados devem estar no diretório atual ou em um de seus subdiretórios. Use a opção `-C` para alterar o diretório atual.

Importante: *Caracteres curinga não podem ser usados nos argumentos `file_or_dir` depois da opção `–C`. (Shells de comando expandem os caracteres curinga antes de transmitir os argumentos para o ADT, o que faz com que o ADT procure arquivos no lugar errado.) Você pode, no entanto, usar ainda o caractere de ponto, ".", para representar o diretório atual. Por exemplo, `-C assets .` copia tudo no diretório de ativos, incluindo qualquer subdiretório, para o nível raiz do pacote do aplicativo.*

`-C dir files_and_dirs` Altera o diretório de trabalho para o valor de *dir* antes de processar arquivos e diretórios subsequentes adicionados ao pacote do aplicativo (especificado em **`files_and_dirs`**). Os arquivos ou diretórios são adicionados à raiz do pacote do aplicativo. A opção `–C` pode ser usada quantas vezes for preciso para incluir arquivos de vários pontos no sistema de arquivos. Se um caminho relativo for especificado para `dir`, o caminho sempre é resolvido do diretório de trabalho original.

À medida que o ADT processa os arquivos e diretórios incluídos no pacote, os caminhos relativos entre o diretório atual e os arquivos de destino são armazenados. Esses caminhos são expandidos na estrutura do diretório do aplicativo quando o pacote é instalado. Portanto, especificar `-C release/bin lib/feature.swf` coloca o arquivo `release/bin/lib/feature.swf` no subdiretório `lib` da pasta do aplicativo raiz.

`-e file_or_dir dir` Coloca o arquivo ou diretório no diretório do pacote especificado. Essa opção não pode ser usada ao compactar um arquivo ANE.

Nota: *O elemento `<content>` do arquivo do descritor do aplicativo deve especificar o local final do arquivo do aplicativo principal na árvore do diretório do pacote do aplicativo.*

-extdir *dir* O valor de`dir` é o nome de um diretório onde buscar extensões nativas (arquivos ANE). Especifique um caminho absoluto ou um caminho relativo para o diretório atual. É possível especificar a opção `-extdir`várias vezes.

O diretório especificado contém arquivos ANE de extensões nativas que o aplicativo utiliza. Cada arquivo ANE nesse diretório tem uma extensão de nome de arquivo .ane. Contudo, o nome de arquivo antes da extensão de nome de arquivo .ane *não* precisa corresponder ao valor do elemento `extensionID` do arquivo descritor do aplicativo.

Por exemplo, se você utilizar `-extdir ./extensions`, as extensões do `diretório` poderão ter a aparência semelhante à seguinte:

```
extensions/
    extension1.ane
    extension2.ane
```

Nota: *O uso da opção -extdir é diferente para a ferramenta ADT e para a ferramenta ADL. Na ADL, a opção especifica um diretório que contém subdiretórios, cada um contendo um arquivo ANE desempacotado. Na ADT, as opções especificam um diretório que contém arquivos ANE.*

Opções de conexão do depurador

Quando o destino do aplicativo for apk-debug, ipa-debug ou ipa-debug-interpreter, as opções de conexão poderão ser usadas para especificar se o aplicativo tentará se conectar a um depurador remoto (normalmente usado para depuração wifi) ou se ouvirá uma conexão de entrada de um depurador remoto (normalmente usado para depuração USB). Use a opção `-connect` para conectar a um depurador; use a opção `-listen` para aceitar uma conexão proveniente de um depurador em uma conexão USB. Essas opções se excluem, ou seja, não podem ser utilizadas simultaneamente.

A opção -connect usa a seguinte sintaxe:

```
-connect hostString
```

-connect Se estiver presente, o aplicativo tentará se conectar a um depurador remoto.

hostString Uma sequência de caracteres que identifica o computador que está executando a ferramenta de depuração Flash FDB. Se não for especificado, o aplicativo tentará se conectar a um depurador em execução no computador no qual o pacote é criado. A sequência de caracteres de host pode ser um nome de domínio totalmente qualificado do computador: *machinename.subgroup.example.com*, ou um endereço IP: *192.168.4.122*. Se a máquina especificada (ou padrão) não puder ser encontrada, o runtime exibirá uma caixa de diálogo solicitando um nome de host válido.

A opção `--listen`usa a seguinte sintaxe:

```
-listen port
```

-listen Se estiver presente, o tempo de execução irá aguardar uma conexão proveniente de um depurador remoto.

port (Opcional) A porta que será escutada. Por padrão, o runtime escuta na porta 7936. Para mais informações sobre o uso da opção `-listen`, consulte "Depuração remota com FDB através de USB" na página 107.

Opções de determinação de perfis do aplicativo do Android

Quando o destino do pacote for apk-profile, as opções do gerador de perfil podem ser usadas para especificar qual arquivo SWF pré-carregado a ser usado para o desempenho e a determinação de perfis de memória. As opções do gerador de perfil usam a seguinte sintaxe:

```
-preloadSWFPath directory
```

-preloadSWFPath Se estiver presente, o aplicativo tentará localizar o arquivo SWF pré-carregado no diretório especificado. Se não for especificado, o ADT inclui o arquivo SWF pré-carregado do AIR SDK.

directory O diretório que contém o arquivo SWF pré-carregado do gerador de perfis.

Opções de extensão nativas

As opções de extensão nativas especificam as opções e arquivos para desempacotar um arquivo ANE para uma extensão nativa. Utilize essas opções com um comando do pacote ADT no qual a opção de `-target` seja `ane`.

```
extension-descriptor -swc swcPath
   -platform platformName
  -platformoptions path/platform.xml
   FILE_OPTIONS
```

extension-descriptor O arquivo de indexação para a extensão nativa.

-swc O arquivo SWC contendo o código ActionScript e recursos para a extensão nativa.

-platform O nome da plataforma que oferece suporte para este arquivo ANE. É possível incluir várias opções de `-platform`, cada uma com suas `FILE_OPTIONS`.

-platformoptions O caminho para um arquivo de opções de plataforma (platform.xml). Use este arquivo para especificar opções de vinculação não padrão, as bibliotecas compartilhadas e bibliotecas estáticas de terceiros usadas pela extensão. Para obter mais informações e exemplos, consulte bibliotecas iOS nativas.

FILE_OPTIONS Identifica os arquivos de plataforma nativos a serem incluídos no pacote, como bibliotecas a serem incluídas no pacote de extensão nativa. As opções de arquivo estão descritas na íntegra em "Opções de caminho e arquivo" na página 185. (Observe que a opção -e não pode ser usada ao compactar um arquivo ANE.)

Mais tópicos da Ajuda

Compactando uma extensão nativa

Mensagens de erro do ADT

As tabelas a seguir listam os erros que o programa ADT pode reportar e suas possíveis causas:

Erros de validação do descritor do aplicativo

Código do erro	Descrição	Observações
100	Não é possível analisar o descritor do aplicativo	Verifique o arquivo do descritor do aplicativo para obter os erros de sintaxe XML, como tags abertas.
101	Espaço para nomes ausente	Adicione o espaço para nomes ausente.
102	Espaço para nomes inválido	Verifique a ortografia do espaço para nomes.
103	Elemento ou atributo inesperado	Remova os elementos e atributos incorretos. Valores personalizados não são permitidos no arquivo de descritor. Verifique a ortografia dos nomes do elemento e dos atributos. Verifique se os elementos estão inseridos no elemento pai correto e se os atributos são usados com os elementos corretos.
104	Elemento ou atributo ausente	Adicione o elemento ou atributo necessário.

Código do erro	Descrição	Observações
105	Elemento ou atributo contém um valor inválido	Corrija o valor incorreto.
106	Combinação de atributos de janela ilegal	Não é possível usar algumas configurações de janela, como `transparency = true` e `systemChrome = standard` juntas. Altere uma das configurações incompatíveis.
107	O tamanho mínimo da janela é maior que o tamanho máximo da janela	Altere a configuração do tamanho mínimo ou do máximo.
108	Atributo já usado no elemento anterior	
109	Elemento duplicado.	Remove o elemento duplicado.
110	É necessário pelo menos um elemento do tipo especificado.	Adicione o elemento ausente.
111	Nenhum dos perfis listados no descritor do aplicativo é compatível com extensões nativas.	Adicione um perfil para a lista supportedProfies que oferece suporte para extensões do nativas.
112	O destino do AIR não oferece extensões nativas.	Escolha um destino que suporte extensões nativas.
113	<nativeLibrary> e <initializer> devem ser fornecidos juntos.	Uma função de inicialização deverá ser especificada para cada biblioteca nativa na extensão nativa.
114	Encontrado <finalizer> sem <nativeLibrary>.	Não especifique um finalizador a menos que a plataforma use uma biblioteca nativa.
115	A plataforma padrão não deve conter uma implementação nativa.	Não especifique uma biblioteca nativa no elemento da plataforma padrão.
116	A execução do navegador não é suportada para esse destino.	O elemento `<allowBrowserInvocation>` não pode ser `true` para o destino de compactação especificado.
117	Esse destino exige pelo menos um namespace para empacotar extensões nativas.	Altere o namespace na indexação do aplicativo para um valor suportado.

Consulte “Arquivos descritores do aplicativo do AIR” na página 208 para obter informações sobre namespaces, elementos, atributos e seus valores válidos.

Erros do ícone do aplicativo

Código do erro	Descrição	Observações
200	Não é possível abrir o arquivo de ícone	Verifique se o arquivo existe no caminho especificado. Use outro aplicativo para garantir que o arquivo possa ser aberto.
201	Ícone no tamanho errado	O tamanho do ícone (em pixels) deve corresponder à tag XML. Por exemplo, dado o elemento do descritor do aplicativo: `<image32x32>icon.png</image32x32>` A imagem em `icon.png` deve ser exatamente de 32x32 pixels.
202	O arquivo do ícone contém um formato de imagem sem suporte	Somente o formato PNG tem suporte. Converta imagens em outros formatos antes de empacotar o aplicativo.

Erros de arquivo no aplicativo

Código do erro	Descrição	Observações
300	Arquivo ausente ou não é possível abrir arquivo	Não é possível encontrar ou abrir um arquivo especificado na linha de comando.
301	Arquivo do descritor de aplicativo ausente ou não é possível abri-lo	O arquivo do descritor do aplicativo não pode ser encontrado no caminho especificado ou não pode ser aberto.
302	Arquivo do conteúdo raiz ausente do pacote	É necessário adicionar o arquivo SWF ou HTML referenciado no elemento `<content>` do descritor do aplicativo ao pacote incluindo-o nos arquivos listados na linha de comando do ADT.
303	Arquivo de ícone ausente do pacote	É necessário adicionar os arquivos de ícone especificados no descritor do aplicativo ao pacote incluindo-os entre os arquivos listados na linha de comando do ADT. Os arquivos de ícone não são adicionados automaticamente.
304	Conteúdo da janela inicial inválido	O arquivo referenciado no elemento `<content>` do descritor do aplicativo não é reconhecido como arquivo HTML ou SWF válido.

Código do erro	Descrição	Observações
305	A versão inicial do SWF de conteúdo da janela excedeu a versão do espaço para nomes	A versão SWF do arquivo referenciado no elemento `<content>` do descritor do aplicativo não tem suporte da versão do AIR especificado no espaço para nomes do descritor. Por exemplo, se você tentar empacotar um arquivo SWF10 (Flash Player 10) como conteúdo inicial de um aplicativo do AIR 1.1, gerará esse erro.
306	Perfil não suportado.	O perfil que você está especificando no descritor do aplicativo não é suportado. Consulte "supportedProfiles" na página 243.
307	O namespace deve ser pelo menos *nnn*.	Use o namespace adequado para os recursos usados no aplicativo (como o namespace 2.0).

Códigos de saída de outros erros

Código de saída	Descrição	Observações
2	Erro de uso	Verifique se há erros nos argumentos da linha de comando.
5	Erro desconhecido	Esse erro indica que não é possível explicar uma situação com condições de erro comuns. As possíveis causas raiz incluem incompatibilidade entre o ADT e o Java Runtime Environment, instalações de ADT ou JRE corrompidas e erros de programação dentro do ADT.
6	Não foi possível gravar no diretório de saída	Verifique se o diretório de saída especificado (ou implícito) está acessível e se a unidade que o contém tem espaço em disco suficiente.
7	Não foi possível acessar o certificado	Verifique se o caminho para o armazenamento de chaves está especificado corretamente. Verifique se o certificado do armazenamento de chaves pode ser acessado. O utilitário Java 1.6 Keytool pode ser usado para ajudar a solucionar problemas de acesso ao certificado.
8	Certificado inválido	O arquivo de certificado foi malformado, modificado, expirado ou revogado.
9	Não foi possível assinar o arquivo AIR	Verifique as opções de assinaturas enviadas para o ADT.
10	Não foi possível criar um carimbo de data e hora	O ADT não estabeleceu uma conexão com o servidor do carimbo de data e hora. Se você se conecta à Internet por meio de um servidor proxy, pode precisar definir as configurações de proxy do JRE.
11	Erro de criação do certificado	Verifique os argumentos de linha de comando usados para criar assinaturas.

Código de saída	Descrição	Observações
12	Entrada inválida	Verifique os caminhos e outros argumentos enviados para o ADT na linha de comando.
13	SDK de dispositivo ausente	Verifique a configuração do SDK do dispositivo. O ADT não consegue localizar o SDK do dispositivo necessário para executar o comando especificado.
14	Erro de dispositivo	O ADT não pode executar o comando por causa de um problema ou restrição de dispositivo. Por exemplo, este código de saída é emitido ao tentar desinstalar um aplicativo que não esteja realmente instalado.
15	Nenhum dispositivo	Verifique se um dispositivo está conectado e ativado ou se um emulador está em execução.
16	Componentes GPL ausentes	O AIR SDK atual não inclui todos os componentes necessários para executar a operação de solicitação.
17	Falha na ferramenta de compactação do dispositivo.	Não foi possível criar o pacote porque alguns componentes do sistema operacional esperados estão ausentes.

Erros do Android

Código de saída	Descrição	Observações
400	A versão Android sdk atual não oferece suporte para o atributo.	Verifique se o nome do atributo foi escrito corretamente e se é um atributo válido para o elemento no qual ele aparece. Você pode precisar configurar o sinalizador -platformsdk no comando do ADT se o atributo foi introduzido após o Android 2.2.
401	A versão Android sdk atual não oferece suporte para o valor do atributo.	Verifique se o valor do atributo está escrito corretamente e se é um valor válido para o atributo. Você pode precisar configurar o sinalizador -platformsdk no comando do ADT se o valor do atributo foi introduzido após o Android 2.2.
402	A versão Android sdk atual não oferece suporte para a marca XML.	Verifique se o nome da marca XML está correta e o elemento do documento de manifesto do Android é válido. Você pode precisar configurar o sinalizador -platformsdk no comando do ADT se o elemento foi introduzido após o Android 2.2.
403	A marca Android não tem permissão para ser substituída	O aplicativo está tentando substituir um elemento do manifesto do Android que está reservado para uso pelo AIR. Consulte "Configurações do Android" na página 76.

Código de saída	Descrição	Observações
404	O atributo do Android não tem permissão para ser substituído	O aplicativo está tentando substituir um atributo do manifesto do Android que está reservado para uso pelo AIR. Consulte "Configurações do Android" na página 76.
405	A marca %1 do Android deverá ser o primeiro elemento na marca manifestAdditions	Mude a marca especificada para o local exigido.
406	O atributo %1 da marca %2 do Android possui um valor de %3 inválido.	Forneça um valor válido para o atributo.

Variáveis de ambiente ADT

O ADT lê os valores das seguintes variáveis do ambiente (se estiverem definidos):

AIR_ANDROID_SDK_HOME especifica o caminho para o diretório raiz do Android SDK (o diretório que contém a pasta de ferramentas). O AIR 2.6 SDK inclui as ferramentas do Android SDK necessárias para implementar os comandos relevantes do ADT. Somente defina este valor para usar uma versão diferente do Android SDK. Se esta variável estiver definida, a opção -platformsdk não precisa ser especificada ao executar os comandos do ADT que a exigem. Se esta variável e a opção de linha de comando estiverem definidas, o caminho especificado na linha de comando é usado.

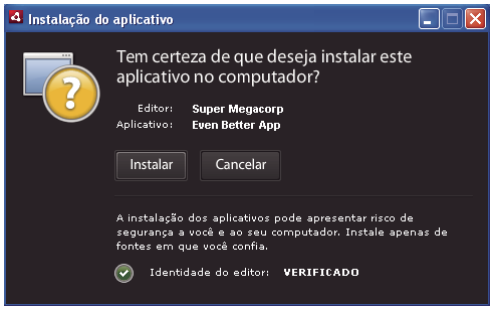
AIR_EXTENSION_PATH especifica uma lista de diretórios para buscar extensões nativas exigidas por um aplicativo. A lista de diretórios é pesquisada em ordem após quaisquer diretórios de extensão nativa especificados na linha de comando da ADT. O comando ADL também utiliza essa variável de ambiente.

Nota: *Em alguns sistemas de computadores, caracteres de bytes duplos nos caminhos do sistema de arquivos armazenados nessas variáveis de ambiente podem ser mal-interpretados. Se isto acontecer, tente definir o JRE usado para executar o ADT para usar o conjunto de caracteres UTF-8. Isso é feito por padrão no script usado para iniciar o ADT no Mac e no Linux. No arquivo adt.bat do Windows, ou ao executar o ADT diretamente do Java, especifique a opção* `-Dfile.encoding=UTF-8` *na linha de comando Java.*

Capítulo 13: Assinatura de aplicativos AIR

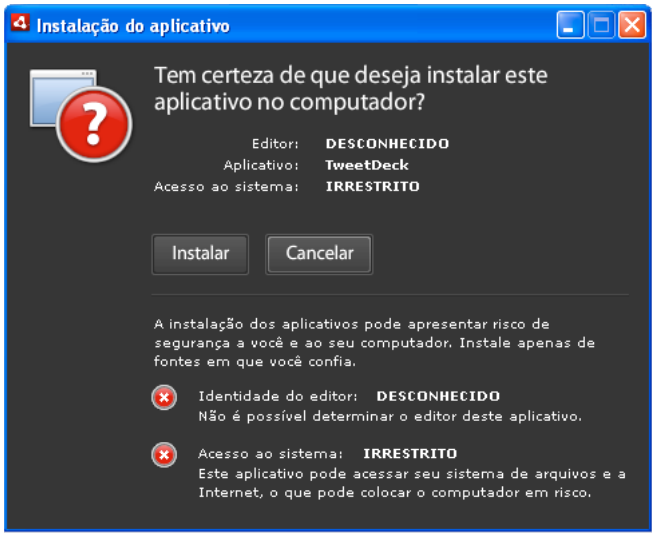
Assinatura digital de um arquivo AIR

Assinar digitalmente seus arquivos de instalação do AIR com um certificado emitido por uma autoridade de certificação reconhecida (CA) fornece uma garantia significativa aos seus usuários de que o aplicativo que estão instalando não foi alterado de modo acidental ou mal-intencionado e o identifica como o signatário (editor). O AIR exibe o nome do editor durante a instalação quando o aplicativo do AIR tiver sido assinado com um certificado confiável ou que esteja *vinculado* a um certificado confiável no computador de instalação:



Caixa de diálogo da confirmação da instalação assinada por certificado confiável

Se você assinar um aplicativo com um certificado autoassinado (ou um certificado que não esteja ligado a um certificado confiável), o usuário deverá assumir um risco de segurança maior ao instalar o seu aplicativo. As caixas de diálogo de instalação refletem este risco adicional:



A caixa de diálogo da confirmação da instalação do aplicativo, assinada por certificado autoassinado.

***Importante:** Uma entidade mal-intencionada poderia falsificar um arquivo AIR com sua identidade se ela, de alguma forma, obtiver seu arquivo de armazenamento de chaves de assinatura ou descobrir sua chave privada.*

Certificados de assinatura de código

As garantias de segurança, limitações e obrigações legais que envolvem o uso de certificados de assinatura de código são descritas nas Declarações de Práticas de Certificação (CPS) e nos contratos de assinatura publicados pela autoridade de certificação emissora. Para obter mais informações sobre os contratos das autoridades de certificação que emitem atualmente certificados de assinatura de código do AIR, consulte:

ChosenSecurity (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

ChosenSecurity CPS (http://www.chosensecurity.com/resource_center/repository.htm)

GlobalSign (http://www.globalsign.com/code-signing/index.html)

CPS da GlobalSign (http://www.globalsign.com/repository/index.htm)

CPS da Thawte (http://www.thawte.com/cps/index.html)

VeriSign CPS (http://www.verisign.com/repository/CPS/)

Contrato de assinante do VeriSign (https://www.verisign.com/repository/subscriber/SUBAGR.html)

Sobre a assinatura de código do AIR

Quando um arquivo AIR é assinado, uma assinatura digital é incluída no arquivo de instalação. A assinatura inclui uma compilação do pacote, usada para verificar se o arquivo AIR não foi alterado desde que foi assinado e se ele inclui informações sobre o certificado de assinatura, usado para verificar a identidade do editor.

O AIR usa a infraestrutura de chave pública (PKI) suportada pelo armazenamento de certificados do sistema operacional para estabelecer se um certificado pode ser confiável. O computador no qual um aplicativo do AIR está instalado deve confiar diretamente no certificado usado para assinar o aplicativo do AIR ou deve confiar em uma cadeia de certificados que vincula o certificado a uma autoridade de certificação confiável para que as informações do editor sejam verificadas.

Se um arquivo AIR for assinado com um certificado que não vincula a nenhum dos certificados raiz confiáveis (e normalmente isso inclui todos os certificados autoassinados), as informações do editor não podem ser verificadas. Embora o AIR possa determinar que o pacote do AIR não foi alterado desde que ele foi assinado, não há como saber quem realmente criou e assinou o arquivo.

Nota: *Um usuário pode optar por confiar em um certificado autoassinado e, em seguida, qualquer aplicativo do AIR assinado com o certificado exibirá o valor do campo de nome comum no certificado como o nome do editor. O AIR não fornece nenhum meio de um usuário designar um certificado como confiável. O certificado (não incluindo a chave privada) deve ser fornecido ao usuário separadamente e o usuário deve usar um dos mecanismos fornecidos pelo sistema operacional ou uma ferramenta apropriada para importar o certificado no local apropriado no armazenamento de certificados do sistema.*

Sobre identificadores de editor do AIR

Importante: *A partir do AIR 1.5.3 o ID do editor não é mais utilizado e não possui mais base calculada no certificado de assinatura de código. Novos aplicativos não precisam utilizar um ID do editor. Ao atualizar aplicativos existente, é necessário especificar o ID do editor original no arquivo descritor do aplicativo.*

Antes do AIR 1.5.3, o instalador de aplicativo do AIR gerava um ID do editor durante a instalação do arquivo AIR. Isto era um identificador único do certificado usado para criar o arquivo AIR. Se você reutilizou o mesmo certificado para vários aplicativos do AIR, eles receberam o mesmo ID do editor. A assinatura de uma atualização de aplicativo com um certificado diferente e até mesmo uma instância renovada do certificado original alterava o ID do editor.

No AIR 1.5.3 e posterior, o ID do editor não é mais atribuído pelo AIR. Um aplicativo publicado com o AIR 1.5.3 pode especificar uma cadeia de caracteres de ID do editor no descritor do aplicativo. Você somente deve especificar um ID do editor ao publicar atualizações de aplicativos originalmente publicados em versões do AIR anteriores à 1.5.3. Se você não especificar o ID original no descritor do aplicativo, o novo pacote AIR não será tratado como uma atualização de um aplicativo existente.

Para descobrir o ID do editor original, localize o arquivo`publisherid` no subdiretório META-INF/AIR onde o aplicativo original está instalado. A sequência de caracteres no arquivo é o ID do editor. O descritor o aplicativo deve especificar o runtime AIR 1.5.3 (ou posterior) na declaração de espaço de nome do arquivo descritor do aplicativo para especificar o ID do editor manualmente.

O ID do editor, quando existir, é utilizado para os seguintes fins:

- Como parte da chave de criptografia o armazenamento local criptografado
- Como parte o caminho para o diretório de armazenamento do aplicativo
- Como parte da sequência de caracteres de conexão para conexões locais
- Como parte da sequência de caracteres de identidade utilizada para chamar o aplicativo com a API interna de navegador AIR
- Como parte o OSID (utilizado na criação de programas personalizados de instalação/desinstalação)

Quando um ID de editor muda, o comportamento de qualquer recurso AIR dependente do ID também muda. Por exemplo, os dados existentes no armazenamento local criptografado não podem mais ser acessados e qualquer instância do Flash ou AIR que cria uma conexão local para o aplicativo deve utilizar o novo ID na sequência de caracteres de conexão. O ID do editor de um aplicativo instalado não pode ser alterado no AIR 1.5.3 ou posterior. Se você utilizar um ID do editor diferente ao publicar um pacote AIR, o instalador tratará o novo pacote como um aplicativo diferente, em vez de uma instalação.

Sobre formatos de certificados

As ferramentas de assinatura do AIR aceitam qualquer armazenamento de chave acessível pela JCA (arquitetura de criptografia Java). Isso inclui armazenamentos de chaves baseados em arquivos como arquivos de formatos PKCS12 (que normalmente usam uma extensão de arquivo .pfx ou .p12), arquivos .keystore Java, armazenamentos de chaves de hardware PKCS11 e armazenamentos de chaves de sistema. Os formatos de armazenamento de chave que o ADT pode acessar dependem da versão e configuração do runtime Java usado para executar o ADT. Acessar alguns tipos de armazenamento de chave, como tokens de hardware PKCS11, pode exigir a instalação e configuração de drivers de software adicionais e plug-ins de JCA.

Para assinar arquivos do AIR, você pode usar a maioria dos certificados de assinatura de código existentes ou obter um novo, emitido expressamente para assinar aplicativos do AIR. Por exemplo, qualquer um dos seguintes tipos de certificado da VeriSign, Thawte, GlobalSign ou ChosenSecurity podem ser usados:

- ChosenSecurity
  - ID de editor TC para Adobe AIR
- GlobalSign
  - Certificado de assinatura de código ObjectSign
- Thawte:
  - Certificado do desenvolvedor do AIR
  - Certificado do desenvolvedor Apple
  - Certificado do desenvolvedor JavaSoft

- Certificado Microsoft Authenticode

- VeriSign:
  - ID digital do Adobe AIR
  - ID digital Microsoft Authenticode
  - ID digital de assinatura Sun Java

Nota: *O certificado deve ser criado para assinatura de código. Você não pode usar um SSL ou outro tipo de certificado para assinar arquivos do AIR.*

Carimbos de data/hora

Quando você assina um arquivo AIR, a ferramenta de empacotamento consulta o servidor de uma autoridade de carimbo de data/hora para obter uma data e hora da assinatura independentemente verificáveis. a marca de data/hora está incorporado no arquivo AIR. Desde que o certificado de assinatura seja válido no momento da assinatura, o arquivo AIR poderá ser instalado, mesmo depois que o certificado expirar. Por outro lado, se nenhum carimbo de data/hora for obtido, o arquivo AIR não poderá mais ser instalado quando o certificado expirar ou for revogado.

Por padrão, as ferramentas de empacotamento do AIR obtêm um carimbo de data/hora. No entanto, para permitir que aplicativos sejam empacotados quando o serviço de carimbo de data/hora estiver indisponível, você pode desativar o recurso de carimbo de data/hora. A Adobe recomenda que todos os arquivos do AIR distribuídos publicamente incluam um carimbo de data/hora.

A autoridade padrão de carimbo de data/hora usada pelas ferramentas de empacotamento do AIR é Geotrust.

Obtenção de um certificado

Para obter um certificado, você normalmente visitaria o site da autoridade de certificação na Web e completaria o processo de obtenção da empresa. As ferramentas usadas para produzir o arquivo de armazenamento de chave necessário pelas ferramentas do AIR dependem do tipo de certificado adquirido, de como o certificado é armazenado no computador recebedor e, em alguns casos, o navegador usado para obter o certificado. Por exemplo, para obter e exportar um certificado do Adobe Developer do Thawte, você deve usar o Mozilla Firefox. O certificado pode então ser exportado como um arquivo .12 diretamente da interface do usuário do Firefox.

Nota: *Versões do Java 1.5 e acima não aceitam caracteres ASCII superiores nas senhas usadas para proteger arquivos de certificado PKCS12. O Java é usado pelas ferramentas de desenvolvimento do AIR para criar os pacotes assinados do AIR. Quando você exportar o certificado como um arquivo .p12 ou .pfx, use somente caracteres ASCII normais na senha.*

Você pode gerar um certificado autoassinado usando a ferramenta para desenvolvedores do AIR (ADT) usada para empacotar arquivos de instalação do AIR. Algumas ferramentas de terceiros também podem ser usadas.

Para obter instruções como gerar um certificado autoassinado, bem como instruções sobre como assinar um arquivo do AIR, consulte "AIR Developer Tool (ADT)" na página 168. Você também pode exportar e assinar arquivos do AIR usando o Flash Builder, Dreamweaver e a atualização do AIR para o Flash.

O exemplo a seguir descreve como obter um Certificado de desenvolvedor do AIR da autoridade de certificação Thawte e prepará-lo para o uso com o ADT.

Exemplo: Obtenção de um certificado do desenvolvedor do AIR da Thawte

Nota: *Este exemplo ilustra apenas uma das várias maneiras de se obter e preparar um certificado de assinatura do código para o uso. Cada autoridade de certificação tem suas próprias políticas e procedimentos.*

Para adquirir um certificado do desenvolvedor do AIR, o site da Thawte na Web requer que você use o navegador Mozilla Firefox. A chave privada para o certificado é armazenada no armazenamento de chave do navegador. Verifique se o armazenamento de chave do Firefox é protegido por uma senha mestre e se o computador em si é fisicamente seguro. (Você pode exportar e remover o certificado e a chave privada do armazenamento de chave do navegador quando o processo de obtenção estiver concluído.)

Como parte do processo de inscrição do certificado, é gerado um par de chave privada/pública. A chave privada é armazenada automaticamente no armazenamento de chave do Firefox. Você deve usar o mesmo computador e navegador para solicitar e recuperar o certificado do site da Thawte na Web.

1. Visite o site da Thawte na Web e navegue até Página de produtos para certificados de assinatura de código.
2. Da lista de certificados de assinatura de código, selecione o certificado do desenvolvedor do Adobe AIR.
3. Complete o processo de inscrição de três etapas. Você precisa fornecer informações organizacionais e de contato. A Thawte executa então seu processo de verificação de identidade e pode solicitar informações adicionais. Após a conclusão da verificação, a Thawte enviará um email com instruções sobre como recuperar o certificado.

   Nota: *informações adicionais sobre o tipo de documentação necessária podem ser encontradas aqui: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.*
4. Recupere o certificado emitido do site da Thawte. O certificado é salvo automaticamente no armazenamento de chave do Firefox.
5. Exporte um arquivo de armazenamento de chave contendo a chave privada e o certificado do armazenamento de chave do Firefox usando as seguintes etapas:

   Nota: *Ao exportar a chave privada/certificado do Firefox, ele é exportado em um formato de .p12 (pfx) que o ADT, Flex, Flash e o Dreamweaver podem usar.*

   a. Abra a caixa de diálogo do *gerenciador de certificados* do Firefox:
   b. No Windows: abra Tools (Ferramentas) -> Options (Opções) -> Advanced (Avançadas) -> Encryption (Criptografia) -> View Certificates (Exibir certificados)
   c. No Mac OS: abra Firefox (Firefox) -> Preferences (Preferências) -> Advanced (Avançadas) -> Encryption (Criptografia) -> View Certificates (Exibir certificados)
   d. No Linux: abra Edit (Editar) -> Preferences (Preferências) -> Advanced (Avançadas) -> Encryption (Criptografia) -> View Certificates (Exibir certificados)
   e. Selecione o certificado de assinatura do código do Adobe AIR da lista de certificados e clique no botão **Backup**.
   f. Digite um nome de arquivo e a localização para a qual exportar o arquivo de armazenamento de chave e clique em **Save** (Salvar).
   g. Se estiver usando a senha mestre do Firefox, será solicitado que você digite sua senha para o dispositivo de segurança do software para exportar o arquivo. (Essa senha é usada apenas pelo Firefox.)
   h. Na caixa de diálogo *Choose a Certificate Backup Password* (Escolha uma senha de backup de certificado), crie uma senha para o arquivo de armazenamento de chave.

      Importante: *Essa senha protege o arquivo de armazenamento de chave e é necessária quando o arquivo é usado para assinar aplicativos do AIR. Uma senha segura deve ser escolhida.*
   i. Clique em OK. Você deve receber uma mensagem de senha de backup bem-sucedida. O arquivo de armazenamento de chave contendo a chave privada e o certificado é salvo com uma extensão de arquivo .p12 (no formato PKCS12)
6. Use o arquivo de armazenamento de chave exportado com o ADT, Flash Builder, Flash Professional ou Dreamweaver. A senha criada para o arquivo é necessária sempre que um aplicativo do AIR é assinado.

Importante: *A chave privada e o certificado ainda são armazenados no armazenamento de chave do Firefox. Enquanto isso permite que você exporte uma cópia adicional do arquivo de certificado, também fornece outro ponto de acesso que deve ser protegido para manter a segurança do seu certificado e da chave privada.*

Alteração de certificados

Em alguns casos, você deve alterar o certificado utilizado para assinar atualizações para seu aplicativo do AIR. Tais circunstâncias incluem:

- Renovando o certificado de assinatura original.
- Atualização de um certificado autoassinado para um certificado emitido por uma autoridade de certificação
- Alteração de um certificado autoassinado prestes a expirar para outro
- Alterar de um certificado comercial para outro, por exemplo, quando sua identidade corporativa for alterada

Para que o AIR reconheça um arquivo AIR como uma atualização, você deve assinar o arquivo AIR original e a atualização como mesmo certificado ou aplicar uma assinatura de migração de certificado à atualização. Uma assinatura de migração é a segunda assinatura aplicada ao pacote AIR de atualização, utilizando o certificado original. A assinatura de migração usa o certificado original para estabelecer que o signatário é o editor original do aplicativo.

Após o arquivo AIR com a assinatura de migração ser instalado, o novo certificado tornar-se o certificado primário. Atualizações subsequentes não requerem uma assinatura de migração. No entanto, você deve aplicar assinaturas de migração pelo máximo de tempo possível, para acomodar usuários que ignoram atualizações.

Importante: *Você deve alterar o certificado e aplicar uma assinatura de migração para a atualização com o certificado original, antes que este expire. Caso contrário, os usuários devem desinstalar a versão existente do aplicativo antes de instalar uma nova versão. Para o AIR 1.5.3 ou posterior, você pode aplicar uma assinatura de migração utilizando um certificado expirado dentro de um período de tolerância de 365 dias depois que ele expirar. Contudo, não é possível utilizar um certificado expirado para aplicar a assinatura de aplicativo principal.*

Para alterar os certificados:

1. Crie uma atualização para o seu aplicativo
2. Empacote e assine o arquivo de atualização do AIR com o **novo** certificado
3. Assine o arquivo AIR novamente com o certificado **original** (usando o comando `-migrate` do ADT)

Um arquivo AIR com uma assinatura de migração é, em outros aspectos, um arquivo AIR normal. Se o aplicativo é instalado em um sistema sem a versão original, o AIR instala a nova versão da maneira normal.

Nota: *Antes do AIR 1.5.3, a assinatura de um aplicativo do AIR com um certificado renovado nem sempre requeria uma assinatura de migração. A partir do AIR 1.5.3, uma assinatura de migração é obrigatória para certificados renovados.*

Para aplicar uma assinatura de migração, use o "Comando migrate do ADT" na página 176, como descrito em "Assinatura de uma versão atualizada de um aplicativo do AIR" na página 203.

Nota: *O comando migrate do ADT não pode ser usado com aplicativos AIR para desktop que incluem extensões nativas, pois estes são empacotados como os instaladores nativos, não como arquivos .air. Para alterar certificados para um aplicativo AIR para desktop que inclua uma extensão nativa, empacote o aplicativo usando o "Comando package do ADT" na página 169 com o sinalizador -migrate.*

Alterações de identidade de aplicativo

Antes do AIR 1.5.3, a identidade de um aplicativo do AIR era alterada quando uma atualização assinada com uma assinatura de migração era instalada. A alteração da identidade de um aplicativo possui diversas implicações, incluindo:

- A nova versão do aplicativo não pode acessar dados no armazenamento local criptografado existente.
- O local do diretório de armazenamento do aplicativo é alterado. Os dados no local antigo não são copiados para o novo diretório. (Mas o novo aplicativo pode localizar o diretório original com base no ID do editor antigo).
- O aplicativo não pode mais abrir conexões locais usando o ID do editor antigo.
- A sequência de caracteres de identidade utilizada para acessar o aplicativo a partir de uma página da web era alterada.
- O OSID do aplicativo era alterado. (O OSID é utilizado durante o desenvolvimento de programas personalizados de instalação/desinstalação).

Ao publicar uma atualização do AIR 1.5.3 ou posterior, a identidade do aplicativo não pode ser alterada. Os ID de editor e o aplicativo original devem ser especificados no descritor do aplicativo o arquivo AIR de atualização. De outra forma, o novo pacote não é reconhecido como uma atualização.

Nota: *Ao publicar um novo aplicativo do AIR com o AIR 1.5.3 ou superior, você não deve especificar um ID do editor.*

Terminologia

Esta seção fornece um glossário de um pouco da terminologia principal que você deve entender ao tomar decisões sobre como assinar seu aplicativo para distribuição pública.

Termo	Descrição
Autoridade de certificação (CA)	Uma entidade em uma rede de infraestrutura de chave pública que serve como um terceiro confiável e, por último, certifica a identidade do proprietário de uma chave pública. Uma CA normalmente emite certificados digitais, assinados por sua própria chave privada, para atestar que ela verificou a identidade do proprietário do certificado.
Declaração de Prática de Certificação (CPS)	Apresenta as práticas e políticas da autoridade de certificação em emitir e verificar certificados. A CPS é parte do contrato entre a CA e seus assinantes e terceiros. Ela também resume as políticas para verificação de identidade e o nível de garantias oferecidas pelos certificados que elas fornecem.
Lista de revogação de certificado (CRL)	Uma lista de certificados emitidos que foram revogados e nos quais não se deve mais confiar. O AIR verifica o CRL no momento em que um aplicativo do AIR é assinado e, se nenhum carimbo de data/hora estiver presente, novamente quando o aplicativo for instalado.
Cadeia de certificados	Uma cadeia de certificados é uma sequência de certificados na qual cada certificado da cadeia foi assinado pelo certificado seguinte.
Certificado digital	Um documento digital que contém informações sobre a identidade do proprietário, a chave pública do proprietário e a identidade do certificado em si. Um certificado emitido por uma autoridade de certificação é em si assinado por um certificado que pertence à CA emissora.
Assinatura digital	Uma mensagem criptografada ou uma compilação que pode apenas ser descriptografada com metade da chave pública de um par de chave pública-privada. Em uma PKI, uma assinatura digital contém um ou mais certificados digitais rastreáveis, por último, para a autoridade de certificação. Uma assinatura digital pode ser usada para validar que uma mensagem (ou arquivo do computador) não foi alterada desde que ela foi assinada (nos limites da garantia fornecida pelo algoritmo criptográfico usado) e, supondo que alguém confia na autoridade de certificação emissora, a identidade do signatário.
Armazenamento de chave	Um banco de dados contendo certificados digitais e, em alguns casos, as chaves privadas relacionadas.

Termo	Descrição
JCA (arquitetura de criptografia Java)	Uma arquitetura extensível para gerenciar e acessar armazenamentos de chaves. Consulte o Guia de referência da arquitetura de criptografia Java para obter mais informações.
PKCS n º 11	O padrão de interface de token criptográfico da RSA Laboratories. Um armazenamento de chave baseado em token de hardware.
PKCS n º 12	O padrão de sintaxe do Exchange de informações pessoais da RSA Laboratories. Um armazenamento de chave baseado em arquivo que normalmente contém uma chave privada e seu certificado digital associado.
Chave privada	A metade privada de um sistema criptográfico assimétrico de chave pública-privada de duas partes. A chave privada deve ser mantida em segredo e nunca deve ser transmitida pela rede. Mensagens assinadas digitalmente são criptografadas com a chave privada pelo signatário.
Chave pública	A metade pública de um sistema criptográfico assimétrico de chave pública-privada de duas partes. A chave pública está abertamente disponível e é usada para descriptografar mensagens criptografadas com a chave privada.
Infraestrutura de chave pública (PKI)	Um sistema de confiança no qual as autoridades de certificação atestam para a identidade dos proprietários de chaves públicas. Clientes da rede confiam nos certificados digitais emitidos por uma CA confiável para verificar a identidade do signatário de uma mensagem digital (ou arquivo).
Carimbo de data/hora	Um dado assinado digitalmente contendo a data e hora em que um evento ocorreu. O ADT pode incluir um carimbo de data/hora de um servidor compatível com hora RFC 3161 em um pacote do AIR. Quando presente, o AIR usa o carimbo de data/hora para estabelecer a validade de um certificado no momento da assinatura. Isso permite que um aplicativo do AIR seja instalado após seu certificado de assinatura ter expirado.
Autoridade de carimbo de data/hora	Uma autoridade que emite carimbos de data/hora. Para ser reconhecido pelo AIR, o carimbo de data/hora deve estar em conformidade com RFC 3161 e a assinatura de carimbo de data/hora deve ser vinculada a um certificado raiz confiável na máquina de instalação.

Certificados do iOS

Os certificados de assinatura de código emitidos pela Apple são usados para aplicações de assinatura do iOS, incluindo aqueles desenvolvidos com o Adobe AIR. É necessária a aplicação de uma assinatura com um certificado de desenvolvimento da Apple para instalar uma aplicação em dispositivos de teste. É necessária a aplicação de uma assinatura com um certificado de distribuição para distribuir o aplicativo concluído.

Para assinar um aplicativo, o ADT requer acesso ao certificado de assinatura do código e à chave privada associada. O arquivo de certificado, por si só, não inclui a chave privada Você deve criar um armazenamento de chave sob a forma de arquivo .P12 ou .pfx (Personal Information Exchange) que contém o certificado e a chave privada. Consulte “Converter um certificado de desenvolvedor em um arquivo de armazenamento de chave P12” na página 201.

Gerar uma solicitação de assinatura de certificado

Para obter um certificado de desenvolvedor gere um arquivo de solicitação de assinatura de certificado, que será enviado para o Portal de aprovisionamento Apple iOS.

O processo de solicitação de assinatura de certificado gera um par de chaves pública-privada. A chave privada permanece no seu computador. Você envia a solicitação de assinatura que contém a chave pública e seus dados de identificação para a Apple, que atua no papel de autoridade de certificação. A Apple assina o certificado com seu próprio certificado WWDR (World Wide Developer Relations).

Gerar a solicitação de assinatura de certificado no Mac OS

No Mac OS, é possível utilizar o aplicativo Acesso Porta-chaves para gerar uma solicitação de assinatura de código. O aplicativo Acesso ao Porta-chaves está no subdiretório "Utilitários" do diretório "Aplicações". As instruções para gerar a solicitação de assinatura de certificado estão disponíveis no Portal de aprovisionamento Apple iOS.

Gerar a solicitação de assinatura de certificado no Windows

Para desenvolvedores do Windows, talvez seja mais fácil obter o certificado de desenvolvedor de iPhone em um computador Mac. No entanto, é possível obter o certificado em um computador Windows. Primeiro, crie o arquivo CSR (Certificate Signing Request) usando o OpenSSL:

1. Instale o OpenSSL em seu computador Windows. (Vá para http://www.openssl.org/related/binaries.html).

   Você deve instalar os arquivos do Visual C++ 2008 Redistributable, indicados na página de download do Open SSL. (A instalação do Visual C++ no computador *não* é necessária).

2. Abra a sessão de comandos do Windows e CD para o diretório OpenSSL bin (por exemplo, c:\OpenSSL\bin\).

3. Crie a chave particular digitando as informações abaixo na linha de comando:

   ```
   openssl genrsa -out mykey.key 2048
   ```

   Salve o arquivo de chave particular. O arquivo será utilizado posteriormente.

   Não ignore as mensagens de erro ao utilizar o OpenSSL. Mesmo que o OpenSSL gere uma mensagem de erro, ele ainda pode gerar os arquivos. No entanto, estes arquivos podem não ser utilizáveis. Se ocorrerem erros, verifique a sintaxe e execute o comando novamente.

4. Crie o arquivo CSR digitando as informações abaixo na linha de comando:

   ```
   openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest  -subj
   "/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
   ```

   Substitua os valores de endereço de email, CN (nome de certificado) e C (país) pelos seus valores.

5. Faça o upload do arquivo CSR para a Apple no site de desenvolvedor iPhone. (Consulte “Registrar-se para obter um certificado de desenvolvedor iPhone e criar um arquivo de aprovisionamento”.)

Converter um certificado de desenvolvedor em um arquivo de armazenamento de chave P12

Para criar um armazenamento de chave P12, você deve combinar o seu certificado de desenvolvedor da Apple e a chave privada associada em um único arquivo. O processo para criar o arquivo de armazenamento de chave depende do método usado para gerar a solicitação de assinatura do certificado original e onde a chave privada é armazenada.

Convertendo o certificado de desenvolvedor de iPhone em um arquivo P12 no Mac OS

Após baixar o certificado da Apple iPhone, exporte-o para o formato de armazenamento de chave P12. Para executar isso no Mac® OS:

1. Abra o aplicativo Acesso ao Porta-chaves (na pasta "Aplicações/Utilitários").
2. Se ainda não adicionou o certificado às Chaves, selecione Ficheiro > Importar elementos... Navegue até o diretório do arquivo de certificado (arquivo .cer) obtido da Apple.
3. Selecione a categoria "Chaves" no Acesso ao Porta-chaves.
4. Selecione a chave particular associada ao seu certificado de desenvolvimento de iPhone.

   O desenvolvedor de iPhone identifica a chave particular: Certificado público <nome> <sobrenome> com o qual coincide.

5. Clique com a tecla Command no certificado de desenvolvedor do iPhone e selecione *Exportar "iPhone Developer: Name..."*.
6. Salve o armazenamento de chave no formato de arquivo .p12 (Personal Information Exchange).

7 Será solicitado que você crie uma senha usada ao utilizar o armazenamento de chave para assinar aplicativos ou para transferir a chave e o certificado neste armazenamento de chave para outro.

Converter um certificado de desenvolvedor da Apple em um arquivo P12 no Windows

Para desenvolver aplicativos AIR for iOS, você deve usar um arquivo de certificado P12. Crie esse certificado com base no arquivo de certificado de desenvolvedor de iPhone da Apple que você recebe da Apple.

1 Converta o arquivo de certificado de desenvolvedor que recebe da Apple em um arquivo de certificado PEM. Digite o seguinte comando na linha de comando do diretório "bin" do OpenSSL:

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```

2 Se estiver usando a chave particular de um keychain em um computador Mac, converta-a em uma chave PEM.

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```

3 É possível criar um arquivo P12 válido, com base na chave e na versão PEM do certificado de desenvolvedor de iPhone:

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

Se estiver usando a chave do Mac OS Keychain, use a versão PEM que criou na etapa anterior. Do contrário, use a chave do OpenSSL que criou anteriormente (no Windows).

Criação de um arquivo intermediário do AIR não assinado com o ADT

Use o comando `-prepare` para criar um arquivo intermediário do AIR não assinado. Um arquivo intermediário do AIR deve ser assinado com o comando do ADT `-sign` para produzir um arquivo de instalação do AIR válido.

O comando `-prepare` emprega os mesmos sinalizadores e parâmetros do comando `-package` (exceto para as opções de assinatura). A única diferença é que o arquivo de saída não é assinado. O arquivo intermediário é gerado com a extensão de nome de arquivo: `airi`.

Para assinar um arquivo intermediário do AIR, use o comando do ADT `-sign`. (Consulte "Comando prepare do ADT" na página 175).

Exemplo do comando -prepare do ADT

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Assinatura de um arquivo intermediário do AIR com o ADT

Para assinar um arquivo intermediário do AIR com o ADT, use o comando `-sign`. O comando sign funciona apenas com arquivos intermediários do AIR (extensão `airi`). Um arquivo AIR não pode ser assinado uma segunda vez.

Para criar um arquivo intermediário do AIR, use o comando do ADT `-prepare`. (Consulte "Comando prepare do ADT" na página 175).

Assinatura de um arquivo AIRI

❖ Use o comando do ADT `-sign` com a seguinte sintaxe:

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS As opções de assinatura identificam a chave privada e o certificado com o qual o arquivo AIR será assinado. Essas opções são descritas em "Opções de assinatura de código ADT" na página 182.

airi_file O caminho para o arquivo intermediário não assinado do AIR a ser assinado.

air_file O nome do arquivo AIR a ser criado.

Exemplo do comando -sign do ADT

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Para obter mais informações, consulte "Comando sign do ADT" na página 176.

Assinatura de uma versão atualizada de um aplicativo do AIR

A cada vez que você criar uma versão atualizada de um aplicativo AIR existente, o aplicativo atualizado é assinado. Na melhor das hipóteses, você pode assinar a versão atualizada com o mesmo certificado usado para assinar a versão anterior. Neste caso, a assinatura é exatamente a mesma usada no aplicativo pela primeira vez.

Se o certificado usado para assinar a versão anterior do aplicativo expirou e foi renovado ou substituído, é possível usar o certificado renovado ou novo (substituição) para assinar a versão atualizada. Para isto, assine o aplicativo com o novo certificado *e* aplique uma assinatura de migração usando o certificado original. A assinatura de migração confirma que o proprietário do certificado original publicou a atualização.

Antes de aplicar uma assinatura de migração, considere os seguintes pontos:

- Para aplicar uma assinatura de migração, o certificado original deve ser válido ou estar expirado por no máximo 365 dias. Esse período é denominado como o "período de prorrogação" e a duração pode mudar no futuro.

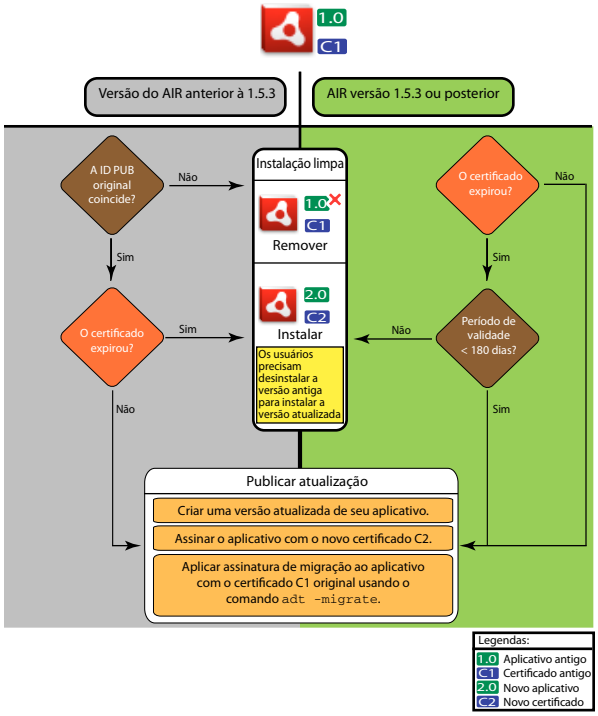
  Nota: *Até o AIR 2.6, o período de prorrogação era de 180 dias.*

- Você não pode aplicar uma assinatura de migração após o certificado expirar e transcorrerem os 365 dias do período de prorrogação. Neste caso, os usuários devem desinstalar a versão existente antes da instalação da versão atualizada.
- O período de tolerância de 365 dias somente se aplica a aplicativos que especificam o AIR versão 1.5.3 ou superior no namespace do descritor do aplicativo.

Importante: *As atualizações de assinatura com assinaturas de migração de certificados expirados é uma solução temporária. Para uma solução global, crie um fluxo de trabalho de assinatura padronizada para gerenciar a implementação de atualizações de aplicativos. Por exemplo, assine cada atualização com o certificado mais recente e aplique um certificado de migração utilizando o certificado usado para assinar a atualização anterior (se aplicável). Carregue cada atualização em seu próprio URL para que os usuários possam baixar o aplicativo. Para obter mais informações, consulte "Marcando o fluxo de trabalho para atualizações do aplicativo" na página 264.*

A tabela e a figura a seguir resumem o fluxo de trabalho para assinaturas de migração:

Cenário	Estado do certificado original	Ação do desenvolvedor	Ação do usuário
Aplicativo com base em runtime do Adobe AIR versão 1.5.3 ou superior	Válido	Publicar a versão mais recente do aplicativo do AIR	Nenhuma ação necessária Aplicativo atualizado automaticamente
	Expirou, mas está no período de tolerância de 365 dias	Assine o aplicativo com o novo certificado. Aplique uma assinatura de migração usando o certificado expirado.	Nenhuma ação necessária Aplicativo atualizado automaticamente
	Expirado e fora do período de prorrogação	Você não pode aplicar a assinatura de migração para a atualização do aplicativo do AIR. Em vez disso, você deve publicar uma outra versão do aplicativo do AIR usando um novo certificado. Os usuários podem instalar a nova versão depois de desinstalar a existente do aplicativo do AIR.	Desinstale a versão atual do aplicativo do AIR e instale a versão mais recente
• Aplicativo com base em runtime do Adobe AIR versão 1.5.2 ou inferior • O ID de editor no descritor do aplicativo da atualização corresponde ao ID de editor da versão anterior	Válido	Publicar a versão mais recente do aplicativo do AIR	Nenhuma ação necessária Aplicativo atualizado automaticamente
	Expirado e fora do período de prorrogação	Você não pode aplicar a assinatura de migração para a atualização do aplicativo do AIR. Em vez disso, você deve publicar uma outra versão do aplicativo do AIR usando um novo certificado. Os usuários podem instalar a nova versão depois de desinstalar a existente do aplicativo do AIR.	Desinstale a versão atual do aplicativo do AIR e instale a versão mais recente
• Aplicativo com base em runtime do Adobe AIR versão 1.5.2 ou inferior • O ID de editor no descritor do aplicativo da atualização **não** corresponde ao ID de editor da versão anterior	Qualquer elemento	Assine o aplicativo AIR usando um certificado válido e publique a versão mais recente do aplicativo AIR	Desinstale a versão atual do aplicativo do AIR e instale a versão mais recente



Marcando o fluxo de trabalho para atualizações

Migração de um aplicativo do AIR para usar um novo certificado

Para migrar um aplicativo AIR para um novo certificado ao atualizar um aplicativo:

1. Crie uma atualização para o seu aplicativo
2. Empacote e assine o arquivo de atualização do AIR com o **novo** certificado
3. Assine o arquivo AIR novamente com o certificado **original** usando o comando `-migrate`

Um arquivo AIR assinado com o comando `-migrate` também pode ser usado para instalar uma nova versão do aplicativo, assim como ser usado para atualizar qualquer versão anterior assinada com o certificado antigo.

Nota: *Ao atualizar um aplicativo publicado para uma versão do AIR anterior a 1.5.3, especifique o ID de editor original no descritor do aplicativo. De outra forma, os usuários de seu aplicativo devem desinstalar a versão anterior, antes de instalar a atualização.*

Use o comando do ADT `-migrate` com a seguinte sintaxe:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** As opções de assinatura identificam a chave privada e o certificado com o qual o arquivo AIR será assinado. Essas opções devem identificar o certificado de assinatura **original** e são descritas em "Opções de assinatura de código ADT" na página 182.
- **air_file_in** O arquivo AIR para a atualização, assinado com o certificado **novo**.
- **air_file_out** O arquivo AIR a ser criado.

Nota: *Os nomes de arquivos usados para os arquivos do AIR de entrada e de saída devem ser diferentes.*

O exemplo a seguir demonstra uma chamada ADT com o sinalizador `-migrate` para aplicar uma assinatura de migração a uma versão atualizada de um aplicativo AIR:

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Nota: *O comando* `-migrate` *foi adicionado ao ADT na versão AIR 1.1.*

Migrar um instalador nativo do aplicativo AIR para usar um novo certificado

Um aplicativo AIR que é publicado como um instalador nativo (por exemplo, um aplicativo que usa a extensão da api nativa) não pode ser assinado usando um comando ADT `-migrate`, pois ele é um aplicativo de plataforma nativa específica, não um arquivo .air. Em vez disso, para migrar um aplicativo AIR que é publicado como uma extensão nativa para um novo certificado:

1. Crie uma atualização para o seu aplicativo.
2. Certifique-se de que a marca `<supportedProfiles>` do arquivo do descritor (app.xml) do seu aplicativo inclua os perfis desktop profile e extendedDesktop (ou remova a marca `<supportedProfiles>` do descritor do aplicativo).
3. Empacote e assine o aplicativo atualizado **como um arquivo .air** usando o comando ADT `-package` com o **novo** certificado.
4. Aplique o certificado de migração ao arquivo .air usando o comando ADT `-migrate` com o certificado **original** (como descrito anteriormente em "Migração de um aplicativo do AIR para usar um novo certificado" na página 205).
5. Empacote o arquivo .air em um instalador nativo usando o comando ADT `-package` com o sinalizador `-target native`. Como o aplicativo já está assinado, você não precisa especificar um certificado de assinatura como parte desta etapa.

O exemplo a seguir demonstra as etapas 3 a 5 deste processo. O código chama o ADT com os comandos `-package`, `-migrate` e novamente com o comando `-package` para empacotar uma versão atualizada do aplicativo AIR como um 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
```

Migrar um aplicativo AIR que usa uma extensão nativa para usar um novo certificado

Um aplicativo AIR que usa uma extensão nativa não pode ser assinado usando o comando ADT `-migrate`. Ele também não pode ser migrado usando o procedimento de migração de um aplicativo AIR com instalador nativo, pois ele não pode ser publicado como um arquivo .air intermediário. Em vez disso, para migrar um aplicativo AIR que usa uma extensão nativa para um novo certificado:

1. Crie uma atualização para o seu aplicativo
2. Empacote e assine o instalador nativo de atualização usando o comando ADT `-package`. Empacote o aplicativo com o **novo** certificado e inclua o sinalizador `-migrate` especificando o certificado **original**.

Use a sintaxe a seguir para chamar o comando ADT `-package` com o sinalizador `-migrate`:

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** As opções de assinatura identificam a chave e o certificado privados com os quais o arquivo AIR é assinado. Estas opções identificam o **novo** certificado de assinatura e são descritas nas "Opções de assinatura de código ADT" na página 182.

- **MIGRATION_SIGNING_OPTIONS** As opções de assinatura identificam a chave e o certificado privados com os quais o arquivo AIR é assinado. Estas opções identificam o certificado de assinatura **original** e são descritos nas "Opções de assinatura de código ADT" na página 182.
- As outras opções são as mesmas usadas para empacotar um instalador nativo do aplicativo AIR e são descritas no "Comando package do ADT" na página 169.

O exemplo a seguir demonstra a chamada do ADT com o comando `-package` e o sinalizador `-migrate` para empacotar uma versão atualizada de um aplicativo AIR e usa uma extensão nativa e aplicar uma assinatura de migração à atualização.

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore
original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Nota: *O sinalizador `-migrate` do comando `-package` está disponível no ADT no AIR 3.6 e posterior.*

Criação de um certificado autoassinado com o ADT

Você pode autoassinar certificados para produzir um arquivo de instalação do AIR válido. Entretanto, certificados autoassinados somente oferecem garantias de segurança limitadas aos usuários. A autenticidade dos certificados autoassinados não pode ser verificada. Quando um arquivo AIR autoassinado é instalado, as informações do editor são exibidas para o usuário como Desconhecidas. Um certificado gerado pelo ADT é válido por cinco anos.

Se você criar uma atualização para um aplicativo do AIR assinada com um certificado autogerado, deverá usar o mesmo certificado para assinar os originais e atualizar os arquivos do AIR. Os certificados que o ADT produz são sempre únicos, mesmo se os mesmos parâmetros são usados. Portanto, se você desejar autoassinar atualizações com um certificado gerado pelo ADT, preserve o certificado original em um local seguro. Além disso, você não poderá produzir um arquivo AIR atualizado depois que o certificado original gerado pelo ADT expirar. (Você pode publicar novos aplicativos com um certificado diferente, mas não novas versões do mesmo aplicativo.)

Importante: *Devido às limitações de certificados autoassinados, a Adobe recomenda altamente o uso de um certificado comercial emitido por uma autoridade de certificação de reputação, para assinar publicamente aplicativos do AIR lançados.*

O certificado e a chave privada associada gerados pelo ADT são armazenados em um arquivo de armazenamento de chaves do tipo PKCS12. A senha especificada é definida na chave em si, e não no armazenamento de chaves.

Exemplos de geração de certificado

```
adt -certificate -cn SelfSign -ou QE -o "Example, Co" -c US 2048-RSA newcert.p12 39#wnetx3tl
adt -certificate -cn ADigitalID 1024-RSA SigningCert.p12 39#wnetx3tl
```

Para usar esses certificados para assinar arquivos do AIR, você usa as seguintes opções de assinatura com os comandos `-package` ou `-prepare` do ADT:

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

Nota: *Versões do Java 1.5 e acima não aceitam caracteres ASCII superiores nas senhas usadas para proteger arquivos de certificado PKCS12. Use somente caracteres ASCII normais nas senhas.*

Capítulo 14: Arquivos descritores do aplicativo do AIR

Cada aplicativo do AIR requer um arquivo descritor do aplicativo. O arquivo descritor do aplicativo é um arquivo XML que define as propriedades básicas do aplicativo.

Muitos ambientes de desenvolvimento que suportam o AIR geram automaticamente um descritor de aplicativo quando você cria um projeto. Do contrário, você deve criar seu próprio arquivo descritor. Um arquivo descrito de amostra, `descriptor-sample.xml`, pode ser encontrado no diretório `samples` dos SDKs do AIR e do Flex.

Qualquer nome de arquivo pode ser usado pelo arquivo do descritor do aplicativo. Quando você empacota o aplicativo, o arquivo do descritor do aplicativo é renomeado para `application.xml` e colocado em um diretório especial dentro do pacote do AIR.

Descritor de aplicativo do exemplo

O documento descritor do aplicativo a seguir define as propriedades básicas utilizadas pela maioria dos aplicativos 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>
```

Se o aplicativo usa um arquivo HTML como seu conteúdo raiz em vez de um arquivo SWF, apenas o elemento `<content>` é diferente:

```
<content>
    HelloWorld.html
</content>
```

Mudanças no descritor do aplicativo

O descritor do aplicativo do AIR mudou nas versões seguintes do AIR.

Mudanças no descritor do AIR 1.1

Os elementos `name` e `description` do aplicativo são localizados no elemento `text` .

Mudanças no descritor do AIR 1.5

`contentType` tornou-se um filho necessário de `fileType`.

Mudanças no descritor AIR 1.5.3

Incluído o elemento `publisherID` para permitir que aplicativos especifiquem um valor de ID de editor.

Mudanças no descritor AIR 2.0

Adicionado:

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (consulte imageNxN)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

Mudanças no descritor AIR 2.5

Removida: `version`

Adicionado:

- `android`
- `extensionID`
- `extensions`
- `image36x36` (consulte imageNxN)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

Mudanças no descritor do AIR 2.6

Adicionado:

- `image114x114` (consulte `imageNxN`)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Mudanças no descritor do AIR 3.0

Adicionado:

- `colorDepth`
- *direct* como valor direto de `renderMode`
- renderMode não é mais ignorado em plataformas de área de trabalho
- O elemento `<uses-sdk>` do Android pode ser especificado. (Não era permitido anteriormente.)

Mudanças no descritor do AIR 3.1

Adicionado:

- "Entitlements" na página 222

Mudanças no descritor do AIR 3.2

Adicionado:

- `depthAndStencil`
- `supportedLanguages`

Mudanças no descritor do AIR 3.3

Adicionado:

- `aspectRatio` agora inclui a opção `ANY`.

Mudanças no descritor do AIR 3.4

Adicionado:

- `image50x50` (consulte "`imageNxN`" na página 230)
- `image58x58` (consulte "`imageNxN`" na página 230)
- `image100x100` (consulte "`imageNxN`" na página 230)
- `image1024x1024` (consulte "`imageNxN`" na página 230)

Mudanças no descritor do AIR 3.6

Adicionado:

- "requestedDisplayResolution" na página 240 no elemento "iPhone" na página 234 agora inclui um atributo `excludeDevices`, permitindo que você especifique quais destinos do iOS usam resolução alta ou padrão.

- novo elemento "requestedDisplayResolution" na página 240 em "initialWindow" na página 232 especifica quando usar a resolução alta ou padrão nas plataformas de desktop, como Macs com telas de alta resolução

Mudanças no descritor do AIR 3.7

Adicionado:

- O elemento "iPhone" na página 234 agora fornece um elemento "externalSwfs" na página 224, que permite que você especifique uma lista de SWFs a ser carregada no tempo de execução.
- O elemento "iPhone" na página 234 agora fornece um elemento"forceCPURenderModeForDevices" na página 227, que permite que você force o modo de renderização da CPU para um conjunto de dispositivos especificado.

A estrutura do arquivo do descritor do aplicativo

O arquivo descritor do aplicativo é um documento XML com a seguinte estrutura:

```
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <allowBrowserInvocation>...<allowBrowserInvocation>
    <android>
        <colorDepth>...</colorDepth>
        <manifestAdditions
                <manifesto>...</manifesto>
            ]]>
        </manifestAdditions
    </android>
    <copyright>...</copyright>
    customUpdateUI>...</
    <description>
        <text xml:lang="...">...</text>
    </description>
    <extensions>
        <extensionID>...</extensionID>
    </extensions>
    <filename>...</filename>
    <fileTypes>
        <fileType>
            <contentType>...</contentType>
            <description>...</description>
            <extension>...</extension>
            <icon>
                <imageNxN>...</imageNxN>
            </icon>
            <name>...</name>
        </fileType>
    </fileTypes>
    <icon>
        <imageNxN>...</imageNxN>
    </icon>
    <id>...</id>
    <initialWindow>
        <aspectRatio>...</aspectRatio>
        <autoOrients>...</autoOrients>
```

```
        <content>...</content>
        <depthAndStencil>...</depthAndStencil>
        <fullScreen>...</fullScreen>
        <height>...</height>
        <maximizable>...</maximizable>
        <maxSize>...</maxSize>
        <minimizable>...</minimizable>
        <minSize>...</minSize>
        <renderMode>...</renderMode>
        <requestedDisplayResolution>...</requestedDisplayResolution>
        <resizable>...</resizable>
        <softKeyboardBehavior>...</softKeyboardBehavior>
        <systemChrome>...</systemChrome>
        <title>...</title>
        <transparent>...</transparent>
        <visible>...</visible>
        <width>...</width>
        <x>...</x>
        <y>...</y>
    </initialWindow>
    <installFolder>...</installFolder>
    <iPhone>
        <Entitlements>...</Entitlements>
        <InfoAdditions>...</InfoAdditions>
        <requestedDisplayResolution>...</requestedDisplayResolution>
        <forceCPURenderModeForDevices>...</forceCPURenderModeForDevices>
        <externalSwfs>...</externalSwfs>
    </iPhone>
    <name>
        <text xml:lang="...">...</text>
    </name>
    <programMenuFolder>...</programMenuFolder>
    <publisherID>...</publisherID>
    <"supportedLanguages" na página 242>...</"supportedLanguages" na página 242>
    <supportedProfiles>...</supportedProfiles>
    <versionNumber>...</versionNumber>
    <versionLabel>...</versionLabel>
</application>
```

Elementos descritores do aplicativo do AIR

O dicionário de elementos a seguir descreve cada um dos elementos válidos de um arquivo descritor de aplicativo do AIR.

allowBrowserInvocation

Adobe AIR 1.0 e posterior - Opcional

Permite que a API no navegador AIR detecte e ative o aplicativo.

Se você definiu esse valor como `true`, certifique-se de considerar implicações de segurança. Elas são descritas em Invocação de um aplicativo do AIR do navegador (para desenvolvedores em ActionScript) ou em Invoking an AIR application from the browser (para desenvolvedores em HTML).

Para obter mais informações, consulte "Inicialização de um aplicativo do AIR instalado do navegador" na página 260.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo

`true` ou `false` (padrão)

Exemplo

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 e posterior - opcional

Permite adicionar elementos ao arquivo de manifesto do Android. O AIR cria o arquivo AndroidManifest.xml para cada pacote APK. Você pode usar o elemento android no descritor do aplicativo do AIR para adicionar outros itens adicionais a este. Ignorado em todas as plataformas, exceto no Android.

Elemento pai: "application" na página 213

Elementos filho:

- "colorDepth" na página 217
- "manifestAdditions" na página 235

Conteúdo

Elementos que definem as propriedades específicas do Android para adicionar o manifesto do aplicativo do Android.

Exemplo

```
<android>
    <manifestAdditions>
        ...
    </manifestAdditions>
</android>
```

Mais tópicos da Ajuda

"Configurações do Android" na página 76

O arquivo AndroidManifest.xml

application

Adobe AIR 1.0 e posterior - Necessário

O elemento raiz de um documento de descrição do aplicativo do AIR.

Elementos pai: nenhum

Elementos filho:

- "allowBrowserInvocation" na página 212
- "android" na página 213

Atributos

minimumPatchLevel — O nível mínimo de caminho do runtime AIR exigido por esta aplicação.

xmlns — o atributo de namespace XML determina a versão do runtime AIR necessária ao aplicativo.

O namespace é alterado com cada versão principal do AIR (mas não com patches secundários). O último segmento do espaço para nomes, como "3.0", indica a versão em runtime exigida pelo aplicativo.

Os valores xmlns para as principais versões AIR são:

```
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"
```

Nos aplicativos baseados em SWF, a versão do runtime do AIR especificada na descrição do aplicativo determina a versão máxima do SWF que pode ser carregada como o conteúdo inicial do aplicativo. Aplicativos que especificam AIR 1.0 ou AIR 1.1 só podem usar arquivos SWF9 (Flash Player 9) como conteúdo inicial, mesmo quando executados usando o runtime do AIR 2. Os aplicativos que especificam AIR 1.5 (ou posteriores) podem usar arquivos SWF9 ou SWF10 (Flash Player 10) como conteúdo inicial.

A versão do SWF determina que versão do AIR e APIs do Flash Player estão disponíveis. Se um arquivo SWF9 for usado como conteúdo inicial de um aplicativo do AIR 1.5, esse aplicativo só terá acesso ao AIR 1.1 e às APIs do Flash Player 9. Além disso, alterações de comportamento feitas em APIs existentes no AIR 2.0 ou no Flash Player 10.1 não serão eficazes. (Alterações importantes relacionadas à segurança, feitas em APIs, são uma exceção a esse princípio e podem ser aplicadas de forma retroativa em patches atuais ou futuros do runtime).

Em aplicativos baseados em HTML, a versão do runtime especificada na descrição do aplicativo determina qual versão do AIR e de APIs do Flash Player estão disponíveis para o aplicativo. Os comportamentos de HTML, CSS e JavaScript são sempre determinados pela versão do Webkit usada no runtime do AIR instalado, não pela descrição do aplicativo.

Quando um aplicativo do AIR carrega conteúdo SWF, a versão do AIR e das APIs do Flash Player disponíveis para esse conteúdo depende de como o conteúdo é carregado. Às vezes, a versão efetiva é determinada pelo espaço de nome do descritor do aplicativo, às vezes é determinada pela versão do conteúdo de carregamento e às vezes é determinada pela versão do conteúdo carregado. A tabela a seguir mostra como a versão da API é determinada com base no método de carregamento:

Como o conteúdo é carregado	**Como a versão da API é determinada**
Conteúdo inicial, aplicativo baseado em SWF	Versão SWF do arquivo **carregado**
Conteúdo inicial, aplicativo baseado em HTML	Espaço para nomes da descrição do aplicativo
SWF carregado pelo conteúdo SWF	Versão do conteúdo **carregado**
Biblioteca SWF carregada pelo conteúdo HTML usando a tag <script>	Espaço para nomes da descrição do aplicativo
SWF carregado pelo conteúdo HTML usando o AIR ou as APIs do Flash Player (como flash.display.Loader)	Espaço para nomes da descrição do aplicativo
SWF carregado pelo conteúdo HTML usando as tags <object> ou <embed> (ou as APIs de JavaScript equivalentes)	Versão SWF do arquivo **carregado**

Ao carregar um arquivo SWF de uma versão diferente do conteúdo carregado, você pode se deparar com dois problemas:

- Carregamento de uma versão mais recente de SWF por uma versão mais antiga de SWF— As referências a APIs adicionadas nas novas versões do AIR e do Flash Player no conteúdo carregado ficarão sem ser resolvidas.
- Carregamento de uma versão mais antiga de SWF por uma versão mais recente de SWF— As APIs alteradas nas novas mais recentes do AIR e do Flash Player poderão se comportar de forma inesperada pelo conteúdo carregado.

Conteúdo

O elemento do aplicativo contém elementos filho que definem as propriedades de um aplicativo do AIR.

Exemplo

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <id>HelloWorld</id>
    <version>2.0</version>
    <filename>Hello World</filename>
    <name>Example Co. AIR Hello World</name>
     <description>
        <text xml:lang="en">This is an example.</text>
        <text xml:lang="fr">C'est un exemple.</text>
        <text xml:lang="es">Esto es un ejemplo.</text>
    </description>
    <copyright>Copyright (c) 2010 Example Co.</copyright>
    <initialWindow>
        <title>Hello World</title>
        <content>
            HelloWorld.swf
        </content>
        <systemChrome>none</systemChrome>
        <transparent>true</transparent>
        <visible>true</visible>
        <minSize>320 240</minSize>
    </initialWindow>
    <installFolder>Example Co/Hello World</installFolder>
    <programMenuFolder>Example Co</programMenuFolder>
    <icon>
        <image16x16>icons/smallIcon.png</image16x16>
        <image32x32>icons/mediumIcon.png</image32x32>
        <image48x48>icons/bigIcon.png</image48x48>
        <image128x128>icons/biggestIcon.png</image128x128>
    </icon>
    <customUpdateUI>true</customUpdateUI>
    <allowBrowserInvocation>false</allowBrowserInvocation>
    <fileTypes>
        <fileType>
            <name>adobe.VideoFile</name>
            <extension>avf</extension>
            <description>Adobe Video File</description>
            <contentType>application/vnd.adobe.video-file</contentType>
            <icon>
                <image16x16>icons/avfIcon_16.png</image16x16>
                <image32x32>icons/avfIcon_32.png</image32x32>
                <image48x48>icons/avfIcon_48.png</image48x48>
                <image128x128>icons/avfIcon_128.png</image128x128>
            </icon>
        </fileType>
    </fileTypes>
</application>
```

aspectRatio

Adobe AIR 2.0 e posterior, iOS e Android — opcional

Especifica a taxa de proporção do aplicativo.

Se não for especificado, o aplicativo abrirá com a proporção "natural" e com a orientação do dispositivo. A orientação natural varia de dispositivo para dispositivo. Geralmente, é a proporção retrato em dispositivos de tela pequena tais como telefones. Em alguns dispositivos, como o tablete iPad, o aplicativo se abre na orientação atual. No AIR 3.3 e superior, isso se aplica a todo o aplicativo, não apenas à exibição inicial.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

`retrato, paisagem` ou `qualquer uma`

Exemplo

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

autoOrients

Adobe AIR 2.0 e posterior, iOS e Android — opcional

Especifica se a orientação do conteúdo no aplicativo reorienta automaticamente quando o próprio dispositivo altera a orientação física. Para obter mais informações, consulte Orientação do palco.

Ao usar auto-orientação, considere a configuração das propriedades `align` e `scaleMode` do Palco para o seguinte:

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Essas configurações permitem que o aplicativo gire em torno dos cantos superior e esquerdo e impedem que o conteúdo do seu aplicativo seja escalado automaticamente. Embora os outros modos de escala façam ajustes no conteúdo para adaptarem-se às dimensões do palco submetido à rotação, eles também recortam, distorcem ou comprimem excessivamente esse conteúdo. Melhores resultados quase sempre podem ser obtidos quando você mesmo redesenha ou retransmite o conteúdo.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

`true` ou `false` (padrão)

Exemplo

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

colorDepth

Adobe AIR 3 e posterior - Opcional

Especifica a utilização da cor de 16 bits ou de 32 bits.

A utilização da cor de 16 bits pode aumentar o desempenho de renderização, mas em detrimento da fidelidade de cor. Antes do AIR 3, a cor de 16 bits foi sempre usada no Android. No AIR 3, a cor de 32 bits é usada por padrão.

Nota: *Se o seu aplicativo usar a classe StageVideo, você deverá usar cor de 32 bits.*

Elemento pai:"android" na página 213

Elementos filho: nenhum

Conteúdo
Um dos seguintes valores:

- 16 bits
- 32 bits

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

containsVideo

Especifica se o aplicativo terá ou não um conteúdo de vídeo.

Elemento pai:"android" na página 213

Elementos filho: nenhum

Conteúdo
Um dos seguintes valores:

- true
- false

Exemplo
```
<android>
    <containsVideo>true</containsVideo>
    <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 e posterior - Necessário

O valor especificado para o elemento `content` é o URL para o arquivo principal de conteúdo do aplicativo. Isso pode ser um arquivo SWF ou HTML. O URL é especificado em relação à raiz da pasta de instalação do aplicativo. (Ao executar um aplicativo do AIR com o ADL, o URL é relativo à pasta que contém o arquivo de descritor do aplicativo. Você pode usar o parâmetro `root-dir` do ADL para especificar um diretório de raiz diferente.)

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Uma URL relativa ao diretório do aplicativo. Como o valor do elemento content é tratado como uma URL, os caracteres no nome do arquivo de conteúdo devem ser codificados por URL de acordo com as regras definidas em RFC 1738. Caracteres de espaço, por exemplo, devem ser codificados como `%20`.

Exemplo

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0 a 1.1 — opcional; AIR 1.5 e posterior — obrigatório

O `contentType` é exigido no AIR 1.5 (ele era opcional no AIR 1.0 e 1.1). A propriedade ajuda alguns sistemas operacionais a localizarem o melhor aplicativo para abrir um arquivo. O valor deve ser o tipo MIME do conteúdo do arquivo. Observe que o valor será ignorado no Linux se o tipo de arquivo já estiver registrado e tiver um tipo MIME atribuído.

Elemento pai:"fileType" na página 225

Elementos filho: nenhum

Conteúdo

O tipo e subtipo de MIME. Consulte RFC2045 para obter mais informações sobre tipos de MIME.

Exemplo

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 e posterior - Opcional

As informações de copyright para o aplicativo do AIR. No Mac OS, o texto de copyright é exibido na caixa de diálogo Sobre para o aplicativo instalado. No Mac OS, as informações de copyright também são usadas no campo NSHumanReadableCopyright no arquivo Info.plist para o aplicativo.

Elemento pai:"application" na página 213

Elementos filho: nenhum

Conteúdo

Uma sequência de caracteres que contém as informações de copyright do aplicativo.

Exemplo

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 e posterior - Opcional

Indica se um aplicativo fornecerá seus próprios diálogos de atualização. Se definido como `false`, o AIR apresenta os diálogos de atualização padrão para o usuário. Somente os aplicativos distribuídos como arquivos AIR podem usar o sistema incorporado de atualização do AIR.

Quando a versão instalada do aplicativo tiver o elemento `customUpdateUI` configurado para `true` e, em seguida, o usuário clicar duas vezes no arquivo AIR para obter uma nova versão ou instalar uma atualização do aplicativo usando o recurso de instalação contínua, o runtime abre a versão instalada do aplicativo. O runtime não abre o instalador do aplicativo padrão do AIR. A lógica do seu aplicativo pode então determinar como proceder com a operação de atualização. (O ID do aplicativo e o ID do editor no arquivo AIR devem corresponder aos valores no aplicativo instalado para que uma atualização prossiga.)

***Nota:** O mecanismo `customUpdateUI` apenas começa a funcionar quando o aplicativo já está instalado e o usuário clica duas vezes no arquivo de instalação do AIR que contém uma atualização ou instala uma atualização do aplicativo usando o recurso de instalação direta. Você pode baixar e iniciar uma atualização pela lógica do seu aplicativo, exibindo sua interface de usuário personalizada conforme necessário, seja `customUpdateUI`true ou não.*

Para obter mais informações, consulte "Atualização de aplicativos do AIR" na página 262.

Elemento pai:"application" na página 213

Elementos filho: nenhum

Conteúdo

`true` ou `false` (padrão)

Exemplo

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 e posterior - Opcional

Indica que o aplicativo exige o uso do buffer de profundidade ou de estêncil. Normalmente, você usa esses buffers ao trabalhar com conteúdo 3D. Por padrão, o valor desse elemento é `false` para desativar os buffers de profundidade e de estêncil. Esse elemento é necessário, pois os buffers devem ser alocados ao inicializar o aplicativo, antes de carregar qualquer conteúdo.

A configuração desse elemento deve corresponder ao valor passado para o argumento `enableDepthAndStencil` para o método `Context3D.configureBackBuffer()`. Se os valores não corresponderem, o AIR emite um erro.

Esse elemento é aplicável somente quando `renderMode` = `direct`. Se `renderMode` não equivaler a `direct`, o ADT emite o erro 118:

```
<depthAndStencil> element unexpected for render mode cpu.  It requires "direct" render mode.
```

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

`true` ou `false` (padrão)

Exemplo

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

description

Adobe AIR 1.0 e posterior - Opcional

A descrição do aplicativo, exibida no instalador do aplicativo do AIR.

Se você especificar um único nó de texto (e não vários elementos text), o instalador do aplicativo do AIR usará essa descrição, independentemente do idioma do sistema. Caso contrário, o instalador do aplicativo do AIR usa a descrição que mais se aproxima do idioma da interface do usuário do sistema operacional do usuário. Por exemplo, considere uma instalação na qual o elemento `description` do arquivo do descritor do aplicativo inclui um valor para o local en (inglês). O instalador do aplicativo do AIR usará a descrição en se o sistema do usuário identificar en (inglês) como o idioma da interface do usuário. Ele também usa a descrição en se o idioma da interface do usuário do sistema for en-US (inglês norte-americano). No entanto, se o idioma da interface do usuário do sistema é en-US e o arquivo do descritor do aplicativo define os nomes en-US e en-GB, o instalador do aplicativo do AIR usa o valor en-US. Se o aplicativo não define nenhuma descrição que corresponda ao idioma da interface do usuário do sistema, o instalador do aplicativo do AIR usa o primeiro valor `description` definido no arquivo do descritor do aplicativo.

Para obter mais informações sobre o desenvolvimento de aplicativos com vários idiomas, consulte "Localização de aplicativos AIR" na página 297.

Elemento pai:"application" na página 213

Elementos filho:"text" na página 244

Conteúdo
O esquema do descritor do aplicativo do AIR 1.0 permite apenas um simples nó de texto a ser definido para o nome (e não vários elementos `text`).

No AIR 1.1 (ou acima), você pode especificar vários idiomas no elemento `description`. O atributo `xml:lang` para cada elemento de texto especifica um código de idioma, como definido em RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

Exemplo
Descrição com nó de texto simples:

```
<description>This is a sample AIR application.</description>
```

Descrição com elementos do texto traduzido para inglês, francês e espanhol (válido no AIR 1.1 e 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 e posterior - Necessário

A descrição do tipo de arquivo é exibida para o usuário pelo sistema operacional. A descrição do tipo de arquivo não é traduzida.

Consulte também: "description" na página 221 como filho do elemento do aplicativo

Elemento pai: "fileType" na página 225

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres que descreve o conteúdo do arquivo.

Exemplo
```
<description>PNG image</description>
```

embedFonts

Permite que você use fontes personalizadas no StageText no aplicativo do AIR. Esse elemento é opcional.

Elemento pai: "application" na página 213

Elementos filho:"fonte" na página 226

Conteúdo
O elemento embedFonts pode conter qualquer número de elementos.

Exemplo
```
<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 e posterior - somente iOS, opcional

O iOS usa propriedades chamadas entitlements para fornecer o acesso do aplicativo a recursos adicionais. Use o elemento Entitlements para especificar esta informação em um aplicativo iOS móvel.

Elemento pai: "iPhone" na página 234

Elementos filho: Elementos iOS Entitlements.plist

Conteúdo
Contém elementos filho que especificam os pares com valor-chave para usar como configurações de Entitlements.plist para o aplicativo. O conteúdo do elemento Entitlements deve ser incluído em um bloco CDATA. Para obter mais informações, consulte Principais referências de Entitlement na biblioteca do desenvolvedor iOS.

Exemplo

```
<iphone>
...
   <Entitlements>
      <![CDATA[
         <key>aps-environment</key>
         <string>development</string>
      ]]>
   </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 e posterior - Necessário

A sequência de caracteres de extensão de um tipo de arquivo.

Elemento pai:“fileType” na página 225

Elementos filho: nenhum

Conteúdo
Uma sequência que identifica os caracteres de extensão do arquivo (sem o ponto, ".").

Exemplo

```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 e posteriores

Especifica o ID de uma extensão ActionScript usada pelo aplicativo. O ID é definida no documento descritor da extensão.

Elemento pai: “extensions” na página 223

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres que identifica o ID da extensão ActionScript.

Exemplo

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

extensions

Adobe AIR 2.5 e posterior - opcional

Identifica as extensões ActionScript usadas por um aplicativo.

Elemento pai: “application” na página 213

Elementos filho:“extensionID” na página 223

Conteúdo
Elementos filho `extensionID` que contêm os IDs de extensão ActionScript do arquivo descritor de extensão.

Exemplo

```
<extensions>
    <extensionID>extension.first</extensionID>
    <extensionID>extension.next</extensionID>
    <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 e posterior, somente iOS — opcional

Especifica o nome de um arquivo de texto que contenha uma lista de arquivos SWF a serem configurados pelo ADT para hospedagem remota. Você pode minimizar o tamanho inicial de download do aplicativo ao empacotar um subconjunto dos SWFs usados por seu aplicativo e carregar os SWFs externos restantes (apenas ativos) no runtime usando o método `Loader.load()`. Para usar esse recurso, é necessário empacotar o aplicativo para que o ADT mova todo o ActionScript ByteCode (ABC) dos arquivos SWF carregados externamente para o aplicativo SWF principal, deixando um arquivo SWF que contenha todos os recursos. Isso é feito para obedecer à regra da Apple Store que proíbe o download de códigos depois que um aplicativo é instalado.

Para obter mais informações, consulte "Minimizar o tamanho do download carregando SWFs externos com somente ativos" na página 87.

Elemento pai:"iPhone" na página 234, "initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
O nome de um arquivo de texto que contenha uma lista determinada por linhas dos SWFs que serão hospedados remotamente.

Atributos:
Nenhum.

Exemplos
iOS:

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

filename

Adobe AIR 1.0 e posterior - Necessário

A sequência de caracteres a ser usada como um filename do aplicativo (sem extensão) quando o aplicativo é instalado. O arquivo do aplicativo inicia o aplicativo do AIR no tempo de execução. Se nenhum valor `name` for fornecido, `filename` também será usado como o nome da pasta de instalação.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo

A propriedade `filename` pode conter qualquer caractere Unicode (UTF-8), exceto o seguinte, que tem o uso proibido como filenames em vários sistemas de arquivos:

Caractere	Código hexadecimal
vários	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

O valor `filename` não pode terminar em um ponto.

Exemplo

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 e posterior - Opcional

Descreve um tipo único de arquivo que o aplicativo pode se cadastrar.

Elemento pai:"fileTypes" na página 226

Elementos filho:

- "contentType" na página 219
- "description" na página 221
- "extension" na página 223
- "icon" na página 229
- "name" na página 239

Conteúdo

Elementos que descrevem um tipo de arquivo.

Exemplo

```
<fileType>
    <name>foo.example</name>
    <extension>foo</extension>
    <description>Example file type</description>
    <contentType>text/plain</contentType>
    <icon>
        <image16x16>icons/fooIcon16.png</image16x16>
        <image48x48>icons/fooIcon48.png</imge48x48>
    <icon>
</fileType>
```

fileTypes

Adobe AIR 1.0 e posterior - Opcional

O elemento `fileTypes` permite declarar os tipos de arquivos com os quais um aplicativo do AIR pode ser associado.

Quando um aplicativo do AIR for instalado, qualquer tipo de arquivo declarado é registrado com o sistema operacional. Se esses tipos de arquivo ainda não estiverem associados a um outro aplicativo, eles são associados ao aplicativo do AIR. Para substituir uma associação existente entre um tipo de arquivo e outro aplicativo, use o método `NativeApplication.setAsDefaultApplication()` em runtime (preferencialmente com a permissão do usuário).

***Nota:** Os métodos runtime podem apenas gerenciar associações para os tipos de arquivos declarados no descritor do aplicativo.*

O elemento `fileTypes` é opcional.

Elemento pai: “application” na página 213

Elementos filho:“fileType” na página 225

Conteúdo

O elemento `fileTypes` pode conter qualquer número de elementos `fileType`.

Exemplo

```
<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>
```

fonte

Descreve uma única fonte personalizada que pode ser usada no aplicativo do AIR.

Elemento pai:"embedFonts" na página 222

Elementos filho:"fontName" na página 227, "fontPath" na página 227

Conteúdo
Elementos que especificam o nome da fonte personalizada e seu caminho respectivo.

Exemplo
```
<font>
   <fontPath>ttf/space age.ttf</fontPath>
   <fontName>space age</fontName>
</font>
```

fontName

Especifica o nome da fonte personalizada.

Elemento pai:"fonte" na página 226

Elementos filho: nenhum

Conteúdo
Nome da fonte personalizada a ser especificada em StageText.fontFamily

Exemplo
```
<fontName>space age</fontName>
```

fontPath

Oferece o local do arquivo da fonte personalizada.

Elemento pai:"fonte" na página 226

Elementos filho: nenhum

Conteúdo
Caminho do arquivo de fonte personalizada (em relação à origem).

Exemplo
```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 e posterior, somente iOS — opcional

Force o CPU a entrar no modo de renderização para um conjunto específico de dispositivos. Esse recurso permite que você efetivamente ative o modo de renderização da GPU de forma seletiva para os dispositivos iOS restantes.

Adicione esta tag como filha da tag `iPhone` e especifique uma lista separada por espaços com os nomes dos modelos de dispositivos. Os nomes de modelos de dispositivo válidos incluem:

iPad 1.1	iPhone 1.1	iPod 1.1
iPad 2.1	iPhone 1.2	iPod 2.1
iPad 2.2	iPhone 2.1	iPod 3.3
iPad 2.3	iPhone 3.1	iPod 4.1
iPad 2.4	iPhone 3.2	iPod 5.1
iPad 2.5	iPhone 4.1	
iPad 3.1	iPhone 5.1	
iPad 3.2		
iPad 3.3		
iPad 3.4		

Elemento pai:"iPhone" na página 234, "initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Lista dos nomes dos modelos de dispositivos separados por espaços.

Atributos:
Nenhum.

Exemplos
iOS:

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

fullScreen

Adobe AIR 2.0 e posterior, iOS e Android — opcional

Especifica se o aplicativo é iniciado no modo de tela cheia.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
`true` ou `false` (padrão)

Exemplo

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 e posterior - Opcional

A altura inicial da janela principal do aplicativo.

Se você não definir uma altura, esta é determinada pelas configurações no arquivo SWF raiz ou, no caso de um aplicativo do AIR baseado em HTML, pelo sistema operacional.

A altura máxima de uma janela muda de 2048 para 4096 pixels no AIR 2.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

Um número inteiro positivo com um valor máximo de 4095.

Exemplo

```
<height>4095</height>
```

icon

Adobe AIR 1.0 e posterior - Opcional

A propriedade `icon` especifica um ou mais arquivos de ícone a serem usados pelo aplicativo. A inclusão de um ícone é opcional. Se você não especificar uma propriedade `icon`, o sistema operacional exibirá um ícone padrão.

O caminho especificado é relativo ao diretório raiz do aplicativo. Os arquivos de ícone devem estar no formato PNG. Você pode especificar todos os tamanhos de ícones a seguir:

Se um elemento para um determinado tamanho estiver presente, a imagem no arquivo deverá ser exatamente do tamanho especificado. Se todos os tamanhos não forem fornecidos, o tamanho mais próximo será dimensionado para se ajustar para um determinado uso do ícone pelo sistema operacional.

Nota: *Os ícones especificados não são automaticamente adicionados ao pacote do AIR. Os arquivos de ícone devem ser incluídos em seus locais corretos relativos quando o aplicativo for empacotado.*

Para melhores resultados, forneça uma imagem para cada um dos tamanhos disponíveis. Além disso, verifique se os ícones estão apresentáveis nos modos de cores de 16 e 32 bits.

Elemento pai: "application" na página 213

Elementos filho: "imageNxN" na página 230

Conteúdo

Um elemento imageNxN para cada tamanho de ícone desejado.

Exemplo

```
<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 e posterior - Necessário

Uma sequência de caracteres de identificador para o aplicativo, conhecida como O ID do aplicativo. Um identificador do estilo DNS reverso é usado frequentemente, mas este estilo não é necessário.

Elemento pai: “application” na página 213

Elementos filho: nenhum

Conteúdo
O valor da ID é restrito aos seguintes caracteres:

- 0–9
- a–z
- A–Z
- . (ponto)
- - (hífen)

O valor deve conter de 1 a 212 caracteres. Esse elemento é necessário.

Exemplo

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 e posterior - Opcional

Define o caminho para um ícone relativo ao diretório do aplicativo.

As imagens de ícone a seguir podem ser usadas, cada um especificando um tamanho diferente de ícone:

- image16x16
- image29x29 (acima de AIR 2)
- image32x32
- image36x36 (acima de AIR 2.5)
- image48x48
- image50x50 (acima de AIR 3.4)
- image57x57 (acima de AIR 2)
- image58x58 (acima de AIR 3.4)
- image72x72 (acima de AIR 2)
- image100x100 (acima de AIR 3.4)
- image114x114 (acima de AIR 2.6)
- image128x128
- image144x144 (acima de AIR 3.4)
- image512x512 (acima de AIR 2)

- image1024x1024 (acima de AIR 3.4)

O ícone deve ser um gráfico PNG exatamente do tamanho indicado pelo elemento de imagem. Os arquivos de ícone devem ser incluídos no pacote de aplicativos; ícones referenciados no documento de descrição do aplicativo não são incluídos automaticamente.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo

O caminho do arquivo pode conter qualquer caractere Unicode (UTF-8), exceto o seguinte, que tem o uso proibido como filenames em vários sistemas de arquivos:

Caractere	Código hexadecimal
vários	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

Exemplo

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 e posterior - Opcional

Permite especificar propriedades adicionais de um aplicativo iOS.

Elemento pai:"iPhone" na página 234

Elementos filho: elementos Info.plist para iOS

Conteúdo

Contém elementos filho que especificam os pares com valor-chave para usar como configurações de Info.plist para o aplicativo. Conteúdo do elemento InfoAdditions deve ser incluído em um bloco CDATA.

Consulte Referência-chave da lista de propriedade de informações na Biblioteca de referência do iPhone Apple para obter informações sobre pares dos principais valores e sobre como expressá-los em XML.

Exemplo

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

Mais tópicos da Ajuda

"Configurações do iOS" na página 82

initialWindow

Adobe AIR 1.0 e posterior - Necessário

Define o principal conteúdo do arquivo e aparência inicial do aplicativo.

Elemento pai: "application" na página 213

Elementos filho: Todos os elementos a seguir podem aparecer como filhos do elemento initialWindow. Contudo, alguns elementos são ignorados, dependendo se o AIR está disponível com janelas em uma plataforma:

Elemento	Área de trabalho	Móvel
"aspectRatio" na página 216	ignorada	usada
"autoOrients" na página 217	ignorada	usada
"content" na página 218	usada	usada
"depthAndStencil" na página 220	usada	usada
"fullScreen" na página 228	ignorada	usada
"height" na página 229	usada	ignorada
"maximizable" na página 236	usada	ignorada
"maxSize" na página 237	usada	ignorada
"minimizable" na página 237	usada	ignorada
"minSize" na página 237	usada	ignorada
"renderMode" na página 240	usado (AIR 3.0 e superior)	usada
"requestedDisplayResolution" na página 240	usado (AIR 3.6 e superior)	ignorada
"resizable" na página 241	usada	ignorada
"softKeyboardBehavior" na página 242	ignorada	usada
"systemChrome" na página 244	usada	ignorada

Elemento	Área de trabalho	Móvel
"title" na página 245	usada	ignorada
"transparent" na página 245	usada	ignorada
"visible" na página 247	usada	ignorada
"width" na página 247	usada	ignorada
"x" na página 248	usada	ignorada
"y" na página 248	usada	ignorada

Conteúdo

Elementos filho que definem o comportamento e a aparência do aplicativo.

Exemplo

```
<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 e posterior - Opcional

Identifica o subdiretório do diretório de instalação padrão.

No Windows, o subdiretório de instalação padrão é Arquivos de Programas. No Mac OS, é o diretório /Applications. No Linux, é /opt/. Por exemplo, se a propriedade `installFolder` é definida como `"Acme"` e um aplicativo é chamado de `"ExampleApp"`, o aplicativo é instalado em C:\Arquivos de Programas\Acme\ExampleApp no Windows, em /Applications/Acme/Example.app no Mac OS e em /opt/Acme/ExampleApp no Linux.

A propriedade `installFolder` é opcional. Se você não especificar nenhuma propriedade `installFolder`, o aplicativo será instalado em um subdiretório do diretório de instalação padrão, com base na propriedade `name`.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo
A propriedade `installFolder` pode conter qualquer caractere Unicode (UTF-8), exceto aqueles cujo uso é proibido como nomes de pastas em vários sistemas de arquivos (consulte a propriedade `filename` para obter a lista de exceções).

Use o caractere de barra (/) como o caractere separador de diretório se desejar especificar um subdiretório aninhado.

Exemplo
```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, somente iOS — opcional

Define as propriedades de aplicativos específicos para iOS.

Elemento pai: "application" na página 213

Elementos filho:

- "Entitlements" na página 222
- "externalSwfs" na página 224
- "forceCPURenderModeForDevices" na página 227
- "InfoAdditions" na página 231
- "requestedDisplayResolution" na página 240

Mais tópicos da Ajuda

"Configurações do iOS" na página 82

manifesto

Adobe AIR 2.5 e posterior, somente Android — opcional

Especifica informações para adicionar ao arquivo de manifesto do Android para o aplicativo.

Elemento pai:"manifestAdditions" na página 235

Elementos filho: Definido pelo Android SDK.

Conteúdo
O elemento do manifesto não é, tecnicamente falando, uma parte do esquema descritor do aplicativo do AIR. É a raiz do documento XML do manifesto do Android. Qualquer conteúdo que você coloca dentro do elemento do manifesto deve estar em conformidade com o esquema do AndroidManifest.xml. Ao gerar um arquivo APK com as ferramentas do AIR, as informações do elemento do manifesto é copiado para a parte correspondente do AndroidManifest.xml gerado pelo aplicativo.

Se você especificar valores de manifesto do Android que estão disponíveis apenas em uma versão SDK mais recente do que a suportada diretamente pelo AIR, você precisa configurar o sinalizador `-platformsdk` para o ADT ao empacotar o aplicativo. Defina o sinalizador para o caminho do sistema de arquivos para um SDK do Android que suporte os valores que você está adicionando.

O próprio elemento do manifesto deve ser incluído em um bloco CDATA dentro do descritor do aplicativo do AIR.

Exemplo

```
<![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>
]]>
```

Mais tópicos da Ajuda

"Configurações do Android" na página 76

O arquivo AndroidManifest.xml

manifestAdditions

Adobe AIR 2.5 e posterior, somente Android

Especifica informações para adicionar ao arquivo do manifesto do Android.

Cada aplicativo do Android inclui um arquivo de manifesto que define as propriedades básicas do aplicativo. O manifesto do Android é semelhante em conceito ao descritor do aplicativo do AIR. Um aplicativo do AIR for Android tem tanto um descritor de aplicativo quanto um arquivo de manifesto do Android gerado automaticamente. Quando um aplicativo do AIR for Android é compactado, as informações contidas neste elemento `manifestAdditions` são adicionadas às partes correspondentes do documento de manifesto do Android.

Elemento pai:"android" na página 213

Elementos filho:"manifesto" na página 234

Conteúdo

As informações no elemento `manifestAdditions` são adicionadas ao documento XML do AndroidManifest.

O AIR define várias entradas de manifesto no documento de manifesto Android gerado para garantir que os recursos de aplicação e runtime funcionem corretamente. Não é possível substituir as configurações a seguir:

Não é possível definir os seguintes atributos do elemento de manifesto:

- package
- android:versionCode
- android:versionName

Não é possível definir os seguintes atributos do elemento de atividade principal:

- android:label

- android:icon

Não é possível definir os seguintes atributos do elemento de aplicativo:

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

Exemplo

```
<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>
```

Mais tópicos da Ajuda

"Configurações do Android" na página 76

O arquivo AndroidManifest.xml

maximizable

Adobe AIR 1.0 e posterior - Opcional

Especifica se a janela pode ser maximizada.

***Nota:** Em sistemas operacionais como o Mac OS X, em que maximizar janelas é uma operação de redimensionamento, tanto maximizable quanto resizable devem ser definidos como `false` para impedir que a janela seja ampliada ou redimensionada.*

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

`true` (padrão) ou `false`

Exemplo

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 e posterior - Opcional

O tamanho máximo da janela. Se você não definir um tamanho máximo, este será determinado pelo sistema operacional.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Dois inteiros que representam a largura e a altura máximas, separadas por espaço em branco.

Nota: *O tamanho máximo da janela disponível pelo AIR aumentou de 2048x2048 para 4096x4096 pixels no AIR 2. (Como as coordenadas da tela são baseadas em zero, o valor máximo que você pode usar para largura ou altura é 4095.)*

Exemplo

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 e posterior - Opcional

Especifica se a janela pode ser minimizada.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
`true` (padrão) ou `false`

Exemplo

```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 e posterior - Opcional

Especifica o tamanho mínimo permitido para a janela.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Dois inteiros que representam a largura e a altura mínimas, separadas por espaço em branco. Observe que o tamanho mínimo imposto pelo sistema operacional tem precedência sobre o valor definido no descritor do aplicativo.

Exemplo

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 e posterior - Opcional

O título do aplicativo que é exibido pelo instalador do aplicativo do AIR.

Se nenhum elemento `name` for especificado, o instalador do aplicativo do AIR exibirá `filename` como o nome do aplicativo.

Elemento pai: "application" na página 213

Elementos filho:"text" na página 244

Conteúdo

Se você especificar um único nó de texto (em vez de vários elementos `<text>`), o instalador do aplicativo do AIR usa esse nome, independentemente do idioma do sistema.

O esquema do descritor do aplicativo do AIR 1.0 permite apenas um simples nó de texto a ser definido para o nome (e não vários elementos `text`). No AIR 1.1 (ou acima), você pode especificar vários idiomas no elemento `name`.

O atributo `xml:lang` para cada elemento de texto especifica um código de idioma, como definido em RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

O instalador do aplicativo do AIR usa o nome que mais se aproxima do idioma da interface do usuário do sistema operacional do usuário. Por exemplo, considere uma instalação na qual o elemento `name` do arquivo do descritor do aplicativo inclui um valor para o local en (inglês). O instalador do aplicativo do AIR usa o nome en se o sistema operacional identifica en (inglês) como o idioma da interface do usuário. Ele também usa o nome en se o idioma da interface do usuário do sistema for en-US (inglês norte-americano). No entanto, se o idioma da interface do usuário é en-US e o arquivo do descritor do aplicativo define os nomes en-US e en-GB, o instalador do aplicativo do AIR usa o valor en-US. Se o aplicativo não define nenhum nome que corresponda aos idiomas da interface do usuário do sistema, o instalador do aplicativo do AIR usa o primeiro valor `name` definido no arquivo do descritor do aplicativo.

O elemento `name` define apenas o título do aplicativo usado no instalador do aplicativo do AIR. O instalador do aplicativo do AIR suporta vários idiomas: chinês tradicional, chinês simplificado, tcheco, holandês, inglês, francês, alemão, italiano, japonês, coreano, português do Brasil, russo, espanhol, sueco e turco. O instalador do aplicativo do AIR seleciona o idioma exibido (para outros textos que não o título do aplicativo e a descrição) com base no idioma da interface do usuário do sistema. Essa seleção de idioma independe das configurações no arquivo do descritor do aplicativo.

O elemento `name`*não* define as localidades disponíveis para o aplicativo instalado em execução. Para obter detalhes sobre o desenvolvimento de aplicativos com vários idiomas, consulte "Localização de aplicativos AIR" na página 297.

Exemplo

O exemplo a seguir define um nome com um nó de texto simples:

```
<name>Test Application</name>
```

O exemplo a seguir, válido no AIR 1.1 e posterior, especifica o nome em três idiomas (inglês, francês e espanhol), utilizando nós do 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 e posterior - Necessário

Identifica o nome de um tipo de arquivo.

Elemento pai:"fileType" na página 225

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres que representa o nome do tipo de arquivo.

Exemplo

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 e posterior - Opcional

Identifica o local no qual colocar atalhos para o aplicativo no menu Todos os Programas do sistema operacional Windows ou no menu Aplicativos do Linux. (Essa configuração é atualmente ignorada em outros sistemas operacionais.)

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo
A sequência de caracteres para o valor `programMenuFolder` pode conter qualquer caractere Unicode (UTF-8), exceto aqueles cujo uso é proibido, como nomes de pastas em vários sistemas de arquivos (consulte o elemento `filename` para ver a lista de exceções). *Não* use um caractere de barra (/) como o último caractere desse valor.

Exemplo

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 e posterior - opcional

Identifica o ID do editor para atualizar um aplicativo do AIR criada originalmente com AIR versão 1.5.2 ou anterior.

Somente especifique um ID do editor ao criar uma atualização do aplicativo. O valor do elemento `publisherID` deve corresponder à ID do editor gerada pelo AIR para a versão anterior do aplicativo. Para um aplicativo instalado, o ID do editor pode ser encontrada na pasta em que um aplicativo é instalado, no arquivo `META-INF/AIR/publisherid`.

Novos aplicativos criados com o AIR 1.5.3 ou posterior não devem especificar um ID do editor.

Para obter mais informações, consulte "Sobre identificadores de editor do AIR" na página 194.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres do ID do editor.

Exemplo
```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 e posterior - opcional

Especifica se deve usar aceleração de unidade de processamento gráfico (GPU), se disponível no dispositivo de computação atual.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Um dos seguintes valores:

- `auto` (padrão) — volta no momento para o modo de CPU.
- `cpu` — a aceleração por hardware não é usada.
- `direct` — a composição de renderização ocorre na CPU; o blitting usa a GPU. Disponível no AIR 3+.

  Nota: *Para aproveitar a aceleração GPU do conteúdo Flash com o AIR para plataformas móveis, o Adobe recomenda que você use renderMode="direct", o Stage3D e não o renderMode="gpu". A Adobe oferece suporte e recomenda oficialmente as seguintes estruturas baseadas em Stage3D: Starling (2D) e Away3D (3D). Para obter mais detalhes do Stage3D e do Starling/Away3D (3D), consulte http://gaming.adobe.com/getstarted.*
- `gpu` — a aceleração por hardware é usada, se disponível.

  Importante: *Não utilize o modo de renderização de GPU para aplicativos Flex.*

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

requestedDisplayResolution

Adobe AIR 2.6 ou posterior, somente iOS; Adobe AIR 3.6 e posterior OS X — Opcional

Especifica se o aplicativo deseja usar a resolução padrão ou alta em um dispositivo ou monitor de computador com uma tela de alta resolução. Quando definida para *padrão*, a tela aparecerá para o aplicativo como uma tela de resolução padrão. Quando definida para *alta*, o aplicativo pode tratar cada pixel de alta resolução.

Por exemplo, em uma tela de iPhone de 640x960 de alta resolução, se a configuração for *standard*, as dimensões do palco em tela inteira são 320x480 e cada pixel do aplicativo é renderizado usando quatro pixels da tela. Se a configuração for *high*, as dimensões do palco em tela inteira são 640x960.

Em dispositivos com telas de resolução padrão, as dimensões do palco combinam com de tela, sem importar qual definição é usada.

Se o elemento `requestedDisplayResolution` estiver aninhado no elemento `iPhone`, ele se aplica aos dispositivos iOS. Neste caso, o atributo `excludeDevices` pode ser usado para especificar dispositivos aos quais a configuração não é aplicada.

Se o elemento `requestedDisplayResolution` estiver aninhado ao elemento `initialWindow`, ele se aplica aos aplicativos de desktop AIR nos computadores MacBook Pro compatíveis com exibições em alta resolução. O valor especificado se aplica a todas as janelas nativas usadas no aplicativo. O aninhamento do elemento `requestedDisplayResolution` ao elemento`initialWindow` é suportado no AIR 3.6 e posterior.

Elemento pai:"iPhone" na página 234, "initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Seja *padrão* ou *alta*.

Atributo:
excludeDevices — uma lista de nomes de modelos do iOS ou prefixos de nomes de modelos separados por espaços. Isto permite ao desenvolvedor ter alguns dispositivos que usam alta resolução e outros que usam resolução padrão. Este atributo está disponível apenas no iOS (o elemento `requestedDisplayResolution` é aninhado no elemento `iPhone`). O atributo `excludeDevices` está disponível no AIR 3.6 e posterior.

Para qualquer dispositivo cujo o nome do modelo esteja especificado neste atributo, o valor `requestedDisplayResolution` é o oposto do valor especificado. Em outras palavras, se o valor `requestedDisplayResolution` for *high*, os dispositivos excluídos usam a resolução padrão. Se o valor `requestedDisplayResolution` for *standard*, os dispositivos excluídos usam a alta resolução.

Os valores são nomes de modelos de dispositivos iOS ou prefixos de nomes de modelos. Por exemplo, o valor iPad3.1 se refere especificamente a um iPad com Wi-Fi de 3ª geração (mas não aos iPads GSM ou CDMA de 3ª geração). Como alternativa, o valor iPad3 se refere a qualquer iPad de 3ª geração. Uma lista não oficial de nomes de modelos de iOS está disponível na página wiki de modelos de iPhone.

Exemplos
Desktop:

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

iOS:

```
<iPhone>
    <requestedDisplayResolution excludeDevices="iPad3
iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 e posterior - Opcional

Especifica se a janela pode ser redimensionada.

***Nota:** Em sistemas operacionais como o Mac OS X, em que maximizar janelas é uma operação de redimensionamento, tanto maximizable quanto resizable devem ser definidos como `false` para impedir que a janela seja ampliada ou redimensionada.*

Elemento pai:"initialWindow" na página 232

Elementos filho:

Conteúdo
`true` (padrão) ou `false`

Exemplo
```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 e posterior (perfil móvel) - opcional

Especifica o comportamento padrão do aplicativo quando um teclado virtual é exibido. O comportamento padrão é deslocar o aplicativo para cima. O runtime mantém o objeto interativo ou o campo de texto focalizado na tela. Use a opção *pan* se o aplicativo não fornecer sua própria lógica de manipulação do teclado.

Você também pode desativar o comportamento automático, definindo o elemento `softKeyboardBehavior` para *nenhum*. Neste caso, os campos de texto e objetos interativos enviam um SoftKeyboardEvent quando o teclado virtual é gerado, mas o runtime não desloca ou redimensiona o aplicativo. É responsabilidade do seu aplicativo manter a área de entrada de texto em exibição.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Seja *none* ou *pan*. O valor padrão é *pan*.

Exemplo
```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Mais tópicos da Ajuda

SoftKeyboardEvent

supportedLanguages

Adobe AIR 3.2 e posterior - Opcional

Identifica os idiomas suportados pelo aplicativo. Esse elemento é usado apenas pelo iOS, runtime cativo do Mac e pelos aplicativos Android. Esse elemento é ignorado por todos os tipos de aplicativo.

Se você não especificar esse elemento, por padrão o empacotador realiza as seguintes ações com base no tipo de aplicativo:

- iOS — todos os idiomas suportados pelo runtime do AIR são listados na app store do iOS como idiomas suportados do aplicativo.
- Runtime cativo do Mac — Aplicativo empacotado com conjunto cativo não possui informações de localização.
- Android — O pacote do aplicativo tem recursos para todos os idiomas suportados pelo runtime do AIR.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo

Uma lista delimitada por espaços dos idiomas suportadas. Os valores de idiomas válidos são valores ISO 639-1 para os idiomas compatíveis com o runtime do AIR: `en`, `de`, `es`, `fr`, `it`, `ja`, `ko`, `pt`, `ru`, `cs`, `nl`, `pl`, `sv`, `tr`, `zh`, `da`, `nb` e `iw`.

O empacotador gera um erro para um valor vazio do elemento `<supportedLanguages>`.

Nota: *Marcas localizadas (como a marca de nome) ignoram o valor de um idioma se você usar a marca* `<supportedLanguages>` *e ela não contiver o idioma em questão. Se uma extensão nativa tem recursos para um idioma não especificado pela marca* `<supportedLangauges>`*, é emitido um aviso e os recursos são ignorados para o idioma em questão.*

Exemplo

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 e posterior - opcional

Identifica os perfis que são compatíveis com o aplicativo.

Elemento pai: “application” na página 213

Elementos filho: nenhum

Conteúdo

Você pode incluir qualquer um desses valores no elemento `supportedProfiles`:

- `desktop` — O perfil desktop é para aplicativos do AIR que são instalados em um computador de mesa que usa um arquivo do AIR. Esses aplicativos não precisam ter acesso à classe NativeProcess (que fornece comunicação com os aplicativos nativos).
- `extendedDesktop` — O perfil desktop estendido define os aplicativos do AIR que estão instalados em um computador de mesa usando um instalador do aplicativo nativo. Esses aplicativos precisam ter acesso à classe NativeProcess (que fornece comunicação com os aplicativos nativos).
- `mobileDevice` — O perfil de dispositivo móvel é para aplicativos móveis.
- `extendedMobileDevice`— O perfil dispositivo móvel estendido não está em uso atualmente.

A propriedade `supportedProfiles` é opcional. Quando você não incluir este elemento no arquivo descritor do aplicativo, o aplicativo pode ser compilado e implantado para qualquer perfil.

Para especificar vários perfis, separe cada um com um caractere de espaço. Por exemplo, a configuração a seguir especifica que o aplicativo somente está disponível na área de trabalho e nos perfis estendidos.

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

Nota: *Ao executar um aplicativo com ADL e não especificar um valor para a opção* `-profile` *do ADL, o primeiro perfil no descritor do aplicativo é utilizado. (Se nenhum perfil estiver especificado no descritor do aplicativo, o perfil de desktop é usado.)*

Exemplo

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

Mais tópicos da Ajuda

"Perfis de dispositivo" na página 249

"Perfis disponíveis" na página 75

systemChrome

Adobe AIR 1.0 e posterior - Opcional

Especifica se a janela inicial do aplicativo é criada com controles, bordas e barra de título padrão fornecidos pelo sistema operacional.

A configuração do cromo do sistema da janela não pode ser alterada em runtime.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

Um dos seguintes valores:

- `none` — Nenhum cromo do sistema é fornecido. O aplicativo (ou um framework do aplicativo, como Flex) é responsável por exibir o cromo da janela.
- `standard` (padrão) — O cromo do sistema é fornecido pelo sistema operacional.

Exemplo

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 e posterior - opcional

Especifica uma sequência de caracteres traduzida.

O atributo `xml:lang` de um elemento de texto especifica um código de idioma, como definido em RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

O instalador do aplicativo do AIR usa o elemento `text` com o valor de atributo `xml:lang` que mais se aproxima do idioma da interface do sistema operacional do usuário.

Por exemplo, considere uma instalação na qual um elemento `text` inclui um valor para o local en (inglês). O instalador do aplicativo do AIR usa o nome en se o sistema operacional identifica en (inglês) como o idioma da interface do usuário. Ele também usa o nome en se o idioma da interface do usuário do sistema for en-US (inglês norte-americano). No entanto, se o idioma da interface do usuário é en-US e o arquivo do descritor do aplicativo define os nomes en-US e en-GB, o instalador do aplicativo do AIR usa o valor en-US.

Se o aplicativo não define nenhum elemento `text` que corresponda aos idiomas da interface do usuário do sistema, o instalador do aplicativo do AIR usa o primeiro valor `name` definido no arquivo do descritor do aplicativo.

Elementos pai:

- "name" na página 238
- "description" na página 221

Elementos filho: nenhum

Conteúdo
Um atributo `xml:lang` que especifica um local e uma sequência de caracteres do texto traduzido.

Exemplo
```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 e posterior - Opcional

Especifica o título exibido na barra de título da janela inicial do aplicativo.

Um título é exibido apenas se o elemento `systemChrome` estiver definido para `standard`.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres que contém o título da janela.

Exemplo
```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 e posterior - Opcional

Especifica se a janela inicial do aplicativo é alfa mesclada com o desktop.

Uma janela com transparência ativada pode ser desenhada mais lentamente e exigir mais memória. A configuração de transparente não pode ser alterada no tempo de execução.

Importante: *Você pode definir apenas* `transparent` *como* `true` *quando* `systemChrome` *for* `none`.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
`true` ou `false` (padrão)

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

version

Adobe AIR 1.0 a 2.0 — obrigatório; não permitido em AIR 2.5 e posterior

Especifica as informações de versão para o aplicativo.

A sequência de caracteres da versão é um designador definido pelo aplicativo. O AIR não interpreta de maneira nenhuma a string de versão. Portanto, não se supõe que a versão "3.0" é mais atual que a versão "2.0." Exemplos: `"1.0"`, `".4"`, `"0.5"`, `"4.9"`, `"1.3.4a"`.

Em AIR 2.5 e posterior, o elemento `version` é substituído pelos elementos `versionNumber` e `versionLabel`.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres que contém uma versão do aplicativo.

Exemplo
```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 e posterior - opcional

Especifica uma sequência de caracteres da versão legível por pessoas.

O valor do rótulo da versão é exibido em diálogos de instalação em vez de o valor do elemento `versionNumber`. Se `versionLabel` não for usado, o `versionNumber` é usado para ambos.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo
Uma sequência de caracteres que contém o texto da versão exibida publicamente.

Exemplo
```
<versionLabel>0.9 Beta</versionlabel>
```

versionNumber

Adobe AIR 2.5 e posterior - obrigatório

O número da versão do aplicativo.

Elemento pai: "application" na página 213

Elementos filho: nenhum

Conteúdo
O número da versão pode conter uma sequência de até três números inteiros separados por pontos. Cada inteiro deve ser um número entre 0 e 999 (inclusive).

Exemplos

```
<versionNumber>1.0.657</versionNumber>

<versionNumber>10</versionNumber>

<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 e posterior - Opcional

Especifica se a janela inicial do aplicativo fica visível assim que é criada.

Janelas do AIR, incluindo a janela inicial, são criadas em estado invisível por padrão. Você pode exibir a janela chamando o método `activate()` do objeto NativeWindow ou definindo a propriedade `visible` como `true`. Você pode querer deixar a janela principal oculta inicialmente, para que alterações na posição da janela, no tamanho da janela e o layout de seu conteúdo não sejam exibidos.

O componente mx:WindowedApplication do Flex exibe e ativa de maneira automática a janela imediatamente antes que o evento `applicationComplete` seja despachado, a menos que o atributo `visible` seja definido como `false` na definição MXML.

Nos dispositivos em perfis móveis, que não são compatíveis com janelas, a configuração visível é ignorada.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

`true` ou `false` (padrão)

Exemplo

```
<visible>true</visible>
```

width

Adobe AIR 1.0 e posterior - Opcional

A largura inicial da janela principal do aplicativo.

Se você não definir uma largura, esta é determinada pelas configurações no arquivo SWF raiz ou, no caso de um aplicativo do AIR baseado em HTML, pelo sistema operacional.

A largura máxima de uma janela muda de 2048 para 4096 pixels no AIR 2.

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo

Um número inteiro positivo com um valor máximo de 4095.

Exemplo

```
<width>1024</width>
```

x

Adobe AIR 1.0 e posterior - Opcional

A posição horizontal da janela inicial do aplicativo.

Na maioria dos casos, é melhor deixar o sistema operacional determinar a posição inicial da janela em vez de atribuir um valor fixo.

A origem do sistema de coordenadas da tela (0,0) é o canto superior esquerdo da tela principal do desktop (conforme determinado pelo sistema operacional).

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Um valor inteiro.

Exemplo

```
<x>120</x>
```

y

Adobe AIR 1.0 e posterior - Opcional

A posição vertical da janela inicial do aplicativo.

Na maioria dos casos, é melhor deixar o sistema operacional determinar a posição inicial da janela em vez de atribuir um valor fixo.

A origem do sistema de coordenadas da tela (0,0) é o canto superior esquerdo da tela principal do desktop (conforme determinado pelo sistema operacional).

Elemento pai:"initialWindow" na página 232

Elementos filho: nenhum

Conteúdo
Um valor inteiro.

Exemplo

```
<y>250</y>
```

Capítulo 15: Perfis de dispositivo

Adobe AIR 2 e posterior

Os perfis são um mecanismo para definir as classes de dispositivos de computação onde funciona seu aplicativo. Um perfil define um conjunto de APIs e de recursos normalmente disponíveis em uma classe particular do dispositivo. Os perfis disponíveis incluem:

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Você pode definir os perfis para seu aplicativo no descritor do aplicativo. Usuários de computadores e dispositivos em perfis incluídos podem instalar o aplicativo; usuários de outros computadores e dispositivos não podem. Por exemplo, se você incluir somente o perfil desktop no descritor do aplicativo, os usuários podem instalar e executar o aplicativo apenas em computadores desktop.

Se você incluir um perfil que seu aplicativo não seja realmente compatível, a experiência do usuário em tais ambientes podem ser pobres. Se você não especificar nenhum perfil no descritor do aplicativo, o AIR não limitará o aplicativo. Você pode compactar o aplicativo em qualquer um dos formatos compatíveis, e os usuários com dispositivos de qualquer perfil podem instalá-lo - porém, pode não funcionar corretamente em runtime.

Sempre que possível, são impostas restrições de perfil ao compactar o aplicativo. Por exemplo, se você incluir somente o perfil extendedDesktop, não poderá compactar o aplicativo como um arquivo AIR - apenas como um instalador nativo. Da mesma forma, se você não incluir o perfil MobileDevice, não poderá compactar o aplicativo como um APK do Android.

Um único dispositivo de computação pode ser compatível com mais de um perfil. Por exemplo, o AIR em computadores desktop são compatíveis com aplicativos de perfis desktop e extendedDesktop. No entanto, um aplicativo de perfil desktop estendido pode se comunicar com processos nativos e DEVE ser compactado como um instalador nativo (exe, dmg, deb ou rpm). Um aplicativo com perfil desktop, por outro lado, não pode se comunicar com um processo nativo. Um aplicativo com perfil desktop pode ser compactado como um arquivo AIR ou um instalador nativo.

A inclusão de um recurso em um perfil indica que o suporte para esse recurso é comum na classe de dispositivos para a qual esse perfil está definido. No entanto, isso não significa que cada dispositivo em um perfil seja compatível com todos os recursos. Por exemplo, a maioria (mas não todos) dos telefones móveis contém um acelerômetro. Classes e recursos que não têm suporte universal geralmente têm uma propriedade booliana que você pode verificar antes de usar o recurso. No caso do acelerômetro, por exemplo, você pode testar a propriedade estática `Accelerometer.isSupported` para determinar se o dispositivo atual tem um acelerômetro compatível.

Os perfis a seguir podem ser atribuídos ao aplicativo do AIR usando o elemento supportedProfiles no descritor do aplicativo:

Área de trabalho O perfil desktop define um conjunto de capacidades para os aplicativos AIR, que são instaladas como arquivos AIR em um computador pessoal. Estes aplicativos são instalados e executados nas plataformas de computador pessoal suportadas (sistemas operacionais Mac, Windows e Linux). Os aplicativos AIR desenvolvidos em versões do AIR anteriores a AIR 2 podem ser considerados inseridos no perfil desktop. Algumas APIs não funcionam neste perfil. Por exemplo, os aplicativos de computação pessoal não podem se comunicar com processos nativos.

Desktop estendido O perfil desktop estendido define um conjunto de capacidades para os aplicativos AIR que compõem o pacote e são instaladas com um programa de instalação nativo. Estes programas de instalação nativos são arquivos EXE no Windows, arquivos DMG no Mac OS, e arquivos DEB ou RPM no Linux. Os aplicativos com o perfil desktop estendido têm capacidades adicionais que não estão disponíveis nos aplicativos com o perfil desktop. Para obter mais informações, consulte "Compactação de um instalador desktop nativo" na página 55.

Dispositivo móvel O perfil dispositivo móvel define um conjunto de capacidades para aplicativos que são instalados em dispositivos móveis, tais como telefones celulares e tablets. Esses aplicativos são instalados e executados em plataformas móveis suportadas, incluindo Android, Blackberry Tablet OS e iOS.

Dispositivo móvel estendido O perfil dispositivo móvel define um extenso conjunto de capacidades para aplicativos que são instalados em dispositivos móveis. Atualmente, não existem dispositivos que suportam este perfil.

Como restringir perfis de destino no arquivo de descrição do aplicativo

Adobe AIR 2 e posterior

Como no AIR 2, o arquivo de descrição do aplicativo inclui um elemento `supportedProfiles`, que permite restringir perfis de destino. Por exemplo, a seguinte configuração especifica que o aplicativo somente está disponível no perfil desktop.

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

Quando este elemento é definido, o aplicativo pode ser empacotado nos perfis que você listar. Use os seguintes valores:

- `desktop`— O perfil desktop
- `extendedDesktop`— O perfil desktop estendido
- `mobileDevice`— O perfil dispositivo móvel

O elemento `supportedProfiles` é opcional. Quando você não incluir este elemento no arquivo e descrição do aplicativo, o aplicativo poderá ser compilado e implementado para qualquer perfil.

Para especificar vários perfis no elemento `supportedProfiles`, separe cada um com um caractere de espaço como no exemplo a seguir:

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

Capacidades de perfis diferentes

Adobe AIR 2 e posterior

A tabela a seguir lista as classes e os recursos que não são compatíveis com todos os perfis.

Classe ou recurso	desktop	extendedDesktop	mobileDevice
Acelerômetro (Accelerometer.isSupported)	Não	Não	Verificar
Acessibilidade (Capabilities.hasAccessibility)	Sim	Sim	Não
Eliminação de eco acústico (microfonia) (Microphone.getEnhancedMicrophone())	Sim	Sim	Não
ActionScript 2	Sim	Sim	Não
Matriz CacheAsBitmap	Não	Não	Sim
Câmera (Camera.isSupported)	Sim	Sim	Sim
CameraRoll	Não	Não	Sim
CameraUI (CameraUI.isSupported)	Não	Não	Sim
Conjuntos de runtime cativo	Sim	Sim	Sim
ContextMenu (ContextMenu.isSupported)	Sim	Sim	Não
DatagramSocket (DatagramSocket.isSupported)	Sim	Sim	Sim
DockIcon (NativeApplication.supportsDockIcon)	Verificar	Verificar	Não
Drag-and-drop (NativeDragManager.isSupported)	Sim	Sim	Verificar
EncyptedLocalStore (EncyptedLocalStore.isSupported)	Sim	Sim	Sim
Acesso Flash (DRMManager.isSupported)	Sim	Sim	Não
GameInput (GameInput.isSupported)	Não	Não	Não
Geolocalização (Geolocation.isSupported)	Não	Não	Verificar
HTMLLoader (HTMLLoader.isSupported)	Sim	Sim	Não
IME (IME.isSupported)	Sim	Sim	Verificar
LocalConnection (LocalConnection.isSupported)	Sim	Sim	Não
Microfone (Microphone.isSupported)	Sim	Sim	Verificar
Áudio multicanal (Capabilities.hasMultiChannelAudio())	Não	Não	Não
Extensões Nativas	Não	Sim	Sim
NativeMenu (NativeMenu.isSupported)	Sim	Sim	Não
NativeProcess (NativeProcess.isSupported)	Não	Sim	Não
NativeWindow (NativeWindow.isSupported)	Sim	Sim	Não
NetworkInfo (NetworkInfo.isSupported)	Sim	Sim	Verificar
Abra os arquivos com o aplicativo padrão	Limitado	Sim	Não
PrintJob (PrintJob.isSupported	Sim	Sim	Não
SecureSocket (SecureSocket.isSupported)	Sim	Sim	Verificar

Classe ou recurso	desktop	extendedDesktop	mobileDevice
ServerSocket (ServerSocket.isSupported)	Sim	Sim	Sim
Shader	Sim	Sim	Limitado
Stage3D (Stage.stage3Ds.length)	Sim	Sim	Sim
Orientação do Palco (Stage.supportsOrientationChange)	Não	Não	Sim
StageVideo	Não	Não	Verificar
StageWebView (StageWebView.isSupported)	Sim	Sim	Sim
Iniciar aplicativo no login (NativeApplication.supportsStartAtLogin)	Sim	Sim	Não
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Sim	Sim	Não
Modo ocioso do sistema	Não	Não	Sim
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Verificar	Verificar	Não
Entrada da Text Layout Framework	Sim	Sim	Não
Updater (Updater.isSupported)	Sim	Não	Não
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Sim	Sim	Não

As entradas na tabela têm o seguinte significado:

- *Verificar* — O recurso é compatível com alguns, mas não com todos os dispositivos no perfil. Você deve verificar no runtime se o recurso é compatível antes de usá-lo.
- *Limitado* — O recurso é compatível, mas tem limitações significativas. Consulte a documentação pertinente para mais informações.
- *Não* — O recurso não é compatível com o perfil.
- *Sim* — O recurso é compatível com o perfil. Observe que faz falta do hardware necessário para um recurso nos dispositivos de computação individual. Por exemplo, nem todos os celulares têm câmeras.

Como especificar problemas ao depurar com ADL

Adobe AIR 2 e posterior

O ADL verifica se você especificou perfis suportados no elemento `supportedProfiles` do arquivo de descrição de aplicativos. Se você fizer isso, ao efetuar a depuração, o ADL, por padrão, como perfil o primeiro perfil suportado da lista.

Você pode especificar um perfil para a sessão de depuração de ADL usando o argumento de linha de comando `-profile`. “AIR Debug Launcher (ADL)” na página 162 Você poderá usar este argumento independentemente de especificar ou não um perfil no elemento `supportedProfiles` do arquivo de descrição do aplicativo. Contudo, se você especificar um elemento `supportedProfiles`, ele deverá incluir o perfil que você especificar na linha de comando. Do contrário, o ADL gera um erro.

Capítulo 16: API no navegador AIR.SWF

Personalização da instalação contínua badge.swf

Além de usar o arquivo badge.swf fornecido com o SDK, você pode criar seu próprio arquivo SWF para usar em uma página do navegador. Seu arquivo SWF personalizado pode interagir com o runtime das seguintes maneiras:

- Ele pode instalar um aplicativo do AIR. Consulte "Instalação de um aplicativo do AIR do navegador" na página 259.
- Ele pode verificar se um aplicativo do AIR específico está instalado. Consulte "Verificar por uma página da Web se um aplicativo do AIR está instalado" na página 258.
- Ele pode verificar se o runtime está instalado. Consulte "Verificar se o runtime está instalado" na página 257.
- Ele pode iniciar um aplicativo do AIR instalado no sistema do usuário. Consulte "Inicialização de um aplicativo do AIR instalado do navegador" na página 260.

Esses recursos são todos fornecidos ao chamar as APIs em um arquivo SWF hospedado em adobe.com: air.swf. Você pode personalizar o arquivo badge.swf e chamar as APIs air.swf a partir do seu próprio arquivo SWF.

Além disso, um arquivo SWF em execução no navegador pode se comunicar com um aplicativo do AIR em execução usando a classe LocalConnection. Para mais informações, consulte Comunicação com outras instâncias do Flash Player e AIR (para desenvolvedores em ActionScript) ou Communicating with other Flash Player and AIR instances (para desenvolvedores em HTML).

Importante: *Os recursos descritos nesta seção (e as APIs no arquivo air.swf) exigem que o usuário final tenha a atualização 3 do Adobe® Flash® Player 9 (ou superior) instalada no navegador da Web no Windows ou Mac OS. No Linux, o recurso de instalação contínua requer o Flash Player 10 (versão 10,0,12,36 ou posterior). Você pode escrever códigos para verificar a versão instalada do Flash Player e fornecer uma interface alternativa ao usuário se a versão exigida do Flash Player não for instalada. Por exemplo, se uma versão mais antiga do Flash Player estiver instalada, você poderia fornecer um link para a versão de download do arquivo AIR (em vez de usar o arquivo badge.swf ou a API do air.swf para instalar um aplicativo).*

Usando o arquivo badge.swf para instalar um aplicativo do AIR

Incluído no SDK do AIR e no SDK do Flex está um arquivo badge.swf, que permite usar facilmente o recurso de instalação direta. O badge.swf pode instalar o runtime e um aplicativo do AIR de um link em uma página da Web. O arquivo badge.swf e seu código-fonte são fornecidos a você para distribuição no seu site da Web.

Incorpore o arquivo badge.swf em uma página da Web

1. Localize os seguintes arquivos, fornecidos no diretório samples/badge do SDK do AIR ou do SDK do Flex, e adicione-os no seu servidor Web.
   - badge.swf
   - default_badge.html
   - AC_RunActiveContent.js
2. Abra a página default_badge.html em um editor de texto.

3 Na página default_badge.html, na função JavaScript `AC_FL_RunContent()`, ajuste as definições do parâmetro `FlashVars` para as seguintes:

Parâmetro	Descrição
`appname`	O nome do aplicativo, exibido pelo arquivo SWF quando o runtime não está instalado.
`appurl`	(Obrigatório). A URL do arquivo AIR a ser obtido por download. Você deve usar uma URL absoluta, e não relativa.
`airversion`	(Obrigatório). Para a versão 1.0 do runtime, defina isso para 1.0.
`imageurl`	A URL da imagem (opcional) para exibir no crachá.
`buttoncolor`	A cor do botão de download (especificada como um valor hexadecimal, como `FFCC00`).
`messagecolor`	A cor da mensagem de texto exibida abaixo do botão quando o runtime não está instalado (especificada como um valor hexadecimal, como `FFCC00`).

4 O tamanho mínimo do arquivo badge.swf é de 217 pixels de largura por 180 pixels de altura. Ajuste os valores dos parâmetros `width` e `height` da função `AC_FL_RunContent()` para se adequar às suas necessidades.

5 Renomeie o arquivo default_badge.html e ajuste seu código (ou inclua-o em outra página HTML) para se adequar às suas necessidades.

Nota: *Para a tag `embed` de HTML que carrega o arquivo badge.swf, não defina o atributo `wmode`; deixe-o definido como a configuração padrão (`"window"`). Outras configurações `wmode` vão impedir a instalação em alguns sistemas. Além disso, usar outras configurações `wmode` produzem um erro: "Erro #2044: ErrorEvent não tratado:. text=Error #2074: O palco está muito pequeno para baixar a iu."*

Você também pode editar e recompilar o arquivo badge.swf. Para obter detalhes, consulte "Modifique o arquivo badge.swf" na página 255.

Instale o aplicativo do AIR a partir de um link de instalação direta em uma página da Web

Depois de ter adicionado o link de instalação direta a uma página, o usuário pode instalar o aplicativo do AIR clicando no link no arquivo SWF.

1 Navegue até a página HTML em um navegador da Web que tenha Flash Player (versão 9 atualização 3 ou posterior no Windows e Mac OS ou versão 10 no Linux) instalada.

2 Na página da Web, clique no link no arquivo badge.swf.

- Se tiver instalado o runtime, passe para a próxima etapa.
- Se não tiver instalado o runtime, uma caixa de diálogo será exibida perguntando se você gostaria de instalá-lo. Instale o runtime (consulte "Instalação do Adobe AIR" na página 3) e continue com a etapa seguinte.

3 Na janela de instalação, deixe as configurações padrão selecionadas e clique em Continuar.

Em um computador Windows, o AIR faz automaticamente o seguinte:

- Instala o aplicativo em c:\Arquivos de Programas\
- Cria um atalho na área de trabalho para o aplicativo
- Cria um atalho no menu Iniciar
- Adiciona uma entrada para o aplicativo no Painel de Controle Adicionar ou Remover Programas

No Mac OS, o instalador adiciona o aplicativo ao diretório Aplicativos (por exemplo, no diretório /Aplicativos no Mac OS).

Em um computador Linux, o AIR faz automaticamente o seguinte:

- Instala o aplicativo na /saída.
- Cria um atalho na área de trabalho para o aplicativo
- Cria um atalho no menu Iniciar
- Adicione uma entrada para o aplicativo no gerenciador de pacotes do sistema

4 Selecione as opções desejadas e clique no botão Instalar.

5 Quando a instalação estiver concluída, clique em Concluir.

Modifique o arquivo badge.swf

O SDK do AIR e o SDK do Flex oferecem os arquivos de origem para o arquivo badge.swf. Esses arquivos estão incluídos na pasta samples/badge do SDK:

Arquivos de origem	Descrição
badge.fla	O arquivo de origem do Flash usado para compilar o arquivo badge.swf. O arquivo badge.fla é compilado em um arquivo do SWF 9 (que pode ser carregado no Flash Player).
AIRBadge.as	Uma classe do ActionScript 3.0 que define a classe base usada no arquivo basdge.fla.

Você pode utilizar o Flash Professional para projetar novamente a interface visual do arquivo badge.fla.

A função de construtor `AIRBadge()`, definida na classe AIRBadge, carrega o arquivo air.swf hospedado em http://airdownload.adobe.com/air/browserapi/air.swf. O arquivo air.swf inclui código para usar o recurso de instalação direta.

O método `onInit()` (na classe AIRBadge) é invocado quando o arquivo air.swf é carregado com sucesso:

```
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;
    }
}
```

O código define a variável global `_air` para a classe principal do arquivo air.swf carregado. Essa classe inclui os seguintes métodos públicos, que o arquivo badge.swf acessa para chamar a funcionalidade de instalação direta:

Método	Descrição
`getStatus()`	Determina se o runtime é instalado (ou pode ser instalado) no computador. Para obter detalhes, consulte "Verificar se o runtime está instalado" na página 257. • `runtimeVersion` — Uma string que indica a versão do runtime (como `"1.0.M6"`) exigida pelo aplicativo a ser instalado.
`installApplication()`	Instala o aplicativo especificado na máquina do usuário. Para obter detalhes, consulte "Instalação de um aplicativo do AIR do navegador" na página 259. • `url` — Uma sequência de caracteres que define a URL. Você deve usar um caminho de URL absoluta, e não relativa. • `runtimeVersion` — Uma string que indica a versão do runtime (como `"2.5."`) exigida pelo aplicativo a ser instalado. • `arguments` — Argumentos a serem transmitidos ao aplicativo se ele for iniciado na instalação. O aplicativo é iniciado na instalação se o elemento `allowBrowserInvocation` é definido como `true` no arquivo do descritor do aplicativo. (Para obter mais informações sobre o arquivo do descritor do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208.) Se o aplicativo for iniciado como resultado de uma instalação direta do navegador (com o usuário optando por iniciar na instalação), o objeto NativeApplication do aplicativo despacha um objeto BrowserInvokeEvent apenas se argumentos forem transmitidos. Considere as implicações de segurança de dados que você transmite ao aplicativo. Para obter detalhes, consulte "Inicialização de um aplicativo do AIR instalado do navegador" na página 260.

As configurações para `url` e `runtimeVersion` são transmitidas no arquivo SWF pelas configurações do FlashVars na página HTML do contêiner.

Se o aplicativo for iniciado automaticamente na instalação, você poderá usar a comunicação LocalConnection para ter o aplicativo instalado. Entre em contato com o arquivo badge.swf na invocação. Para mais informações, consulte Comunicação com outras instâncias do Flash Player e AIR (para desenvolvedores em ActionScript) ou Communicating with other Flash Player and AIR instances (para desenvolvedores em HTML).

Você também pode chamar o método `getApplicationVersion()` do arquivo air.swf para verificar se um aplicativo está instalado. Você pode chamar esse método antes do processo de instalação do aplicativo ou após a instalação ser iniciada. Para obter detalhes, consulte "Verificar por uma página da Web se um aplicativo do AIR está instalado" na página 258.

Carregar o arquivo air.swf

Você pode criar seu próprio arquivo do SWF que usa as APIs no arquivo air.swf para interagir com o runtime e aplicativos do AIR de uma página da Web em um navegador. O arquivo air.swf é hospedado em http://airdownload.adobe.com/air/browserapi/air.swf. Para se referir às APIs do air.swf do seu arquivo SWF, carregue o arquivo air.swf no mesmo domínio de aplicativo do seu arquivo SWF. O código a seguir mostra um exemplo de como carregar o arquivo air.swf no domínio do aplicativo do arquivo SWF que está sendo carregado:

```
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;
}
```

Depois que o arquivo air.swf estiver carregado (quando o objeto Loader do objeto `contentLoaderInfo` enviar o evento `init`), você pode chamar APIs air.swf, descritas nas seções que seguem.

Nota: *O arquivo badge.swf, fornecido com o SDK do Flex e do AIR, carrega automaticamente o arquivo air.swf. Consulte "Usando o arquivo badge.swf para instalar um aplicativo do AIR" na página 253. As instruções desta seção se aplicam à criação do seu próprio arquivo do SWF que carrega o arquivo air.swf.*

Verificar se o runtime está instalado

Um arquivo SWF pode verificar se o runtime está instalado chamando o método `getStatus()` no arquivo air.swf carregado de http://airdownload.adobe.com/air/browserapi/air.swf. Para obter detalhes, consulte "Carregar o arquivo air.swf" na página 257.

Depois que o arquivo air.swf for carregado, o arquivo SWF poderá chamar o método `getStatus()` do arquivo air.swf como a seguir:

```
var status:String = airSWF.getStatus();
```

O método `getStatus()` retorna um dos seguintes valores de sequências de caracteres, com base no status do runtime no computador:

Valor de string	Descrição
`"available"`	O runtime pode ser instalado nesse computador, mas não está instalado no momento.
`"unavailable"`	O runtime não pode ser instalado neste computador.
`"installed"`	O runtime está instalado nesse computador.

O método `getStatus()` lança um erro se a versão necessária do Flash Player (versão 9 atualização 3 ou posterior no Windows e Mac OS ou versão 10 no Linux) não estiver instalada no navegador.

Verificar por uma página da Web se um aplicativo do AIR está instalado

Um arquivo SWF pode verificar se um aplicativo do AIR (com um ID de aplicativo e um ID de editor correspondentes) está instalado chamando o método `getApplicationVersion()` no arquivo air.swf carregado de http://airdownload.adobe.com/air/browserapi/air.swf. Para obter detalhes, consulte "Carregar o arquivo air.swf" na página 257.

Depois que o arquivo air.swf for carregado, o arquivo SWF poderá chamar o método `getApplicationVersion()` do arquivo air.swf como a seguir:

```
 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.
    }
}
```

O método `getApplicationVersion()` possui os seguintes parâmetros:

Parâmetros	Descrição
`appID`	O ID desse aplicativo. Para obter detalhes, consulte "id" na página 230.
`pubID`	O ID do editor do aplicativo. Para obter detalhes, consulte "publisherID" na página 239. Se o aplicativo e em questão não possuir um ID do editor, defina o parâmetro `pubID` para uma sequência de caracteres vazia ("").
`retorno de chamada`	Uma função de retorno de chamada para servir como a função do manipulador. O método getApplicationVersion() opera de modo assíncrono e ao detectar essa versão instalada (ou a falta de uma versão instalada), esse método de retorno de chamada é invocado. A definição do método de retorno de chamada deve incluir um parâmetro, uma sequência de caracteres, definida para a sequência de caracteres da versão do aplicativo instalado. Se o aplicativo não for instalado, um valor de nulo será transmitido à função, como ilustrado no exemplo de código anterior.

O método `getApplicationVersion()` lança um erro se a versão necessária do Flash Player (versão 9 atualização 3 ou posterior no Windows e Mac OS ou versão 10 no Linux) não estiver instalada no navegador.

Nota: *A partir do AIR 1.5.3, o ID do editor não é mais utilizado. Os IDs de publicação não são mais atribuídos a nenhum aplicativo automaticamente. Para manter a compatibilidade com versões anteriores, os aplicativos podem a continuar a especificar o ID do editor.*

Instalação de um aplicativo do AIR do navegador

Um arquivo SWF pode instalar um aplicativo do AIR chamando o método `installApplication()` no arquivo air.swf carregado de `http://airdownload.adobe.com/air/browserapi/air.swf`. Para obter detalhes, consulte "Carregar o arquivo air.swf" na página 257.

Depois que o arquivo air.swf for carregado, o arquivo SWF poderá chamar o método `installApplication()` do arquivo air.swf como no código a seguir:

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

O método `installApplication()` instala o aplicativo especificado na máquina do usuário. Esse método possui os seguintes parâmetros:

Parâmetro	Descrição
`url`	Uma sequência de caracteres que define a URL do arquivo AIR a instalar. Você deve usar um caminho de URL absoluta, e não relativa.
`runtimeVersion`	Uma sequência de caracteres que indica a versão do runtime (como "1.0") exigida pelo aplicativo a ser instalado.
`argumentos`	Uma matriz de argumentos a serem transmitidos ao aplicativo se ele for iniciado na instalação. Somente caracteres alfanuméricos são reconhecidos nos argumentos. Se for necessário passar outros valores, considere o uso de um esquema de codificação. O aplicativo é iniciado na instalação se o elemento `allowBrowserInvocation` é definido como `true` no arquivo do descritor do aplicativo. (Para obter mais informações sobre o arquivo do descritor do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208.) Se o aplicativo for iniciado como resultado de uma instalação direta do navegador (com o usuário optando por iniciar na instalação), o objeto NativeApplication do aplicativo despacha um objeto BrowserInvokeEvent apenas se argumentos tiverem sido transmitidos. Para obter detalhes, consulte "Inicialização de um aplicativo do AIR instalado do navegador" na página 260.

O método `installApplication()` pode operar apenas quando chamado no manipulador de eventos para um evento do usuário, como um clique do mouse.

O método `installApplication()` lança um erro se a versão necessária do Flash Player (versão 9 atualização 3 ou posterior no Windows e Mac OS ou versão 10 no Linux) não estiver instalada no navegador.

No Mac OS, para instalar uma versão atualizada de um aplicativo, o usuário precisa ter privilégios adequados do sistema para instalar no diretório do aplicativo (e privilégios administrativos se o aplicativo atualizar o runtime). No Windows, o usuário deve ter privilégios administrativos.

Você também pode chamar o método `getApplicationVersion()` do arquivo air.swf para verificar se um aplicativo já está instalado. Você pode chamar esse método antes que o processo de instalação do aplicativo seja iniciado ou após a instalação ser iniciada. Para obter detalhes, consulte "Verificar por uma página da Web se um aplicativo do AIR está instalado" na página 258. Depois que o aplicativo estiver em execução, ele pode se comunicar com o conteúdo do SWF no navegador usando a classe LocalConnection. Para mais informações, consulte Comunicação com outras instâncias do Flash Player e AIR (para desenvolvedores em ActionScript) ou Communicating with other Flash Player and AIR instances (para desenvolvedores em HTML).

Inicialização de um aplicativo do AIR instalado do navegador

Para usar o recurso de invocação do navegador (permitindo que ele seja iniciado do navegador), o arquivo do descritor do aplicativo de destino deve incluir a seguinte configuração:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Para obter mais informações sobre o arquivo do descritor do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208.

Um arquivo SWF no navegador pode iniciar um aplicativo do AIR chamando o método `launchApplication()` no arquivo air.swf carregado de http://airdownload.adobe.com/air/browserapi/air.swf. Para obter detalhes, consulte "Carregar o arquivo air.swf" na página 257.

Depois que o arquivo air.swf for carregado, o arquivo SWF poderá chamar o método `launchApplication()` do arquivo air.swf como no código a seguir:

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

O método `launchApplication()` é definido no nível superior do arquivo air.swf (carregado no domínio do aplicativo do arquivo SWF da interface do usuário). Chamar esse método faz com que o AIR inicie o aplicativo especificado (se ele for instalado e a invocação do navegador for permitida, pela configuração `allowBrowserInvocation` no arquivo do descritor do aplicativo). O método tem os seguintes parâmetros:

Parâmetro	Descrição
`appID`	O ID do aplicativo a ser iniciado. Para obter detalhes, consulte "id" na página 230.
`pubID`	O ID do editor do aplicativo a ser iniciado. Para obter detalhes, consulte "publisherID" na página 239. Se o aplicativo e em questão não possuir um ID do editor, defina o parâmetro `pubID` para uma sequência de caracteres vazia ("").
`argumentos`	Uma matriz de argumentos para transmitir ao aplicativo. O objeto NativeApplication do aplicativo despacha um evento BrowserInvokeEvent que possui uma propriedade de argumentos definida para essa matriz. Somente caracteres alfanuméricos são reconhecidos nos argumentos. Se for necessário passar outros valores, considere o uso de um esquema de codificação.

O método `launchApplication()` pode operar apenas quando chamado no manipulador de eventos para um evento do usuário, como um clique do mouse.

O método `launchApplication()` lança um erro se a versão necessária do Flash Player (versão 9 atualização 3 ou posterior no Windows e Mac OS ou versão 10 no Linux) não estiver instalada no navegador.

Se o elemento `allowBrowserInvocation` for definido como `false` no arquivo do descritor do aplicativo, chamar o método `launchApplication()` não terá efeito.

Antes de apresentar a interface do usuário para iniciar o aplicativo, você pode desejar chamar o método `getApplicationVersion()` no arquivo air.swf. Para obter detalhes, consulte "Verificar por uma página da Web se um aplicativo do AIR está instalado" na página 258.

Quando o aplicativo é invocado pelo recurso de invocação do navegador, o objeto NativeApplication do aplicativo despacha um objeto BrowserInvokeEvent. Para obter detalhes, consulte Invocação de um aplicativo do AIR do navegador (para desenvolvedores em ActionScript) ou Invoking an AIR application from the browser (para desenvolvedores em HTML).

Se você usa o recurso de invocação do navegador, certifique-se de considerar implicações de segurança. Estas implicações são descrita em Invocação de um aplicativo do AIR do navegador (para desenvolvedores em ActionScript) e Invoking an AIR application from the browser (para desenvolvedores em HTML).

Depois que o aplicativo estiver em execução, ele pode se comunicar com o conteúdo do SWF no navegador usando a classe LocalConnection. Para mais informações, consulte Comunicação com outras instâncias do Flash Player e AIR (para desenvolvedores em ActionScript) ou Communicating with other Flash Player and AIR instances (para desenvolvedores em HTML).

Nota: *A partir do AIR 1.5.3, o ID do editor não é mais utilizado. Os IDs de publicação não são mais atribuídos a nenhum aplicativo automaticamente. Para manter a compatibilidade com versões anteriores, os aplicativos podem a continuar a especificar o ID do editor.*

Capítulo 17: Atualização de aplicativos do AIR

Os usuários podem instalar ou atualizar um aplicativo do AIR clicando duas vezes no arquivo AIR no computador ou a partir do navegador (usando o recurso de instalação direta). O aplicativo de instalação do Adobe® AIR™ gerencia a instalação, alertando o usuário se ele estiver atualizando um aplicativo já existente.

No entanto, também é possível que um aplicativo instalado se atualize automaticamente para uma nova versão usando a classe Updater. (Um aplicativo instalado pode detectar que uma nova versão está disponível para download e instalação.) A classe Updater inclui um método `update()` que permite apontar para um arquivo AIR no computador do usuário e atualizar para essa versão. Seu aplicativo deve ser compactado como um arquivo AIR, para usar a classe Updater. Aplicativos compactados como executável nativo ou pacote devem utilizar os meios de atualização fornecidos pela plataforma nativa.

As IDs do aplicativo e do editor de um arquivo AIR de atualização devem corresponder às do aplicativo a ser atualizado. I ID do publicador é obtido do certificado de assinatura. Tanto a atualização quanto o aplicativo a serem usados devem ser assinados com o mesmo certificado.

Para o AIR 1.5.3 ou superior, a arquivo de descrição do aplicativo inclui um elemento `<publisherID>`. Você deverá usar este elemento se houver versões do seu aplicativo desenvolvidas com o uso do AIR 1.5.2 ou superior. Para obter mais informações, consulte “publisherID” na página 239.

A partir do AIR 1.1, é possível migrar um aplicativo para usar um novo certificado de autenticação de código. A migração de um aplicativo para usar uma nova assinatura envolve assinar o arquivo AIR de atualização com os certificados novo e original. A migração de certificado é um processo unidirecional. Após a migração, somente os arquivos do AIR assinados com o novo certificado (ou com ambos) serão reconhecidos como atualizações de uma instalação existente.

Gerenciar a atualização de aplicativos pode ser complicado. O AIR 1.5 inclui os novos aplicativos *estrutura de atualização para do Adobe®* AIR™. Essa estrutura fornece APIs para auxiliar os desenvolvedores a fornecer bons recursos de atualização em aplicativos do AIR.

Você pode usar a migração de certificado para mudar de um certificado autoassinado para um certificado comercial de autenticação de código ou de um certificado autoassinado ou comercial para outro. Caso você não migre o certificado, os usuários existentes deverão remover a versão atual do seu aplicativo antes de instalar a nova. Para obter mais informações, consulte “Alteração de certificados” na página 198.

É uma boa prática incluir um mecanismo de atualização no seu aplicativo. Se você criar uma nova versão do aplicativo, o mecanismo de atualização pode solicitar que o usuário instale uma nova versão.

O programa de instalação do aplicativo do AIR cria arquivos de registro quando um aplicativo do AIR é instalado, atualizado ou removido. Você pode consultar estes arquivos de registro para ajudar a determinar a causa de problemas de instalação. Consulte Registros de instalação.

Nota: *As novas versões do runtime do Adobe AIR podem incluir versões atualizadas do WebKit. Uma versão atualizada do WebKit pode resultar em alterações inesperadas no conteúdo em HTML em um aplicativo do AIR implementado. Estas alterações podem exigir que você atualize o seu aplicativo. Um mecanismo de atualização pode informar ao usuário sobre a nova versão do aplicativo. Para mais informações, consulte Sobre o ambiente HTML (para desenvolvedores em ActionScript) ou About the HTML environment (para desenvolvedores em HTML).*

Sobre atualização de aplicativos

A classe Updater (no pacote flash.desktop) inclui um método, `update()`, que você pode usar para atualizar o aplicativo em execução no momento com uma versão diferente. Por exemplo, se o usuário tem uma versão do arquivo AIR ("Sample_App_v2.air") localizada na área de trabalho, o seguinte código atualiza o aplicativo.

Exemplo do 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);
```

Exemplo do 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 um aplicativo usar a classe Updater, o usuário ou o aplicativo deve baixar a versão atualizada do arquivo AIR no computador. Para obter mais informações, consulte "Download de um arquivo AIR no computador do usuário" na página 265.

Resultados de chamar o método Updater.update()

Quando um aplicativo no runtime chama o método `update()`, o runtime fecha o aplicativo e tenta instalar a nova versão do arquivo AIR. O runtime verifica se o ID do aplicativo e o ID do editor especificadas no arquivo AIR correspondem às IDs do aplicativo e do editor do aplicativo que está chamando o método `update()`. (Para obter informações sobre o ID do aplicativo e o ID do editor, consulte "Arquivos descritores do aplicativo do AIR" na página 208.) Ele também verifica se a string de versão corresponde à string `version` passada para o método `update()`. Se a instalação for concluída com êxito, o runtime abrirá a nova versão do aplicativo. Do contrário (se a instalação não for concluída), ele reabrirá a versão existente (pré-instalação) do aplicativo.

No Mac OS, para instalar uma versão atualizada de um aplicativo, o usuário deve ter privilégios adequados do sistema para instalar no diretório do aplicativo. No Windows e no Linux, um usuário precisa de privilégios administrativos.

Se a versão atualizada do aplicativo exigir uma versão atualizada do runtime, a nova versão do runtime será instalada. Para atualizar o runtime, o usuário deve ter privilégios administrativos no computador.

Durante o teste de um aplicativo usando o ADL, se o método `update()` for chamado, será gerada uma exceção do runtime.

Sobre a string de versão

A sequência de caracteres especificada como o parâmetro `version` do método `update()` deve corresponder à string no elemento `version` ou `versionNumber` do arquivo de descrição do aplicativo para o arquivo AIR a ser instalado. É necessário especificar o parâmetro `version` por motivo de segurança. Ao solicitar que o aplicativo verifique o número da versão do arquivo AIR, o aplicativo não instalará uma versão mais antiga de forma inadvertida. (Uma versão mais antiga do aplicativo pode conter uma vulnerabilidade de segurança que foi corrigida no aplicativo instalado no momento.) O aplicativo também deve verificar a string de versão no arquivo AIR com a string de versão no aplicativo instalado para impedir ataques de downgrade.

Antes do AIR 2.5, a sequência de caracteres da versão pode ser de qualquer formato. Por exemplo, pode ser "2.01" ou "versão 2". No AIR 2.5 ou posterior, a sequência de caracteres da versão deve ser uma sequência de até três números de três dígitos separados por pontos. Por exemplo, ".0", "1.0" e "67.89.999" são todos os números de versão válidos. Você deve validar a sequência da versão de atualização antes de atualizar o aplicativo.

Se um aplicativo do Adobe AIR baixa um arquivo AIR pela web, é recomendável ter um mecanismo através do qual o serviço da web possa notificar o aplicativo sobre a versão que está sendo baixada. O aplicativo poderá então usar essa string como o parâmetro `version` do método `update()`. Se o arquivo AIR for obtido por algum outro meio, no qual a versão do arquivo é desconhecida, o aplicativo do AIR poderá examiná-lo para determinar a informação de versão. (Um arquivo AIR consiste em um arquivo compactado no formato ZIP, e o arquivo de descrição do aplicativo é o segundo registro no arquivo.)

Para obter detalhes sobre o arquivo do descritor do aplicativo, consulte "Arquivos descritores do aplicativo do AIR" na página 208.

Marcando o fluxo de trabalho para atualizações do aplicativo

A publicação de atualizações na forma ad hoc complica as tarefas de gerenciamento de versões de vários aplicativos e também faz monitoramento de dificuldade de datas de validade do certificado. Os certificados podem expirar antes que você possa publicar uma atualização

O runtime do Adobe AIR trata uma atualização de aplicativo publicada sem assinatura de migração como um novo aplicativo. Os usuários devem desinstalar seu aplicativo do AIR atual antes que possam instalar a atualização do aplicativo.

Para resolver o problema, carregue cada aplicativo atualizado com o certificado mais recente para uma URL de implementação separada. Inclua um mecanismo que o lembre de aplicar assinaturas de migração quando o certificado estiver dentro do período de prorrogação de 180 dias. Para obter mais informações, consulte "Assinatura de uma versão atualizada de um aplicativo do AIR" na página 203.

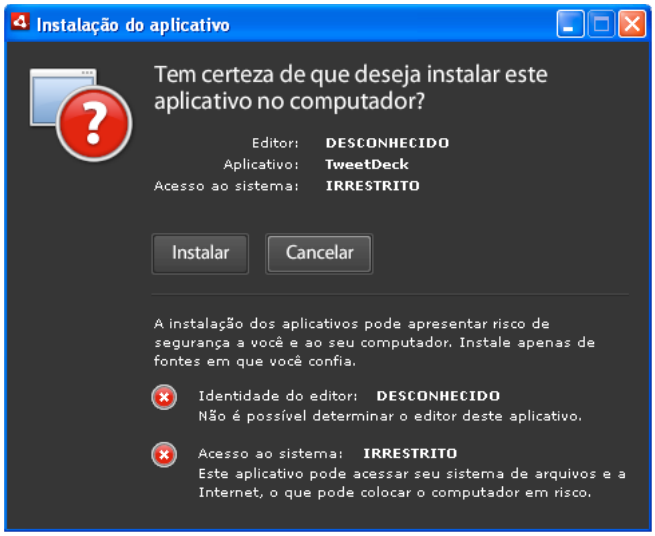
Consulte "Comandos do ADT" na página 168 para obter mais informações sobre como aplicar assinaturas.

Realize as seguintes tarefas para simplificar o processo de aplicação das assinaturas de migração:

- Carregue cada aplicativo atualizado para uma URL de implementação separada.
- Carregue o arquivo XML descritor e o certificado mais recente para a atualização para a mesma URL.
- Marque o aplicativo atualizado com o certificado mais recente.
- Aplique uma assinatura de migração para o aplicativo atualizado com o certificado usado para assinar a versão anterior localizada em uma URL diferente.

Apresentação de uma interface de usuário de atualização do aplicativo personalizado

O AIR vem com uma interface de atualização padrão:



Essa interface sempre é usada quando o usuário instala uma versão de um aplicativo em uma máquina pela primeira vez. No entanto, você pode definir sua própria interface para usá-la em ocorrências subsequentes. Se seu aplicativo definir uma interface de atualização personalizada, especifique um elemento `customUpdateUI` no arquivo de descritor do aplicativo para o aplicativo instalado no momento:

```
<customUpdateUI>true</customUpdateUI>
```

Quando o aplicativo é instalado e o usuário abre um arquivo AIR com um ID de aplicativo e um ID de editor que correspondem aos do aplicativo instalado, o runtime abre o aplicativo em vez no instalador de aplicativo padrão do AIR. Para obter mais informações, consulte “customUpdateUI” na página 219.

O aplicativo pode decidir, quando executado (quando o objeto `NativeApplication.nativeApplication` despacha um evento `load`), se o aplicativo deve ser atualizado (usando a classe Updater). Se ele optar pela atualização, poderá apresentar ao usuário sua própria interface de instalação (que é diferente da interface padrão que está sendo executada).

Download de um arquivo AIR no computador do usuário

Para utilizar a classe Updater, primeiro o usuário ou o aplicativo deve salvar um arquivo AIR localmente no computador do usuário.

Nota: *O AIR 1.5 inclui uma estrutura de atualização, que auxilia desenvolvedores no fornecimento de bons recursos de atualização em aplicativos do AIR. Usar essa estrutura pode ser bem mais fácil que usar o método* `update()` *da classe Update diretamente. Para obter detalhes, consulte “Uso da estrutura de atualização” na página 269.*

O código abaixo lê um arquivo AIR a partir de uma URL (http://example.com/air/updates/Sample_App_v2.air) e salva o arquivo no diretório de armazenamento do aplicativo.

Exemplo do 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.");
}
```

Exemplo do 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 obter mais informações, consulte:

- Fluxo de trabalho de leitura e gravação de arquivos (para desenvolvedores em ActionScript)
- Workflow for reading and writing files (para desenvolvedores em HTML)

Verificar se um aplicativo está sendo executado pela primeira vez

Após atualizar um aplicativo, você pode exibir ao usuário uma mensagem de "introdução" ou de "boas-vindas". Após a inicialização, o aplicativo verifica se está sendo executado pela primeira vez para determinar se deve exibir a mensagem.

Nota: *O AIR 1.5 inclui uma estrutura de atualização, que auxilia desenvolvedores no fornecimento de bons recursos de atualização em aplicativos do AIR. Essa estrutura fornece métodos fáceis para verificar se uma versão de um aplicativo está sendo executada pela primeira vez. Para obter detalhes, consulte "Uso da estrutura de atualização" na página 269.*

Uma forma de fazer isso é salvar um arquivo no diretório de armazenamento do aplicativo depois de inicializá-lo. Sempre que o aplicativo é inicializado, deve averiguar se esse arquivo existe. Se o arquivo não existir, isso indica que o aplicativo está sendo executado pela primeira vez para o usuário atual. Se o arquivo existir, o aplicativo já foi executado pelo menos uma vez. Se o arquivo existir e contiver um número de versão mais antigo que o atual, você saberá que o usuário está executando a nova versão pela primeira vez.

O exemplo a seguir do Flex demonstra o conceito:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    title="Sample Version Checker Application"
    applicationComplete="system extension()">
    <mx:Script>
        <![CDATA[
            import flash.filesystem.*;
            public var file:File;
            public var currentVersion:String = "1.2";
            public function system extension():void {
                file = File.applicationStorageDirectory;
                file = file.resolvePath("Preferences/version.txt");
                trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            private function checkVersion():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.READ);
                var reversion:String = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    log.text = "You have updated to version " + currentVersion + ".\n";
```

```
                } else {
                    saveFile();
                }
                log.text += "Welcome to the application.";
            }
            private function firstRun():void {
                log.text = "Thank you for installing the application. \n"
                    + "This is the first time you have run it.";
                saveFile();
            }
            private function saveFile():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        ]]>
    </mx:Script>
    <mx:TextArea ID="log" width="100%" height="100%" />
</mx:WindowedApplication>
```

O exemplo a seguir demonstra o conceito no 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>
```

Se o seu aplicativo salva dados localmente (como no diretório de armazenamento do aplicativo), convém verificar se existem dados já salvos (de versões anteriores) após a primeira execução.

Uso da estrutura de atualização

O gerenciamento de atualizações para aplicativos pode ser entediante. A *estrutura de atualização para aplicativos AdobeAIR*fornece APIs que permitem que os desenvolvedores forneçam recursos de atualização robustos em aplicativos do AIR. A estrutura de atualização do AIR realiza as seguintes tarefas para os desenvolvedores:

- Verifique periodicamente se há atualizações em um determinado intervalo ou quando o usuário solicita
- Baixe arquivos do AIR (atualizações) de uma fonte da Web
- Alerte o usuário na primeira execução da versão recém-instalada
- Confirme que o usuário deseja procurar atualizações
- Exiba as informações sobre a nova versão de atualização para o usuário
- Exiba o andamento do download e as informações de erro para o usuário

A estrutura de atualização do AIR fornece um exemplo de interface de usuário para seu aplicativo. Ela fornece ao usuário informações básicas e opções de configuração para atualizações do aplicativo. Seu aplicativo também pode definir a interface de usuário personalizada para uso com a estrutura de atualização.

A estrutura de atualização do AIR permite armazenar informações sobre a versão de atualização de um aplicativo do AIR em arquivos de configuração XML simples. Na maioria dos aplicativos, a definição desses arquivos de configuração para incluir código básico fornecem uma boa funcionalidade de atualização para o usuário final.

Mesmo sem usar a estrutura de atualização, o Adobe AIR inclui uma classe Updater que os aplicativos do AIR podem usar para atualizar para novas versões. A classe Updater permite que um aplicativo seja atualizado para uma versão contida em um arquivo do AIR no computador do usuário. No entanto, o gerenciamento de atualização pode envolver mais que simplesmente atualizar o aplicativo com base em um arquivo AIR armazenado localmente.

Arquivos de estrutura de atualização do AIR

A estrutura de atualização do AIR está incluída no diretório frameworks/libs/air do AIR 2 SDK. Isto inclui os seguintes arquivos:

- applicationupdater.swc — Define a funcionalidade básica da biblioteca de atualização para ser usada no ActionScript. Esta versão não contém nenhuma interface do usuário.
- applicationupdater.swc — Define a funcionalidade básica da biblioteca de atualização para ser usada no JavaScript. Esta versão não contém nenhuma interface do usuário.
- ApplicationUpdater_UI.swc — Define a funcionalidade básica da biblioteca de atualização da versão 4 do Flex, incluindo uma interface de usuário que o seu aplicativo pode usar para exibir opções de atualização.
- ApplicationUpdater_UI.swc — Define a funcionalidade básica da biblioteca de atualização na versão JavaScript, incluindo uma interface de usuário que o seu aplicativo pode usar para exibir opções de atualização.

Para mais informações, consulte as seguintes seções:

- "Configuração do ambiente de desenvolvimento em Flex" na página 270
- "Inclusão de arquivos de estrutura em um aplicativo do AIR baseado em HTML" na página 270
- "Exemplo básico: Uso da versão ApplicationUpdaterUI" na página 271

Configuração do ambiente de desenvolvimento em Flex

Os arquivos SWC contidos no diretório frameworks/libs/air do AIR 2 SDK definem classes que você pode usar no desenvolvimento em Flex e Flash.

Para usar a estrutura de atualização ao compilar com o Flex SDK, inclua o arquivo ApplicationUpdater.swc ou o ApplicationUpdater_UI.swc na chamada do compilador amxmlc. No exemplo a seguir, o compilador carrega o arquivo ApplicationUpdater.swc no subdiretório lib do diretório Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

No exemplo a seguir, o compilador carrega o arquivo ApplicationUpdater_UI.swc no subdiretório lib do diretório Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Ao desenvolver usando o construtor Flash Builder, adicione o arquivo SWC à guia Caminho da biblioteca das configurações do caminho de criação de Flex na caixa de diálogo Propriedades.

Assegure-se de copiar os arquivos SWC no diretório que você usará como referência no compilador amxmlc (usando o Flex SDK) ou Flash Builder.

Inclusão de arquivos de estrutura em um aplicativo do AIR baseado em HTML

O diretório frameworks/html da estrutura de atualização inclui estes arquivos:

- applicationupdater.swf — Define a funcionalidade básica da biblioteca de atualização, sem qualquer interface do usuário
- applicationupdater_ui.swf — Define a funcionalidade básica da biblioteca de atualização, incluindo uma interface de usuário que seu aplicativo usa para exibir opções de atualização

O código JavaScript nos aplicativos do AIT podem usar classes definidas nos arquivos SWF.

Para usar a estrutura de atualização, inclua o arquivo applicationupdater.swf ou o applicationupdater_ui.swf no diretório do aplicativo (ou um subdiretório). Em seguida, no arquivo HTML que usará a estrutura (em código JavaScript), inclua uma tag `script` que carregue o arquivo:

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

Ou use essa tag `script` para carregar o arquivo applicationupdater_ui.swf:

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

A API definida nesses dois arquivos é descrita no restante deste documento.

Exemplo básico: Uso da versão ApplicationUpdaterUI

A versão ApplicationUpdaterUI da estrutura de atualização fornece uma interface básica que pode ser facilmente usada no seu aplicativo. A seguir há um exemplo básico:

Primeiro, crie um aplicativo do AIR que chame a estrutura de atualização:

1 Se seu aplicativo for um aplicativo do AIR baseado em HTML, carregue o arquivo applicationupdaterui.swf:

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 Na lógica de programação do aplicativo do AIR, instancie um objeto do ApplicationUpdaterUI.

No ActionScript, use o seguinte código:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

No JavaScript, use o seguinte código:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Você pode adicionar esse código a uma função de inicialização executada quando o aplicativo é carregado.

3 Crie um arquivo de texto updateConfig.xml e adicione o seguinte a ele:

```
<?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 o elemento `URL` do arquivo updateConfig.xml para que corresponda à localização eventual do arquivo de descritor de atualização do seu servidor da Web (veja o próximo procedimento).

O `delay` é o número de dias que o aplicativo aguarda entre as verificações de atualizações.

4 Adicione o arquivo updateConfig.xml ao diretório do projeto do seu aplicativo do AIR.

5 Faça com que o objeto updater referencie o arquivo updateConfig.xml e chame o método `initialize()` do objeto.

No ActionScript, use o seguinte código:

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

No JavaScript, use o seguinte código:

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 Crie uma segunda versão do aplicativo do AIR que tenha uma versão diferente do primeiro aplicativo. (A versão é especificada no arquivo de descritor do aplicativo, no elemento `version`.)

Em seguida, adicione a versão de atualização do aplicativo do AIR ao servidor da Web:

1 Coloque a versão de atualização do arquivo AIR no servidor da Web.

2 Crie um arquivo de texto updateDescriptor.2.5.xml e adicione o seguinte conteúdo a ele:

```
<?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 `versionNumber`, `URL` e `description` do arquivo updateDescriptor.xml para que corresponda ao seu arquivo do AIR. Este formato de descritor de atualização é usado por aplicativos que utilizam a estrutura de atualização incluída com o AIR 2.5 SDK (e posterior).

3 Crie um arquivo de texto updateDescriptor.1.0.xml e adicione o seguinte conteúdo a ele:

```
<?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 `version`, `URL` e `description` do arquivo updateDescriptor.xml para que corresponda ao seu arquivo AIR. Este formato de descritor de atualização é usado por aplicativos que utilizam a estrutura de atualização incluída com o AIR 2 SDK (e anterior).

Nota: *A criação deste segundo arquivo descritor de atualização só é necessária quando você oferece suporte para a atualização para aplicativos criados antes do AIR 2.5.*

4 Inclua os arquivos updateDescriptor.2.5.xml e updateDescriptor.1.0.xml no mesmo diretório do servidor Web que contém o arquivo AIR de atualização.

Esse é um exemplo básico, mas fornece a funcionalidade de atualização suficiente para vários aplicativos. O restante deste documento descreve como usar a estrutura de atualização para atender melhor suas necessidades.

Para obter outro exemplo de uso da estrutura de atualização, consulte o aplicativo de amostra a seguir no Adobe AIR Developer Center:

- Estrutura de atualização em um aplicativo com base no Flash (http://www.adobe.com/go/learn_air_qs_update_framework_flash_br)

Atualização para o AIR 2.5

Conforme as regras para assinatura de números de versão para aplicativos modificados no AIR 2.5, a estrutura de atualização do AIR 2 não pode analisar as informações de versão em um descritor de aplicativo do AIR 2.5. Esta incompatibilidade significa que você deve atualizar seu aplicativo para usar a nova estrutura atualização ANTES de atualizar seu aplicativo para usar o AIR 2.5 SDK. Assim, a atualização do seu aplicativo para AIR 2.5 ou posterior a partir de qualquer versão do AIR antes do 2.5 requer DUAS atualizações. A primeira atualização deve usar o namespace AIR 2 e incluir a biblioteca de estrutura de atualização AIR 2.5 (você ainda pode criar o pacote de aplicativos usando o AIR 2.5 SDK). A segunda atualização pode usar o namespace AIR 2.5 e incluir os novos recursos do seu aplicativo.

Você também pode ter a atualização intermediária sem fazer nada exceto a atualização para seu aplicativo do AIR 2.5 usando a classe Updater do AIR diretamente.

O exemplo a seguir ilustra como atualizar um aplicativo da versão 1.0 para 2.0. A versão 1.0 utiliza o antigo namespace 2.0. A versão 2.0 utiliza o namespace 2.5 e tem novos recursos implementados usando as APIs do AIR 2.5.

1 Crie uma versão intermediária do aplicativo, versão 1.0.1, baseado na versão 1.0 do aplicativo.

   a Use a estrutura Application Updater do AIR 2.5 ao criar o aplicativo.

   Nota: *Use* `applicationupdater.swc` *ou* `applicationupdater_ui.swc` *para aplicativos AIR com base na tecnologia Flash, e* `applicationupdater.swf` *ou* `applicationupdater_ui.swf` *para aplicativos AIR com base HTML.*

   b Crie um arquivo descritor de atualização para versão 1.0.1 usando o antigo namespace e a versão conforme demonstrado abaixo:

```
<?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 Crie a versão 2.0 do aplicativo que usa o namespace 2.5 e as APIs do AIR 2.5.

3 Crie um descritor de atualização para atualizar o aplicativo a partir da versão 1.0.1 para a 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>
```

Definição dos arquivos de descritor de atualização a acréscimo do arquivo do AIR ao servidor da Web

Quando você usa a estrutura de atualização do AIR, define informações básicas sobre a atualização disponível em arquivos de descritor de atualização, armazenados no servidor da Web. Um arquivo de descritor de atualização é um arquivo XML simples. A estrutura de atualização incluída no aplicativo verifica esse arquivo para ver se uma nova versão foi carregada.

O formato do arquivo de descritor de atualização mudou para AIR 2.5. O novo formato usa um namespace diferente. O namespace original é “http://ns.adobe.com/air/framework/update/description/1.0”. O namespace do AIR 2.5 é “http://ns.adobe.com/air/framework/update/description/2.5”.

Os aplicativos do AIR criados antes do AIR 2.5 só podem ler a versão do descritor de atualização 1.0. Os aplicativos do AIR criados que usam a estrutura do atualizador incluída no AIR 2.5 ou posterior só podem ler o descritor de atualização da versão 2.5. Devido a essa incompatibilidade de versão, muitas vezes você precisa criar dois arquivos de descritor de atualização. A lógica de atualização na versões AIR 2.5 de seu aplicativo deve baixar um descritor de atualização que use o novo formato. As versões anteriores do aplicativo do AIR devem continuar a usar o formato original. Ambos os arquivos devem ser modificados para cada atualização que você lançar (até parar de fornecer suporte para as versões criadas antes do AIR 2.5).

O arquivo de descritor de atualização contém os seguintes dados:

- `versionNumber` — A nova versão do aplicativo do AR. Use o elemento `versionNumber` nos descritores de atualização usados para atualizar os aplicativos do AIR 2.5. O valor deve ser a mesma sequência de caracteres usada no elemento `versionNumber` do novo arquivo de descritor do aplicativo do AIR. Se o número da versão no arquivo de descritor de atualização não corresponder ao da versão no arquivo AIR de atualização, a estrutura de atualização lançará uma exceção.

- `version`— A nova versão do aplicativo do AIR. Use o elemento `version` nos descritores de atualização usados para atualizar aplicativos criados antes do AIR 2.5. O valor deve ser a mesma sequência de caracteres usada no elemento `version` do novo arquivo de descritor de aplicativo do AIR. Se a versão do arquivo de descritor de atualização não corresponder à versão do arquivo AIR, a estrutura de atualização lançará uma exceção.
- `versionLabel` — A sequência de caracteres da versão legível destinada a ser exibida aos usuários. A `versionLabel` é opcional, mas só pode ser especificada em arquivos de descritor de atualização de versão 2.5. Use-a se utilizar uma `versionLabel` no descritor do aplicativo e configure-a para o mesmo valor.
- `url` – O local do arquivo AIR de atualização. Esse arquivo contém a versão de atualização do aplicativo do AIR.
- `description` — Detalhes relativos à nova versão. Essas informações podem ser exibidas para o usuário durante o processo de atualização.

Os elementos `version` e `url` são obrigatórios. O elemento `description` é opcional.

Este é um exemplo de arquivo de descritor de atualização da versão 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>
```

Este é um exemplo de arquivo de descritor de atualização 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>
```

Se desejar definir a tag `description` usando vários idiomas, use vários elementos `text` que definam o 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>
```

Coloque o arquivo de descritor de atualização no servidor da Web, juntamente com o arquivo de atualização do AIR.

O diretório modelo incluído com o descritor de atualização inclui exemplos dos arquivos descritores de atualização. Eles incluem versões com um idioma ou vários idiomas.

Instanciação de um objeto atualizador

Após carregar a estrutura de atualização do AIR no seu código (consulte "Configuração do ambiente de desenvolvimento em Flex" na página 270 e "Inclusão de arquivos de estrutura em um aplicativo do AIR baseado em HTML" na página 270), você precisa instanciar um objeto atualizador, conforme a seguir:

Exemplo do ActionScript:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

Exemplo do JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

O código anterior usa a classe ApplicationUpdater (que não fornece interface de usuário). Se você deseja usar a classe ApplicationUpdaterUI (que fornece uma interface de usuário), use o seguinte.

Exemplo do ActionScript:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Exemplo do JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Os exemplos de código restantes neste documento supõem que você tenha instanciado um objeto atualizador `appUpdater`.

Definição das configurações de atualização

O ApplicationUpdater e o ApplicationUpdaterUI podem ser configurados por meio de um arquivo de configuração fornecido com o aplicativo ou por meio de ActionScript ou JavaScript no aplicativo.

Definição das configurações de atualização em um arquivo de configuração XML

O arquivo de configuração de atualização é um arquivo XML. Ele pode conter os seguintes elementos:

- `updateURL`— Uma sequência de caracteres. Representa a localização do descritor de atualização no servidor remoto. Qualquer localização de URLRequest válida é permitida. Você deve definir a propriedade `updateURL` pelo arquivo de configuração ou por script (consulte "Definição dos arquivos de descritor de atualização a acréscimo do arquivo do AIR ao servidor da Web" na página 273). Defina essa propriedade antes de usar o atualizador (antes de chamar o método `initialize()` do objeto atualizador, descrito em "Inicialização da estrutura de atualização" na página 278).
- `delay`— Um número. Representa um intervalo de tempo fornecido em dias (valores como `0,25` são permitidos) para verificação de atualizações. Um valor de 0 (que é o valor padrão) especifica que o atualizador não realiza uma verificação automática periódica.

O arquivo de configuração do ApplicationUpdaterUI pode conter o seguinte elemento, além dos elementos `updateURL` e `delay`:

- `defaultUI`: Uma lista de elementos `dialog`. Cada elemento `dialog` tem um atributo `name` que corresponde à caixa de diálogo na interface do usuário. Cada elemento `dialog` tem um atributo `visible` que define se a caixa de diálogo está visível. O valor padrão é `true`. Valores possíveis para o atributo `name` são:
  - `"checkForUpdate"` — Corresponde às caixas de diálogo Verificar atualizações, Nenhuma atualização e Erro de atualização.
  - `"downloadUpdate"` — Corresponde à caixa de diálogo Fazendo download de atualização.
  - `"downloadProgress"` — Corresponde às caixas de diálogo Download em andamento e Erro de download.
  - `"installUpdate"` — Corresponde à caixa de diálogo Instalar atualização.
  - `"fileUpdate"` — Corresponde às caixas de diálogo Atualização de arquivo, Não atualização de arquivo e Erro de arquivo
- `"unexpectedError"` — Corresponde à caixa de diálogo Erro inesperado

Quando definida como `false`, a caixa de diálogo correspondente não aparece como parte do procedimento de atualização.

Este é um exemplo do arquivo de configuração para a estrutura 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>
```

Este é um exemplo do arquivo de configuração para a estrutura ApplicationUpdaterUI, que inclui uma definição para o 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>
```

Aponte a propriedade `configurationFile` para o local desse arquivo:

Exemplo do ActionScript:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

Exemplo do JavaScript:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

O diretório modelo da estrutura de atualização inclui um exemplo de arquivo de configuração, o config-template.xml.

Definição das configurações de atualização do código ActionScript ou JavaScript

Esses parâmetros de configuração também podem ser definidos usando código no aplicativo, como a seguir:

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

As propriedades do objeto atualizador são `updateURL` e `delay`. Essas propriedades definem as mesmas configurações dos elementos `updateURL` e `delay` no arquivo de configuração: o URL e o arquivo de descritor de atualização e o intervalo de verificação de atualizações. Se você especificar as configurações *and* de um arquivo de configuração no código, todas as propriedades definidas usando o código terão precedência sobre as configurações correspondentes no arquivo de configuração.

Você deve definir a propriedade `updateURL` por meio do arquivo de configuração ou por meio de script (consulte "Definição dos arquivos de descritor de atualização a acréscimo do arquivo do AIR ao servidor da Web" na página 273) antes de usar o atualizador (antes de chamar o método `initialize()` do objeto atualizador, descrito em "Inicialização da estrutura de atualização" na página 278).

A estrutura ApplicationUpdaterUI define essas propriedades adicionais do objeto atualizador:

- `isCheckForUpdateVisible` — Corresponde às caixas de diálogo Verificar atualizações, Nenhuma atualização e Erro de atualização.
- `isDownloadUpdateVisible` — Corresponde à caixa de diálogo Fazendo download de atualização.

- `isDownloadProgressVisible` —Corresponde às caixas de diálogo Download em andamento e Erro de download.
- `isInstallUpdateVisible` — Corresponde à caixa de diálogo Instalar atualização.
- `isFileUpdateVisible` — Corresponde às caixas de diálogo Atualização de arquivo, Não atualização de arquivo e Erro de arquivo
- `isUnexpectedErrorVisible` — Corresponde à caixa de diálogo Erro inesperado

Cada propriedade corresponde a uma ou mais caixa de diálogo da interface de usuário ApplicationUpdaterUI. Cada propriedade é um valor booliano com um valor padrão `true`. Quando definida como `false`, a caixa de diálogo correspondente não aparece como parte do procedimento de atualização.

Essas propriedades de caixa de diálogo substituem as configurações no arquivo de configuração de atualização.

O processo de atualização

A estrutura de atualização do AIR completa o processo de atualização nas seguintes etapas:

1. A inicialização do atualizador verifica se foi realizada uma verificação de atualização no intervalo de tempo definido (consulte “Definição das configurações de atualização” na página 275). Se estiver faltando uma verificação de atualização, o processo de atualização continuará.
2. O atualizador baixa e interpreta o arquivo de descritor de atualização.
3. O atualizador baixa o arquivo AIR de atualização.
4. O atualizador instala a versão atualizada do aplicativo.

O objeto atualizador despacha eventos na conclusão de cada uma das etapas. Na versão do ApplicationUpdater, você pode cancelar os eventos que indicam a conclusão bem-sucedida de uma etapa no processo. Se você cancelar um desses eventos, a próxima etapa do processo será cancelada. Na versão do ApplicationUpdaterUI, o atualizador apresenta uma caixa de diálogo permitindo que o usuário cancele ou continue para a próxima etapa do processo.

Se você cancelar o evento, poderá chamar métodos do objeto atualizador para retomar o processo.

Conforme a versão do ApplicationUpdater do atualizador progride pelo processo de atualização, ela registra seu estado atual em uma propriedade `currentState`. Essa propriedade é definida como uma sequência de caracteres com os seguintes valores possíveis:

- `"UNINITIALIZED"` — O atualizador não foi inicializado.
- `"INITIALIZING"` — O atualizador está sendo inicializado.
- `"READY"` — O atualizador foi inicializado
- `"BEFORE_CHECKING"` — O atualizador ainda não verificou se existe um arquivo de descritor de atualização.
- `"CHECKING"` — O atualizador está verificando se existe um arquivo de descritor de atualização.
- `"AVAILABLE"` — O arquivo de descritor de atualização está disponível.
- `"DOWNLOADING"` — O atualizador está baixando o arquivo AIR.
- `"DOWNLOADED"` — O atualizador baixou o arquivo AIR.
- `"INSTALLING"` — O atualizador está instalando o arquivo AIR.
- `"PENDING_INSTALLING"` — O atualizador foi inicializado e não há atualizações pendentes.

Alguns métodos do objeto atualizador só serão executados se o atualizador estiver em determinado estado.

Inicialização da estrutura de atualização

Depois de definir as propriedades de configuração (consulte "Exemplo básico: Uso da versão ApplicationUpdaterUI" na página 271), chame o método `initialize()` para inicializar a atualização:

```
appUpdater.initialize();
```

Esse método faz o seguinte:

- Ele inicializa a estrutura de atualização, instalando de forma silenciosa e síncrona todas as atualizações pendentes. É necessário para chamar esse método durante a inicialização do aplicativo, pois ele pode reiniciar o aplicativo quando chamado.
- Ele verifica se existe uma atualização adiada e a instala.
- Se houver um erro durante o processo de atualização, ele limpa o arquivo de atualização e as informações de versão da área de armazenamento do aplicativo.
- Se o tempo limite tiver expirado, o processo de atualização e iniciado. Caso contrário, ele inicia o timer.

Chamar esse método pode resultar no despacho dos seguintes eventos pelo objeto atualizador:

- `UpdateEvent.INITIALIZED` — Despachado quando a inicialização é concluída.
- `ErrorEvent.ERROR` — Despachado quando há um erro na inicialização.

No despacho do evento `UpdateEvent.INITIALIZED`, o processo de atualização é concluído.

Quando você chama o método `initialize()`, o atualizador inicia o processo de atualização e conclui todas as etapas com base na configuração de tempo do timer. No entanto, você também pode iniciar o processo de atualização a qualquer momento chamando o método `checkNow()` do objeto atualizador:

```
appUpdater.checkNow();
```

Esse método não faz nada se o processo de atualização já estiver em execução. Caso contrário, ele começa o processo de atualização.

O objeto atualizador pode despachar o seguinte evento como resultado de chamar o método `checkNow()`:

- Evento `UpdateEvent.CHECK_FOR_UPDATE`, antes de ele tentar baixar o arquivo de descritor de atualização.

Se você cancelar o evento `checkForUpdate`, poderá chamar o método `checkForUpdate()` do objeto atualizador. (Consulte a próxima seção.) Se você não cancelar o evento, o processo de atualização continuará para verificar se há arquivo de descritor de atualização.

Gerenciamento do processo de atualização na versão ApplicationUpdaterUI

Na versão ApplicationUpdaterUI, o usuário pode cancelar o processo pelos botões Cancelar das caixas de diálogo da interface de usuário. Além disso, você pode cancelar de forma programática o processo de atualização chamando o método `cancelUpdate()` do objeto ApplicationUpdaterUI.

Você pode definir as propriedades do objeto ApplicationUpdaterUI ou definir elementos no arquivo de configuração de atualização para especificar quais confirmações de caixa de diálogo o atualizador exibe. Para obter detalhes, consulte "Definição das configurações de atualização" na página 275.

Gerenciamento do processo de atualização na versão ApplicationUpdater

Você pode chamar o método `preventDefault()` dos objetos de evento despachados pelo objeto ApplicationUpdater para cancelar etapas do processo de atualização (consulte "O processo de atualização" na página 277). Cancelar o comportamento padrão fornece ao seu aplicativo uma chance de exibir uma mensagem ao usuário perguntando se ele deseja continuar.

As seções a seguir descrevem como continuar o processo de atualização quando uma etapa do processo é cancelada.

Download e interpretação do arquivo de descritor de atualização

O objeto ApplicationUpdater despacha o evento `checkForUpdate` antes do processo de atualização ser iniciado, logo antes de o atualizador tentar baixar o arquivo de descritor de atualização. Se você cancelar o comportamento padrão do evento `checkForUpdate`, o atualizador não baixará o arquivo de descritor de atualização. Você pode chamar o método `checkForUpdate()` para retomar o processo de atualização:

```
appUpdater.checkForUpdate();
```

Chamar o método `checkForUpdate()` faz com que o atualizador baixe e interprete o arquivo do descritor do aplicativo de forma assíncrona. Como resultado de chamar o método `checkForUpdate()`, o objeto atualizador poderá despachar os seguintes eventos:

- `StatusUpdateEvent.UPDATE_STATUS` — O atualizador baixou e interpretou o arquivo de descritor de atualização com êxito. Esse evento tem estas propriedades:
  - `available` — Um valor booliano. Configure para `true` se existir uma versão disponível diferente do aplicativo atual; caso contrário `false` (a versão é a mesma).
  - `version` — Uma sequência de caracteres. A versão do arquivo de descritor de aplicativo do arquivo de atualização
  - `details` — Uma matriz. Se não houver versões localizadas da descrição, essa matriz retornará uma sequência de caracteres vazia (`""`) como primeiro elemento e a descrição como segundo elemento.

    Se houver várias versões da descrição (no arquivo de descritor de atualização), a matriz conterá várias submatrizes. Cada matriz tem dois elementos: o primeiro é um código de idiomas (como `"en"`) e o segundo é a descrição correspondente (uma sequência de caracteres) para o idioma. Consulte "Definição dos arquivos de descritor de atualização a acréscimo do arquivo do AIR ao servidor da Web" na página 273.
- `StatusUpdateErrorEvent.UPDATE_ERROR` — Ocorreu e o atualizador não pôde baixar ou interpretar o arquivo de descritor de eventos.

Download do arquivo de atualização do AIR

O objeto ApplicationUpdater despacha o evento `updateStatus` depois que o atualizador baixa e interpreta com êxito o arquivo de descritor de atualização. O comportamento padrão é começar a baixar o arquivo de atualização, se ele estiver disponível. Se você cancelar o comportamento padrão, poderá chamar o método `downloadUpdate()` para retomar o processo de atualização.

```
appUpdater.downloadUpdate();
```

Chamar esse método faz com que o atualizador baixe de forma assíncrona a versão de atualização do arquivo do AIR.

O método `downloadUpdate()` pode despachar os seguintes eventos:

- `UpdateEvent.DOWNLOAD_START` — Foi estabelecida a conexão com o servidor. Quando você usa a biblioteca ApplicationUpdaterUI, esse evento exibe uma caixa de diálogo com uma barra de progresso para controlar o andamento do download.
- `ProgressEvent.PROGRESS` — Despachado periodicamente conforme o download do arquivo progride.

- `DownloadErrorEvent.DOWNLOAD_ERROR` — Despachado se houver um erro na conexão ou no download do arquivo de atualização. Também é despachado para status de HTTP inválidos (como " 404 - Arquivo não encontrado"). Esse evento tem uma propriedade `errorID`, um inteiro que define informações de erro adicionais. Uma propriedade `subErrorID` adicional pode conter mais informações de erro.
- `UpdateEvent.DOWNLOAD_COMPLETE` — O atualizador baixou e interpretou o arquivo de descritor de atualização com êxito. Se você não cancelar esse evento, a versão do ApplicationUpdater continuará a instalar a versão de atualização. Na versão do ApplicationUpdaterUI, o usuário visualiza uma caixa de diálogo que fornece a opção de continuar.

Atualização do aplicativo

O objeto ApplicationUpdater despacha o evento `downloadComplete` quando o download do arquivo de atualização é concluído. Se você cancelar o comportamento padrão, poderá chamar o método `installUpdate()` para retomar o processo de atualização.

```
appUpdater.installUpdate(file);
```

Chamar esse método faz com que o atualizador instale uma versão de atualização do arquivo AIR. O método inclui um parâmetro, `file`, que é um objeto File que referencia o arquivo AIR a ser usado como atualização.

O objeto ApplicationUpdater pode despachar e evento `beforeInstall` como resultado de chamar o método `installUpdate()`:

- `UpdateEvent.BEFORE_INSTALL` — Despachado antes de instalar a atualização. Às vezes, é útil impedir a instalação da atualização nesse momento, para que o usuário possa concluir o trabalho atual antes de a atualização continuar. Chamar o método `preventDefault()` do objeto Event adia a instalação até o próximo reinício, e nenhum processo de atualização adicional pode ser iniciado. (Isso inclui atualizações que resultariam de chamar o método `checkNow()` ou de verificações periódicas.)

Instalação de um arquivo AIR arbitrário

Você pode chamar o método `installFromAIRFile()` para instalar a versão de atualização para instalar de um arquivo AIR no computador do usuário.

```
appUpdater.installFromAIRFile();
```

Esse método faz com que o atualizador instale uma versão de atualização do aplicativo a partir do arquivo AIR.

O método `installFromAIRFile()` pode despachar os seguintes eventos:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS` — Despachado depois que ApplicationUpdater valida com êxito o arquivo envaido usando o método `installFromAIRFile()`. Esse evento tem as seguintes propriedades:
  - `available` — Definida como `true` se houver uma versão diferente da versão do aplicativo atual; `false` caso contrário (as versões são as mesmas).
  - `version` — A string que representa a nova versão disponível.
  - `path` — Representa o caminho nativo do arquivo de atualização.

  Você pode cancelar esse evento se a propriedade disponível do objeto StatusFileUpdateEvent estiver definida como `true`. O cancelamento do evento impede a continuidade da atualização. Chame o método `installUpdate()` para continuar a atualização cancelada.
- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR` — Ocorreu um erro e o atualizador não pôde instalar o aplicativo do AIR.

Cancelamento do processo de atualização

Você pode chamar o método `cancelUpdate()` para cancelar o processo de atualização:

```
appUpdater.cancelUpdate();
```

Esse método cancela os downloads pendentes, excluindo arquivos baixados incompletos, e reinicia o timer de verificação periódica.

O método não faz nada se o objeto atualizador estiver sendo inicializado.

Localização da interface ApplicationUpdaterUI

A classe ApplicationUpdaterUI fornece uma interface de usuário padrão para o processo de atualização. Isso inclui caixas de diálogo que permitem que o usuário inicie o processo, cancele o processo e realize outras ações relacionadas.

O elemento `description` do arquivo de descritor de atualização permite definir a descrição do aplicativo em vários idiomas. Use vários elementos `text` que definem atributos `lang`, como a seguir:

```
<?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>
```

A estrutura de atualização usa a descrição mais adequada para a cadeia de localização do usuário. Para obter mais informações, consulte Definição do arquivo de descritor de atualização a acréscimo do arquivo AIR ao servidor da Web.

Desenvolvedores de Flex podem adicionar diretamente um novo idioma ao grupo `"ApplicationUpdaterDialogs"`.

Desenvolvedores de JavaScript podem chamar o método `addResources()` do objeto atualizador. Esse método adiciona dinamicamente um novo conjunto de recursos para um idioma. O conjunto de recursos define sequências de caracteres localizadas para um idioma. Essas sequências de caracteres são usadas em vários campos de texto de caixa de diálogo.

Desenvolvedores de JavaScript podem usar a propriedade `localeChain` da classe ApplicationUpdaterUI para definir a cadeia de localização usada pela interface do usuário. Geralmente, somente desenvolvedores de JavaScript (HTML) usam essa propriedade. Desenvolvedores de Flex usam o ResourceManager para gerenciar a cadeia de localização.

Por exemplo, o código de JavaScript a seguir define grupos de recursos para romano e 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 obter detalhes, consulte a descrição do método `addResources()` da classe ApplicationUpdaterUI na referência de idiomas.

Capítulo 18: Visualização do código-fonte

Da mesma forma que o usuário pode visualizar o código-fonte para uma página HTML em um navegador, os usuários podem visualizar o código-fonte de um aplicativo do AIR baseado em HTML. O SDK do Adobe® AIR® inclui um arquivo AIRSourceViewer.js de JavaScript que você pode usar em seu aplicativo para revelar facilmente o código-fonte para os usuários finais.

Carregamento, configuração e abertura do Visualizador de Código-Fonte

O código do Visualizador de Código-Fonte está incluído em um arquivo JavaScript, AIRSourceViewer.js, que, por sua vez, está incluído no diretório de estruturas do SDK do AIR. Para usar o Visualizador do Código-Fonte no seu aplicativo, copie o arquivo AIRSourceViewer.js para o diretório do projeto do seu aplicativo e carregue o arquivo através da tag de script no arquivo HTML principal no seu aplicativo:

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

O arquivo AIRSourceViewer.js define uma classe, o SourceViewer, que você pode acessar do código do JavaScript chamando `air.SourceViewer`.

A classe SourceViewer define três métodos: `getDefault()`, `setup()` e `viewSource()`.

Método	Descrição
`getDefault()`	Um método estático. Retorna uma instância do SourceViewer, que você pode usar para chamar os outros métodos.
`setup()`	Aplica as configurações ao Visualizador do Código-Fonte. Para obter detalhes, consulte "Configuração do Visualizador do Código-Fonte" na página 282.
`viewSource()`	Abre uma nova janela na qual o usuário pode navegar e abrir os arquivos de origem do aplicativo de host.

Nota: *O código que usa o Visualizador do Código-Fonte deve estar em uma caixa de proteção do aplicativo (em um arquivo no diretório do aplicativo).*

Por exemplo, o código JavaScript a seguir instancia um objeto do Visualizador do Código-Fonte e abre a janela do Visualizador que lista todos os arquivos de origem:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Configuração do Visualizador do Código-Fonte

O método `config()` aplica determinadas configurações ao Visualizador do Código-Fonte. Esse método adota um parâmetro: `configObject`. O objeto `configObject` tem propriedades que definem as configurações para o Visualizador do Código-Fonte. As propriedades são `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` e `typesToAdd`.

default
Um string que especifica o caminho relativo ao arquivo inicial a ser exibido no Visualizador do Código-Fonte.

Por exemplo, o código JavaScript a seguir abre a janela do Visualizador do Código-Fonte com o arquivo index.html como arquivo inicial mostrado:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

Uma sequência de strings que especifica os arquivos ou diretórios a serem excluídos da listagem do Visualizador do Código-Fonte. Os caminhos se referem ao diretório do aplicativo. Os caracteres curinga não recebem suporte.

Por exemplo, o código JavaScript a seguir abre a janela do Visualizador do Código-Fonte que lista todos os arquivos de origem, exceto para o arquivo AIRSourceViewer.js, e os arquivos nos subdiretórios Imagens e sons:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

Uma matriz que inclui dois números, especificando as coordenadas x e y iniciais da janela do Visualizador do Código-Fonte.

Por exemplo, o código de JavaScript a seguir abre a janela do Visualizador do Código-Fonte nas coordenadas da tela [40, 60] (X = 40, Y = 60):

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

Um valor booliano que especifica se o Visualizador do Código-Fonte deve ser uma janela modal (verdadeiro) ou não modal (falso). Por padrão, a janela do Visualizador do Código-Fonte é modal.

Por exemplo, o código de JavaScript a seguir abre a janela do Visualizador do Código-Fonte de tal forma que o usuário possa interagir com a janela do Visualizador do Código-Fonte e as janelas de qualquer aplicativo:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

Uma matriz de strings que especifica os tipos de arquivo a serem incluídos na listagem do Visualizador do Código-Fonte, além dos tipos padrão inclusos.

Por padrão, o Visualizador do Código-Fonte lista os seguintes tipos de arquivos:

- Arquivos de texto — TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Arquivos de imagem — JPG, JPEG, PNG, GIF

  Se nenhum valor for especificado, todos os tipos padrão são incluídos (exceto aqueles especificados na propriedade `typesToExclude`).

Por exemplo, o código de JavaScript a seguir abre a janela do Visualizador do Código-Fonte que inclui os arquivos VCF e VCARD:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToAdd = ["text.vcf", "text.vcard"];
viewer.viewSource(configObj);
```

Para cada tipo de arquivo que você listar, especifique "text" (para os arquivos do tipo texto) ou "image" (para arquivos do tipo imagem).

typesToExclude
Uma matriz de strings que especifica os tipos de arquivo a serem excluídos do Visualizador do Código-Fonte.

Por padrão, o Visualizador do Código-Fonte lista os seguintes tipos de arquivos:

- Arquivos de texto — TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Arquivos de imagem — JPG, JPEG, PNG, GIF

Por exemplo, o código de JavaScript a seguir abre a janela do Visualizador do Código-Fonte sem listar os arquivos GIF ou XML:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

Para cada tipo de arquivo que você listar, especifique `"text"` (para os arquivos do tipo texto) ou `"image"` (para arquivos do tipo imagem).

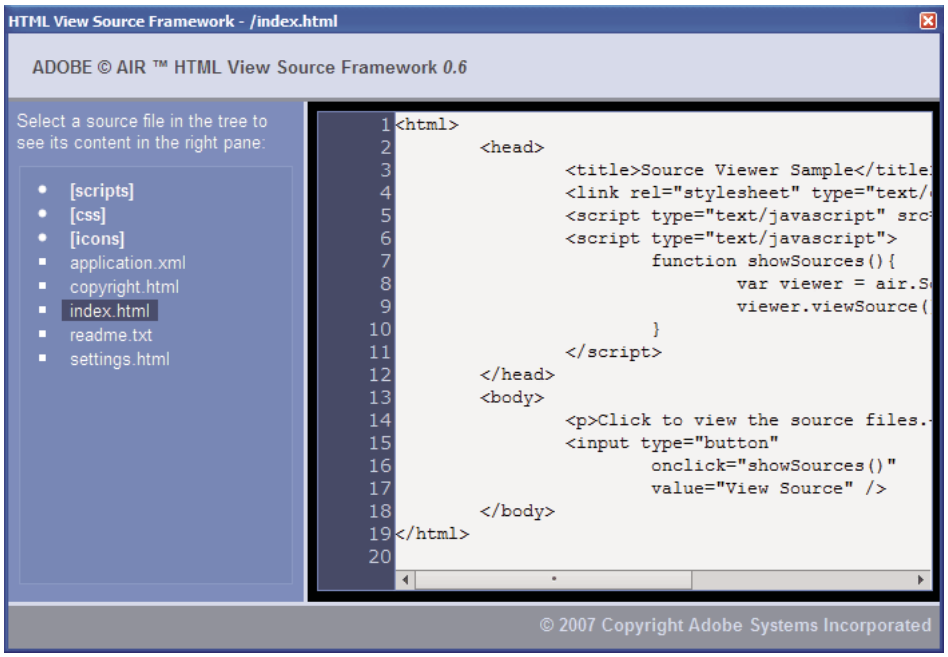
Abertura do Visualizador do Código-Fonte

Você deve incluir um elemento da interface do usuário, como um link, botão ou comando do menu, que chama o código do Visualizador do Código-Fonte quando o usuário o selecionar. Por exemplo, o aplicativo simples a seguir abre o Visualizador do Código-Fonte quando o usuário clica em um link:

```
<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>
```

Interface do usuário do Visualizador do Código-Fonte

Quando o aplicativo chama o método `viewSource()` de um objeto SourceViewer, o aplicativo do AIR abre uma janela do Visualizador do Código-fonte. A janela inclui uma lista de arquivos e diretórios de origem (à esquerda) e uma área de exibição que mostra o código-fonte para o arquivo selecionado (à direita):



Os diretórios são listados entre parênteses. O usuário pode clicar em um parêntese para expandir ou encolher a listagem de um diretório.

O Visualizador do Código-Fonte pode exibir o código-fonte para os arquivos de texto com extensões reconhecidas (como, HTML, JS, TXT, XML e outros) ou para arquivos de imagem com extensões reconhecidas (JPG, JPEG, PNG e GIF). Se o usuário selecionar um arquivo que não tem uma extensão de arquivo reconhecida, será exibida uma mensagem de erro ("Não é possível recuperar o conteúdo de texto deste tipo de arquivo").

Os arquivos de origem que são excluídos através do método `setup()` não são listados (consulte “Carregamento, configuração e abertura do Visualizador de Código-Fonte” na página 282).

Capítulo 19: Depuração com o AIR HTML Introspector

O SDK do Adobe® AIR® inclui um arquivo AIRIntrospector.js de JavaScript que você pode incluir em seu aplicativo para ajudar a depurar os aplicativos baseados em HTML.

Sobre o AIR Introspector

O Adobe AIR HTML/JavaScript Application Introspector (chamado AIR HTML Introspector) fornece recursos úteis para ajudar no desenvolvimento e na depuração do aplicativo baseados em HTML:

- Inclui uma ferramenta de introspecção que lhe permite apontar para um elemento da interface do usuário no aplicativo e vê suas propriedades de marcação e DOM.
- Além disso, inclui um console para enviar referências de objetos para introspeção, e você pode ajustar valores de propriedade e executar o código de JavaScript. Você também pode serializar objetos no console, que o limita na edição dos dados. Também é possível copiar e salvar texto a partir do console.
- Inclui uma visualização de árvore para as propriedades e funções DOM.
- Com isso, você poderá editar os atributos e nós de texto para os elementos DOM.
- Ele lista os links, estilos, imagens e arquivos JavaScript carregados no seu aplicativo.
- Assim, você poderá ver o código-fonte inicial de HTML e o código-fonte de marcação atual para a interface do usuário.
- Ele permite acessar os arquivos no diretório do aplicativo. (Esse recurso só está disponível para o console do AIR HTML Introspector aberto para a caixa de proteção do aplicativo. Não disponível para os consoles abertos para o conteúdo que não está na caixa de proteção do aplicativo.)
- Ele inclui um visualizar para os objetos XMLHttpRequest e suas propriedades, incluindo as propriedades `responseText` e `responseXML` (quando disponível).
- Você pode pesquisar o texto correspondente no código e nos arquivos de origem.

Carregamento do código do AIR Introspector

O código do AIR Introspector está incluído em um arquivo JavaScript, AIRIntrospector.js, que, por sua vez, está incluído no diretório de estruturas do SDK do AIR. Para usar o AIR Introspector no seu aplicativo, copie o arquivo AIRIntrospector.js para o diretório do projeto do seu aplicativo e carregue o arquivo através da tag de script no arquivo HTML principal no seu aplicativo:

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

Também inclua o arquivo em todos os arquivos HTML que correspondem a janelas nativas diferentes no seu aplicativo.

Importante: *Inclua o arquivo AIRIntrospector.js apenas ao desenvolver e depurar o aplicativo. Remova-o no aplicativo empacotador do AIR que você distribua.*

O arquivo AIRIntrospector.js define uma classe, Console, que você pode acessar do código do JavaScript chamando `air.Introspector.Console`.

Nota: *O código que usa o AIR Introspector deve estar em uma caixa de proteção do aplicativo (em um arquivo no diretório do aplicativo).*

Inspeção de um objeto na guia Console

A classe Console define cinco métodos: `log()`, `warn()`, `info()`, `error()` e `dump()`.

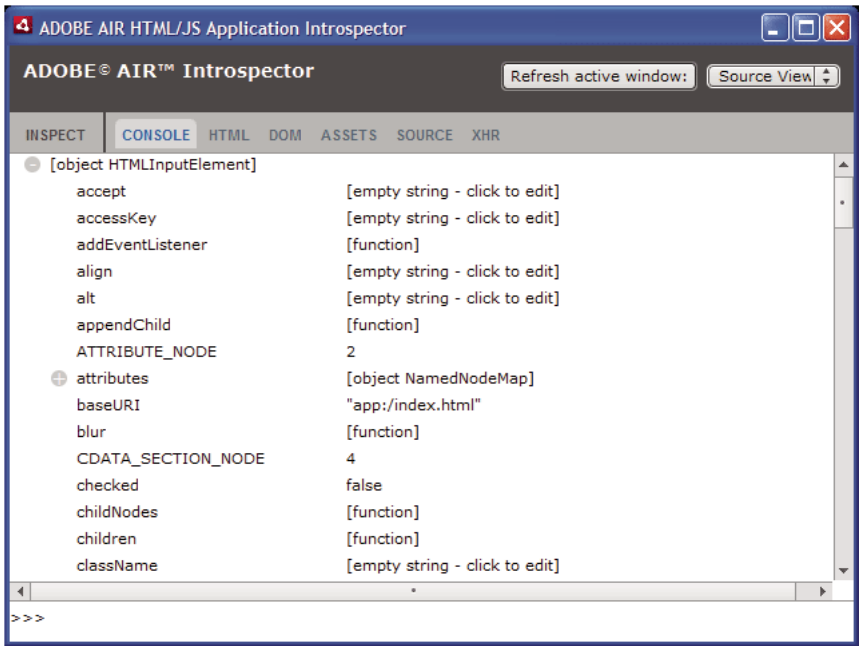
Os métodos `log()`, `warn()`, `info()` e `error()` permitem que você envie um objeto à guia Console. O método mais básico de todos é o método `log()`. O código a seguir envia um objeto simples, representado pela variável `test`, para a guia Console:

```
var test = "hello";
air.Introspector.Console.log(test);
```

No entanto, é mais útil enviar um objeto complexo para a guia Console. Por exemplo, a página HTML a seguir inclui um botão (`btn1`) que chama uma função que envia o próprio objeto do botão para a guia Console:

```
<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>
```

Ao clicar no botão, a guia Console exibe o objeto btn1, e você pode expandir a visualização de árvore do objeto para inspecionar suas propriedades:



Você pode editar uma propriedade do objeto clicando na lista à direita do nome da propriedade e modificando a listagem do texto.

Os métodos `info()`, `error()` e `warn()` são parecidos com o método `log()`. No entanto, ao chamar esses métodos, o Console exibe um ícone no começo da linha:

Método	Ícone
`info()`	
`error()`	
`warn()`	

Os métodos `log()`, `warn()`, `info()` e `error()` enviam uma referência apenas para um objeto real, por isso as propriedades disponíveis são as únicas no momento da visualização. Se desejar serializar o objeto real, use o método `dump()`. O método tem dois parâmetros:

Parâmetro	Descrição
`dumpObject`	O objeto a ser serializado.
`levels`	O número máximo de níveis a serem examinado na árvore do objeto (além do nível de raiz). O valor padrão é 1 (indicando que um é mostrado um nível além do nível de raiz da árvore). Este parâmetro é opcional.

Chamar o método `dump()` serializa um objeto antes de enviá-lo para a guia Console, de forma que você não possa editar as propriedades dos objetos. Por exemplo, considere o seguinte código:

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

Ao executar este código, o Console exibe o objeto `testObject` e suas propriedades, mas você não pode editar os valores da propriedade no Console.

Configuração do AIR Introspector

Você pode configurar o console definindo as propriedades da variável global `AIRIntrospectorConfig`. Por exemplo, o código de JavaScript a seguir configura o AIR Introspector para empacotar colunas com 100 caracteres:

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

Certifique-se de definir as propriedades da variável `AIRIntrospectorConfig` antes de carregar o arquivo AIRIntrospector.js (através de uma tag `script`).

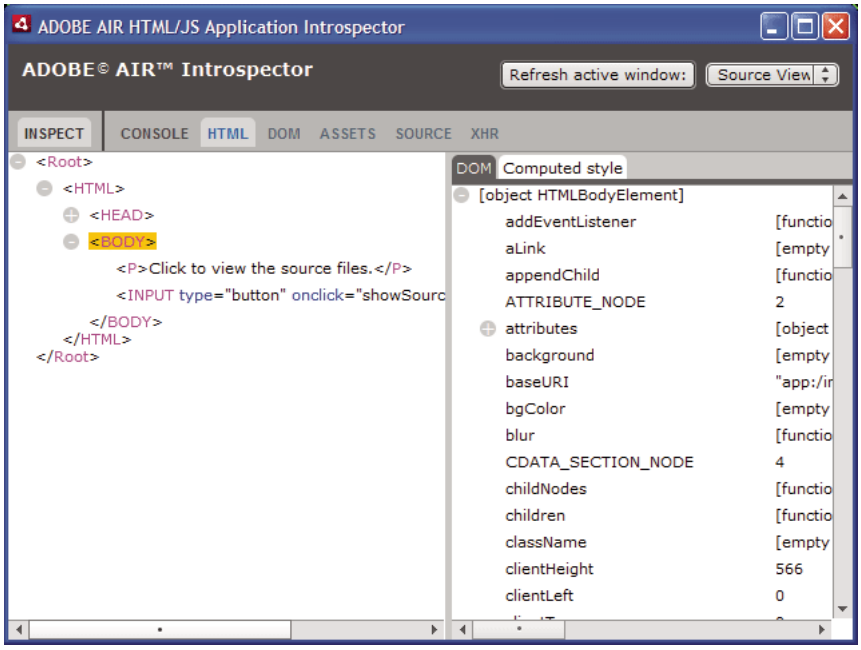
Há oito propriedades na variável `AIRIntrospectorConfig`:

Propriedade	Valor padrão	Descrição
`closeIntrospectorOnExit`	`true`	Define que a janela Inspector feche quando todas as outras janelas do aplicativo estão fechadas.
`debuggerKey`	123 (a tecla F12)	O código da tecla para o atalho no teclado para mostrar e ocultar a janela do AIR Introspector.
`debugRuntimeObjects`	`true`	Define que o Introspector expanda os objetos do runtime além dos objetos definidos no JavaScript.
`flashTabLabels`	`true`	Define que as guias Console e XMLHttpRequest pisquem, indicando quando ocorre uma mudança nelas (por exemplo, quando o texto é registrado nessas guias).
`introspectorKey`	122 (a tecla F11)	O código de tecla para o atalho no teclado para abrir o painel Inspect.
`showTimestamp`	`true`	Define que a guia Console exiba os carimbos de data/hora no começo de cada linha.
`showSender`	`true`	Define que a guia Console exiba as informações no objeto que está enviando a mensagem no começo de cada linha.
`wrapColumns`	2000	O número de colunas nas quais os arquivos de origem são empacotados.

Interface do AIR Introspector

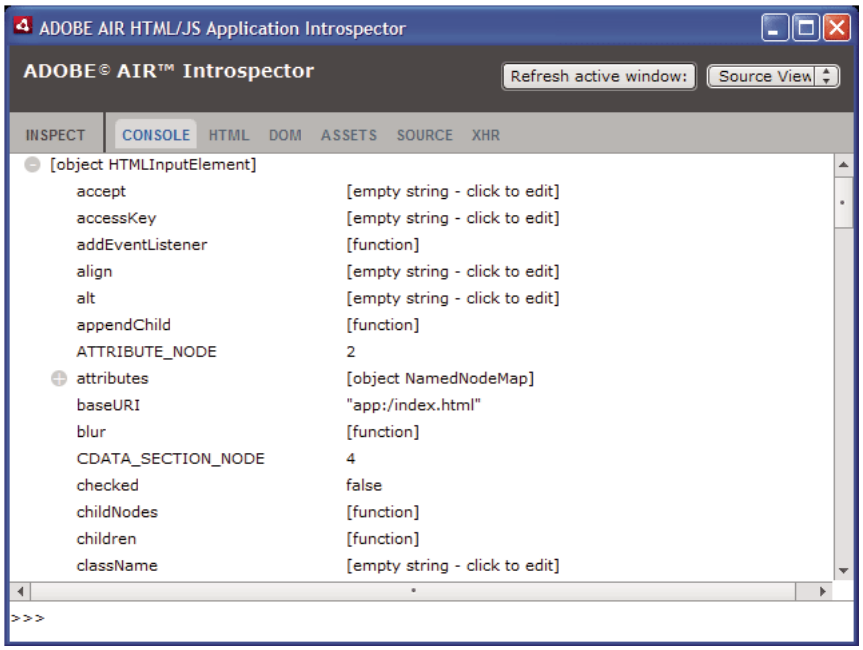
Para abrir a janela do AIR Introspector ao depurar o aplicativo, pressione a tecla F12 ou chame um dos métodos da classe Console (consulte “Inspeção de um objeto na guia Console” na página 287). Você pode configurar a "hot key" como uma tecla que não seja a tecla F12; consulte “Configuração do AIR Introspector” na página 289.

A janela do AIR Introspector tem seis guias — Console, HTML, DOM, Ativos, Código-Fonte e XHR — como mostrado na ilustração a seguir:



A guia Console

A guia Console exibe os valores das propriedades passadas como parâmetros para um dos métodos da classe air.Introspector.Console. Para obter detalhes, consulte “Inspeção de um objeto na guia Console” na página 287.

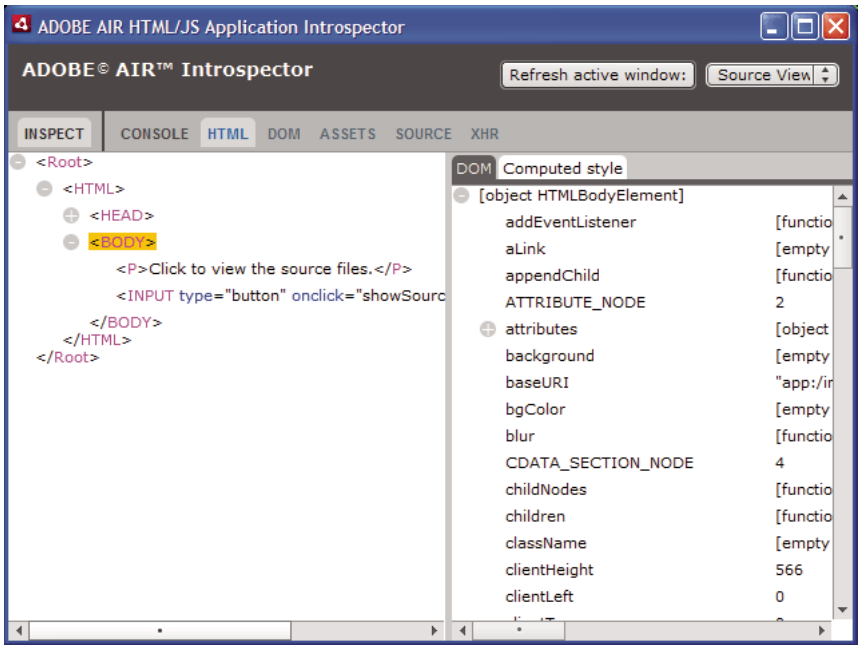


- Para limpar o console, clique com o botão direito do mouse no texto e selecionar Limpar console.

- Para salvar texto na guia Console em um arquivo, clique com o botão direito na guia Console e selecione Salvar console em arquivo.
- Para salvar texto na guia Console na área de transferência, clique com o botão direito na guia Console e selecione Salvar console na área de transferência. Para copiar apenas o texto na área de transferência, clique com o botão direito no texto e selecione Copiar.
- Para salvar texto na classe Console em um arquivo, clique com o botão direito na guia Console e selecione Salvar console em arquivo.
- Para pesquisar texto correspondente exibido na guia, clique em CTRL+F no Windows ou em Command+F no Mac OS. (Os nós de árvore que são visíveis não são pesquisados.)

A guia HTML

A guia HTML permite visualizar todo o DOM de HTML em uma estrutura de árvore. Clique em um elemento para exibir suas propriedades no lado direito da guia. Clique nos ícones + e - para expandir e recolher um nó na árvore.



Você pode editar qualquer atributo ou elemento de texto na guia HTML e o valor editado se reflete no aplicativo.

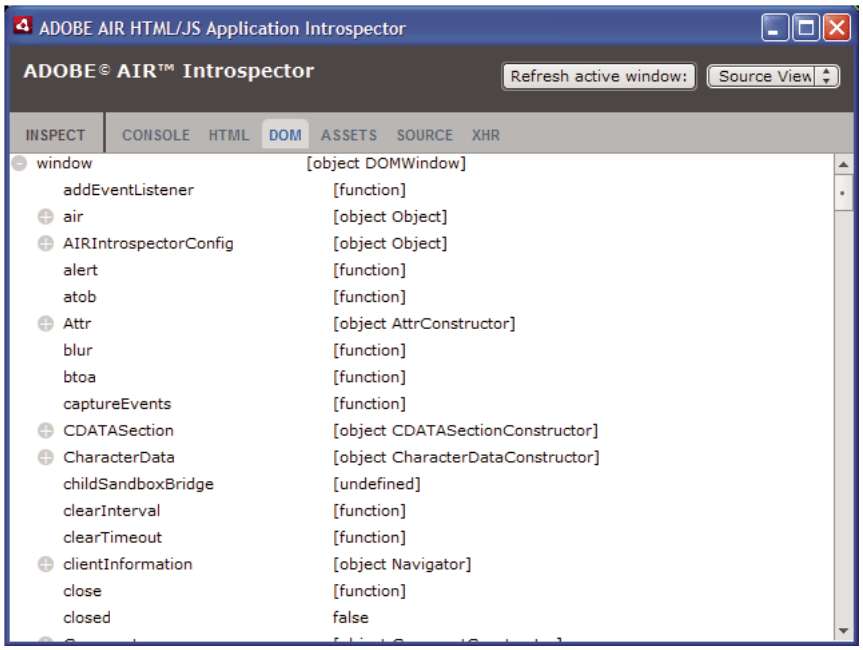
Clique no botão Inspecionar (à esquerda da lista de guias na janela do AIR Introspector). Você pode clicar em qualquer elemento na página HTML da janela principal e o objeto DOM associado é exibido na guia HTML. Quando a janela principal tem foco, você também pode pressionar o atalho no teclado para ativar e desativar o botão Inspecionar. O atalho no teclado é F11 por padrão. Você pode configurar o atalho no teclado como uma tecla que não seja a tecla F11; consulte "Configuração do AIR Introspector" na página 289.

Clique no botão Atualizar janela ativa (na parte superior da janela do AIR Introspector) para atualizar os dados exibidos na guia HTML.

Clique em CTRL+F no Windows ou em Command+F no Mac OS para pesquisar o texto correspondente exibido na guia. (Os nós de árvore que são visíveis não são pesquisados.)

A guia DOM

A guia DOM mostra o objeto da janela em uma estrutura de árvore. Você pode editar qualquer propriedade numérica e de string, e o valor editado se reflete no aplicativo.

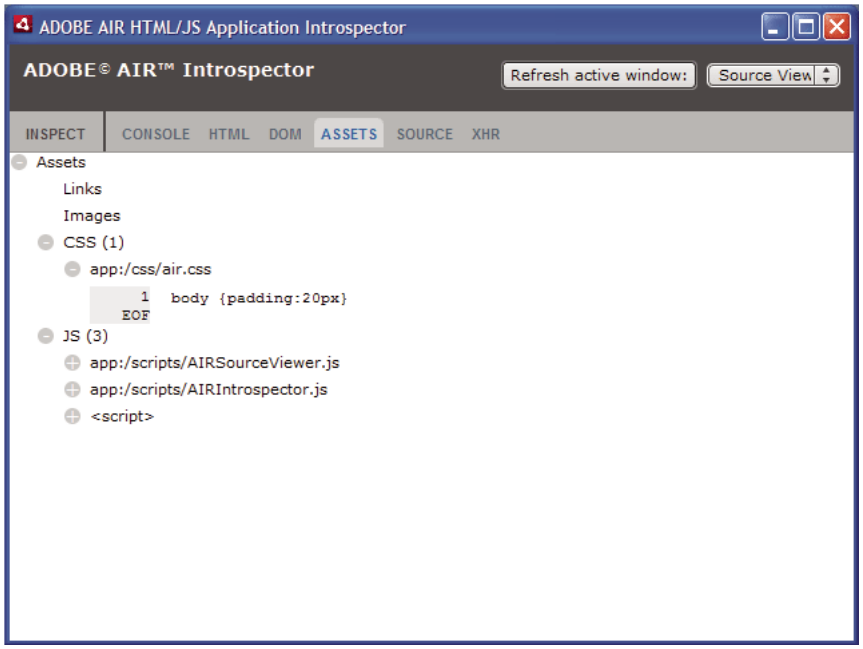


Clique no botão Atualizar janela ativa (na parte superior da janela do AIR Introspector) para atualizar os dados exibidos na guia DOM.

Clique em CTRL+F no Windows ou em Command+F no Mac OS para pesquisar o texto correspondente exibido na guia. (Os nós de árvore que são visíveis não são pesquisados.)

A guia Ativos

A guia Ativos permite verificar os links, imagens, CSS e arquivos JavaScript carregados na janela nativa. A expansão de um desses nós mostra o conteúdo do arquivo ou exibe a imagem real usada.



Clique no botão Atualizar janela ativa (na parte superior da janela do AIR Introspector) para atualizar os dados exibidos na guia Ativos.

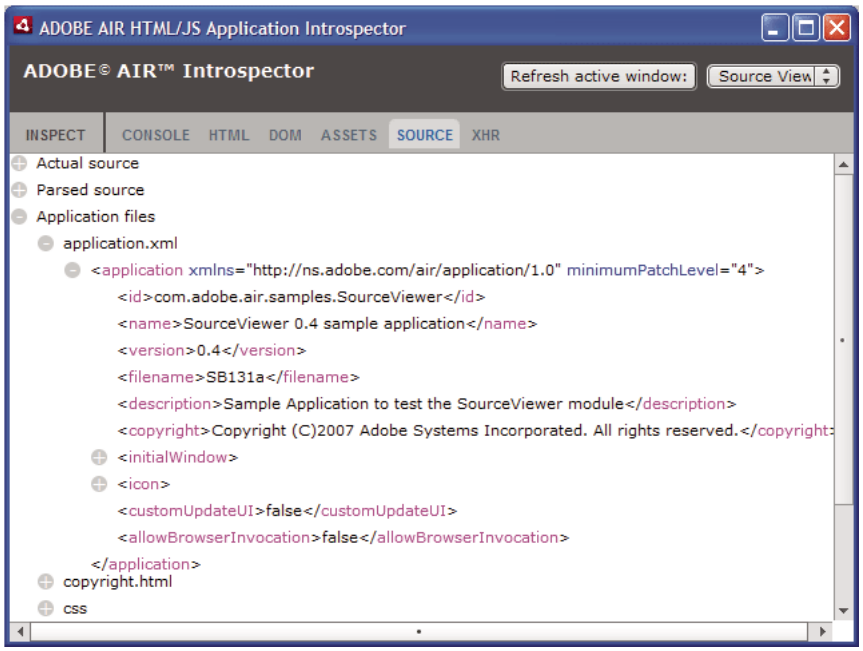
Clique em CTRL+F no Windows ou em Command+F no Mac OS para pesquisar o texto correspondente exibido na guia. (Os nós de árvore que são visíveis não são pesquisados.)

A guia Fonte

A guia Fonte inclui três seções:

- Fonte real — Mostra a fonte de HTML da página carregada como conteúdo de raiz quando o aplicativo for iniciado.
- Fonte analisada — Mostra a marcação real que compõe a IU do aplicativo, que pode ser diferente da fonte real, uma vez que o aplicativo gera o código de marcação durante o processo usando técnicas Ajax.

- Arquivos do aplicativo — Lista os arquivos no diretório do aplicativo. Essa listagem só está disponível para o AIR Introspector quando aberto a partir do conteúdo da caixa de segurança do aplicativo. Nessa seção, você pode ver o conteúdo dos arquivos de texto ou visualizar imagens.

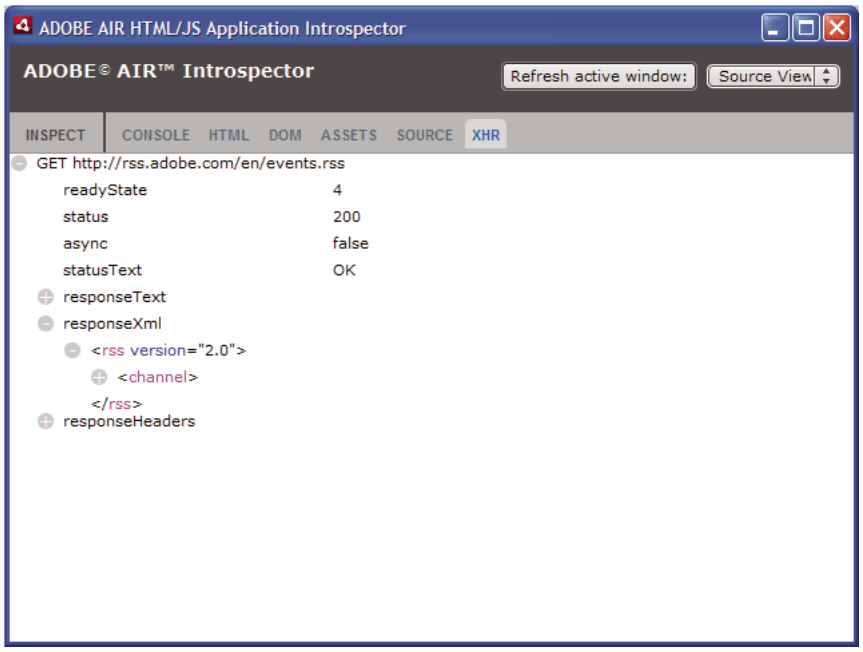


Clique no botão Atualizar janela ativa (na parte superior da janela do AIR Introspector) para atualizar os dados exibidos na guia Fonte.

Clique em CTRL+F no Windows ou em Command+F no Mac OS para pesquisar o texto correspondente exibido na guia. (Os nós de árvore que são visíveis não são pesquisados.)

A guia XHR

A guia XHR intercepta toda a comunicação XMLHttpRequest no aplicativo e registra as informações. Isso lhe permite ver as propriedades XMLHttpRequest, incluindo `responseText` e `responseXML` (quando disponíveis) em uma visualização de árvore.



Clique em CTRL+F no Windows ou em Command+F no Mac OS para pesquisar o texto correspondente exibido na guia. (Os nós de árvore que são visíveis não são pesquisados.)

Uso do AIR Introspector com conteúdo em uma caixa de proteção não do aplicativo

Você pode carregar o conteúdo do diretório do aplicativo em um iframe ou quadro que é mapeado em uma caixa de proteção externa ao aplicativo. (Consulte Segurança HTML no Adobe AIR para desenvolvedores em ActionScript ou Segurança HTML no Adobe AIR para desenvolvedores em HTML). Você pode usar o AIR Introspector com esse conteúdo, mas observe as seguintes regras:

- O arquivo AIRIntrospector.js deve ser incluído no conteúdo da caixa de proteção do aplicativo e também não do aplicativo (o iframe).
- Não sobrescreva a propriedade `parentSandboxBridge`; o código do AIR Introspector usa essa propriedade. Acrescente as propriedades conforme necessário. Por isso, em vez de escrever o seguinte:

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)}} ;
  ```

  Use a sintaxe da seguinte forma:

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- Do conteúdo da caixa de proteção não do aplicativo, você não pode abrir o AIR Introspector pressionando a tecla F12 ou chamando um dos métodos na classe air.Introspector.Console. Você pode abrir a janela do Introspector apenas clicando no botão Abrir Introspector. O botão é acrescentado por padrão no canto superior direito do iframe ou do quadro. (Devido a restrições de segurança impostas ao conteúdo da caixa de proteção não do aplicativo, pode-se abrir uma nova janela apenas como resultado de um gesto do usuário, como clicar em um botão.)
- Você pode abrir janelas separadas do AIR Introspector para a caixa de proteção do aplicativo e não do aplicativo. É possível diferenciar as duas usando o título exibido nas janelas do AIR Introspector.
- A guia Fonte não exibe os arquivos de aplicativo quando o AIR Introspector é executado de uma caixa de proteção não do aplicativo.
- O AIR Introspector só pode analisar o código na caixa de proteção da qual ele foi aberto.

Capítulo 20: Localização de aplicativos AIR

Adobe AIR 1.1 e posterior

O Adobe® AIR® inclui suporte para vários idiomas.

Para ter uma visão geral da localização de conteúdo no ActionScript 3.0 e na estrutura do Flex, consulte “Localização de aplicativos” no Guia do Desenvolvedor do ActionScript 3.0.

Idiomas suportados no AIR

O suporte para localização de aplicativos do AIR nos idiomas a seguir foi apresentado na versão AIR 1.1:

- Chinês simplificado
- Chinês tradicional
- Francês
- Alemão
- Italiano
- Japonês
- Coreano
- Português (Brasil)
- Russo
- Espanhol

Na versão AIR 1.5, foram acrescentados os seguintes idiomas:

- Tcheco
- Holandês
- Polonês
- Sueco
- Turco

Mais tópicos da Ajuda

Construção de aplicativos multilíngues do Flex no Adobe AIR

Construção de um aplicativo multilíngue baseado no HTML

Localização do nome e da descrição do aplicativo no instalador do aplicativo do AIR

Adobe AIR 1.1 e posterior

Você pode especificar vários idiomas para os elementos `name` e `description` no arquivo do descritor do aplicativo. Por exemplo, o seguinte especifica o nome do aplicativo em três idiomas (inglês, francês e alemão):

```
<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>
```

O atributo `xml:lang` de cada elemento de texto especifica um código de idioma, como definido por RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

O elemento name define o nome do aplicativo que o instalador de aplicativo do AIR exibe. O instalador de aplicativo do AIR usa o valor localizado que melhor corresponda aos idiomas da interface do usuário definidos pelas configurações do sistema operacional.

De modo semelhante, você pode especificar várias versões de idioma do elemento `description` no arquivo do descritor do aplicativo. Esse elemento define o texto de descrição que instalador de aplicativo do AIR exibe.

Essas configurações só se aplicam aos idiomas disponíveis no instalador de aplicativo do AIR. Elas não definem os códigos de idiomas disponíveis para a execução do aplicativo instalado. Os aplicativos do AIR podem oferecer interfaces do usuário que suportam vários idiomas, incluindo idiomas disponíveis ao instalador do aplicativo do AIR.

Para mais informações, consulte “Elementos descritores do aplicativo do AIR” na página 212.

Mais tópicos da Ajuda

Construção de aplicativos multilíngues do Flex no Adobe AIR

Construção de um aplicativo multilíngue baseado no HTML

Localização de conteúdo HTML com a estrutura de localização de HTML do AIR

Adobe AIR 1.1 e posterior

O AIR 1.1 SDK inclui uma estrutura de localização HTML. O arquivo de JavaScript AIRLocalizer.js define a estrutura. O diretório de estruturas do AIR SDK contém o arquivo AIRLocalizer.js. Esse arquivo inclui a classe Localizer, que oferece funcionalidade para auxiliar na criação de aplicativos que ofereçam suporte a várias versões localizadas.

Carregamento do código de estrutura de localização HTML do AIR

Para usar a estrutura de localização, copie o arquivo AIRLocalizer.js em seu projeto. Em seguida, inclua-a no arquivo HTML principal do aplicativo, usando uma tag de script:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

Em seguida, o JavaScript pode chamar o objeto `air.Localizer.localizer`:

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

O objeto `air.Localizer.localizer` é um objeto singleton que define métodos e propriedades para usar e gerenciar recursos de localização. A classe Localizer inclui os seguintes métodos:

Método	Descrição
`getFile()`	Obtém o texto de um grupo de recursos especificados para um código de idiomas especificado. Consulte "Obtenção de recursos para um código de idiomas específico" na página 305.
`getLocaleChain()`	Retorna os idiomas na cadeia de código de idiomas. Consulte "Definição da cadeia de código de idiomas" na página 304.
`getResourceBundle()`	Retorna as chaves do grupo e os valores correspondentes como um objeto. Consulte "Obtenção de recursos para um código de idiomas específico" na página 305.
`getString()`	Obtém a sequência definida para um recurso. Consulte "Obtenção de recursos para um código de idiomas específico" na página 305.
`setBundlesDirectory()`	Define o local do diretório de compactados. Consulte "Personalização de configurações HTML Localizer do AIR" na página 303
`setLocalAttributePrefix()`	Define o prefixo usado pelos atributos do localizer usados nos elementos HTML DOM. Consulte "Personalização de configurações HTML Localizer do AIR" na página 303
`setLocaleChain()`	Define a ordem de idiomas na cadeia de código de idiomas. Consulte "Definição da cadeia de código de idiomas" na página 304.
`sortLanguagesByPreference()`	Classifica os códigos de idiomas na cadeia de código de idiomas com base na ordem de códigos de idiomas nas configurações do sistema operacional. Consulte "Definição da cadeia de código de idiomas" na página 304.
`update()`	Atualiza o HTML DOM (ou um elemento DOM) com sequências localizadas da cadeia de código de idiomas atual. Para obter uma discussão sobre cadeias de códigos de idiomas, consulte "Gerenciamento de cadeias de códigos de idiomas" na página 301. Para obter mais informações sobre o método `update()`, consulte "Atualização de elementos DOM para uso de código de idiomas atual" na página 302.

A classe Localizer inclui as seguintes propriedades estáticas:

Propriedade	Descrição
`localizer`	Retorna uma referência para o objeto singleton Localizer do aplicativo.
`ultimateFallbackLocale`	Código de idiomas usado quando o aplicativo não oferece suporte a nenhuma preferência de usuário. Consulte "Definição da cadeia de código de idiomas" na página 304.

Especificação dos idiomas suportados

Use o elemento `<supportedLanguages>` no descritor do aplicativo para identificar os idiomas suportados pelo aplicativo. Esse elemento é usado apenas pelo iOS, runtime cativo do Mac e aplicativos Android e é ignorado por todos os outros tipos de aplicativo.

Se você não especificar esse elemento `<supportedLanguages>`, por padrão o empacotador realiza as seguintes ações com base no tipo de aplicativo:

- iOS — todos os idiomas suportados pelo runtime do AIR são listados na app store do iOS como idiomas suportados do aplicativo.
- Runtime cativo do Mac — Aplicativo empacotado com conjunto cativo não possui informações de localização.

- Android — O pacote do aplicativo tem recursos para todos os idiomas suportados pelo runtime do AIR.

Para obter mais informações, consulte “supportedLanguages” na página 242.

Definição de grupos de recursos

A estrutura de localização HTML faz a leitura de versões localizadas de sequências de arquivos de *localização*. O arquivo de localização é um conjunto de valores baseados em chaves, serializados em um arquivo de texto. O arquivo de localização é algumas vezes tratado como um *compactado*.

Crie um subdiretório do diretório de projeto do aplicativo, chamado código de idiomas. (Você também pode usar um nome diferente; consulte “Personalização de configurações HTML Localizer do AIR” na página 303.) Esse diretório incluirá os arquivos de localização. Esse diretório é conhecido como o *diretório de compactações*.

Para cada código de idiomas a que seu aplicativo oferece suporte, crie um subdiretório do diretório de compactações. Nomeie cada subdiretório para corresponder ao código de idiomas. Por exemplo, nomeie o diretório French como "fr" e o diretório English como "en". Você pode usar o caractere sublinhado (_) para definir o código de idiomas que tenha um código de país e idioma. Por exemplo, nomeie o diretório U.S. English como “en_us”. (Como alternativa, você pode usar um hífen, em vez de um sublinhado, como em “en-us.” A estrutura de localização HTML oferece suporte às duas opções).

Você pode adicionar qualquer número de arquivos de recursos a um subdiretório código de idiomas. Normalmente, você cria um arquivo de localização para cada idioma (e coloca o arquivo no diretório desse idioma). A estrutura de localização HTML inclui o método `getFile()` que permite ler o conteúdo de um arquivo (consulte “Obtenção de recursos para um código de idiomas específico” na página 305.

Arquivos com a extensão de arquivo .properties são conhecidos como arquivos de propriedades de localização. Você pode usá-los para definir pares de valores chave para um código de idiomas. O arquivo de propriedade define um valor de sequência em cada linha. Por exemplo, o seguinte define o valor de sequência `"Oi em inglês".` para uma chave de nome `greeting`:

```
greeting=Hello in English.
```

O arquivo de propriedades contendo o texto a seguir define seis pares de valores chave:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Este exemplo mostra uma versão em inglês do arquivo de propriedades que deve ser armazenado no diretório en.

A versão francesa desse arquivo de propriedades é colocada no diretório fr:

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Você pode definir vários arquivos de recursos para diferentes tipos de informações. Por exemplo, o arquivo legal.properties pode conter texto de padrão legal (como informações de direitos autorais). Você pode usar novamente esse recurso em diversos aplicativos. De modo semelhante, você pode definir arquivos separados que definem conteúdo localizado para diferentes partes da interface do usuário.

Use a codificação UTF-8 para esses arquivos, para oferecer suporte a vários idiomas.

Gerenciamento de cadeias de códigos de idiomas

Quando o aplicativo carrega o arquivo AIRLocalizer.js, ele examina os códigos de idiomas definidos no aplicativo. Esses códigos de idiomas correspondem aos subdiretórios do diretório de compactações (consulte "Definição de grupos de recursos" na página 300). Essa lista de códigos de idiomas disponíveis é conhecida como a *cadeia de código de idiomas*. O arquivo AIRLocalizer.js classifica automaticamente a cadeia de código de idiomas com base na ordem de preferência definida pelas configurações do sistema operacional. (A propriedade `Capabilities.languages` lista os idiomas da interface do usuário do sistema operacional, por ordem de preferência).

Portanto, se um aplicativo define recursos para os códigos de idiomas "en", "en_US" e "en_UK", a estrutura HTML Localizer do AIR classifica a cadeia de código de idiomas de maneira apropriada. Quando o aplicativo é iniciado em um sistema que informa "en" como código de idiomas principal, a cadeia de código de idiomas é classificada como `["en", "en_US", "en_UK"]`. Nesse caso o aplicativo procura primeiramente por recursos no grupo "en" e, em seguida, no grupo "en_US".

No entanto, se o sistema informar "en-US" como código de idiomas principal, a classificação usará `["en_US", "en", en_UK"]`. Nesse caso, o aplicativo procura primeiramente recursos no grupo "en_US" e, em seguida, no grupo "en".

Por padrão, o aplicativo define o primeiro código de idiomas na cadeia de código de idiomas como o código de idiomas padrão para usar. Você pode solicitar que o usuário selecione um código de idiomas na primeira execução do aplicativo. Em seguida, você pode optar por armazenar a seleção em um arquivo de preferências e usar esse código de idiomas na inicialização subsequente do aplicativo.

O aplicativo pode usar as sequências de recursos em qualquer código de idiomas da cadeia de código de idiomas. Se um código de idiomas específico não definir uma sequência de recursos, o aplicativo usará a próxima sequência de recursos correspondente em outros códigos de idiomas definidos na cadeia de código de idiomas.

Você pode personalizar a cadeia de código de idiomas chamando o método `setLocaleChain()` do objeto Localizer. Consulte "Definição da cadeia de código de idiomas" na página 304.

Atualização de elementos DOM com conteúdo localizado

O elemento no aplicativo pode fazer referência a um valor chave em um arquivo de propriedades de localização. Por exemplo, o elemento `title` no exemplo a seguir especifica um atributo `local_innerHTML`. A estrutura de localização usa esse atributo para pesquisar um valor localizado. Por padrão, a estrutura pesquisa nomes de atributos que iniciam com `"local_"`. A estrutura atualiza os atributos com nomes que correspondem ao texto em seguida a `"local_"`. Nesse caso, a estrutura define o atributo `innerHTML` do elemento `title`. O atributo `innerHTML` usa o valor definido para a chave `mainWindowTitle` no arquivo de propriedades padrão (default.properties):

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Se o código de idiomas atual não definir nenhum valor correspondente, a estrutura do localizador procura o restante da cadeia de códigos de idiomas. Ele usa o próximo código de idiomas na cadeia de código de idiomas para o qual um valor está definido.

No exemplo a seguir, o texto (atributo `innerHTML`) do elemento `p` usa o valor da chave `greeting` definido no arquivo de propriedades padrão:

```
<p local_innerHTML="default.greeting" />
```

No exemplo a seguir, o atributo de valor (e texto exibido) do elemento `input` usa o valor da chave `btnBlue` definido no arquivo de propriedades padrão:

```
<input type="button" local_value="default.btnBlue" />
```

Para atualizar o HTML DOM para usar as sequências definidas na cadeia de código de idiomas atual, chame o método `update()` do objeto Localizer. Chamar o método `update()` faz com que o objeto Localizer analise o DOM e aplique manipulações onde encontrar atributos de localização (`"local_..."`):

```
air.Localizer.localizer.update();
```

Você pode definir valores tanto para um atributo (como "innerHTML") quanto para o respectivo atributo de localização correspondente (como "local_innerHTML"). Nesse caso, a estrutura de localização só sobrescreve o valor de atributo se ela encontrar um valor correspondente na cadeia de localização. Por exemplo, o elemento a seguir define os atributos `value` e `local_value`:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

Você também pode atualizar apenas um elemento DOM específico Consulte a próxima seção, "Atualização de elementos DOM para uso de código de idiomas atual" na página 302.

Por padrão, o HTML Localizer do AIR usa `"local_"` como o prefixo de atributos que definem as configurações de localização do elemento. Por exemplo, por padrão, o atributo `local_innerHTML` define o nome do grupo e recurso usados no valor `innerHTML` do elemento. Além disso, por padrão, o atributo `local_value` define o nome do grupo e recurso usados no atributo `value` do elemento. Você pode configurar o Localizer para usar um prefixo de atributo além de `"local_"`. Consulte "Personalização de configurações HTML Localizer do AIR" na página 303

Atualização de elementos DOM para uso de código de idiomas atual

Quando o objeto Localizer atualiza o HTML DOM, isso faz com que elementos marcados usem valores de atributos com base nas sequências definidas na cadeia de código de idiomas atual. Para fazer com que o localizador HTML atualize o HTML DOM, chame o método `update()` do objeto `Localizer`:

```
air.Localizer.localizer.update();
```

Para atualizar apenas um elemento DOM especificado, passe-o como parâmetro para o método `update()`. O método `update()` só tem um parâmetro, `parentNode`, que é opcional. Quando especificado, o parâmetro `parentNode` define o elemento DOM para ser localizado. Chamar o método `update()` e especificar o parâmetro `parentNode` define os valores localizados de todos os elementos filhos que especificam atributos de localização.

Por exemplo, considere o seguinte 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 atualizar esse elemento para usar as sequências localizadas definidas na cadeia de código de idiomas atual, use o seguinte código JavaScript:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Se o valor chave não for encontrado na cadeia de código de idiomas, a estrutura de localização definirá o valor de atributo como o valor do atributo `"local_"`. Por exemplo, no exemplo anterior, suponhamos que a estrutura de localização não possa encontrar o valor da chave `lblColors` (em qualquer um dos arquivos default.properties na cadeia de código de idiomas). Nesse caso, ela usará `"default.lblColors"` como valor `innerHTML`. Usar esse valor indica (para o desenvolvedor) ausência de recursos.

O método `update()` despacha o evento `resourceNotFound` quando não consegue encontrar um recurso na cadeia de código de idiomas. A constante `air.Localizer.RESOURCE_NOT_FOUND` define a sequência `"resourceNotFound"`. O evento tem três propriedades: `bundleName`, `resourceName` e `locale`. A propriedade `bundleName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `resourceName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `locale` é o nome do código de idiomas em que o recurso não foi encontrado.

O método `update()` despacha o evento `bundleNotFound` quando não consegue encontrar o grupo especificado. A constante `air.Localizer.BUNDLE_NOT_FOUND` define a sequência `"bundleNotFound"`. O evento tem duas propriedades: `bundleName` e `locale`. A propriedade `bundleName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `locale` é o nome do código de idiomas em que o recurso não foi encontrado.

O método `update()` funciona de forma assíncrona (e despacha os eventos `resourceNotFound` e `bundleNotFound` de forma assíncrona). O código a seguir define ouvintes de evento dos eventos `resourceNotFound` e `bundleNotFound`:

```
air.Localizer.localizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.update();
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

Personalização de configurações HTML Localizer do AIR

O método `setBundlesDirectory()` do objeto Localizer permite personalizar o caminho do diretório de compactações. O método `setLocalAttributePrefix()` do objeto Localizer permite personalizar o caminho do diretório de compactações e personalizar o valor de atributo usado pelo Localizer.

O diretório de compactações padrão é definido como o subdiretório código de idiomas do diretório do aplicativo. Você pode especificar outro diretório chamando o método `setBundlesDirectory()` do objeto Localizer. Esse método usa o parâmetro `path`, que é o caminho para o diretório de compactações desejado, como uma sequência. O valor do parâmetro `path` pode ser algum dos seguintes:

- Uma sequência que define o caminho relativo para o diretório do aplicativo, como `"locales"`
- Uma sequência que define a URL válida que usa os esquemas de URL `app`, `app-storage` ou `file`, como `"app://languages"` (*não* usa o esquema de URL `http`)
- O objeto File

Para informações sobre URLs e caminhos de diretórios, consulte:

- Caminhos dos objetos File (para desenvolvedores em ActionScript)
- Paths of File objects (para desenvolvedores em HTML)

Por exemplo, o código a seguir define o diretório de compactações como subdiretório de idiomas do diretório de armazenamento do aplicativo (não o diretório de aplicativo):

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

Passe um caminho válido como o parâmetro `path`. Do contrário, o método emitirá uma exceção BundlePathNotFoundError. Esse erro tem `"BundlePathNotFoundError"` como sua propriedade `name` e a respectiva propriedade `message` especifica o caminho inválido.

Por padrão, o HTML Localizer do AIR usa `"local_"` como o prefixo de atributos que definem as configurações de localização do elemento. Por exemplo, o atributo `local_innerHTML` define o nome do grupo e recurso usado no valor `innerHTML` do seguinte elemento `input`:

```
<p local_innerHTML="default.greeting" />
```

O método `setLocalAttributePrefix()` do objeto Localizer permite usar outro prefixo de atributo que não seja `"local_"`. Esse método estático usa um parâmetro, que é a sequência que você deseja usar como prefixo de atributo. Por exemplo, o código a seguir define que a estrutura de localização use "loc_" como o prefixo de atributo:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Você pode personalizar o prefixo de atributo que a estrutura de localização utiliza. Talvez você deseje personalizar o prefixo, caso o valor padrão (`"local_"`) entre em conflito com o nome de outro atributo usado por seu código. Certifique-se de usar caracteres válidos em atributos HTML quando chamar esse método. (Por exemplo, o valor não pode conter um caractere de espaço em branco).

Para obter mais informações sobre como usar os atributos de localização em elementos HTML, consulte "Atualização de elementos DOM com conteúdo localizado" na página 301.

As configurações do diretório de compactações e prefixo de atributo não persistem entre sessões de aplicativos diferentes. Se você usar uma configuração personalizada de diretório de compactações ou de prefixo de atributo, certifique-se de defini-las sempre que iniciar o aplicativo.

Definição da cadeia de código de idiomas

Por padrão, quando você carrega o código AIRLocalizer.js, ele define a cadeia de código de idiomas padrão. Os códigos de idiomas disponíveis nas configurações de diretório de compactações e de idioma do sistema operacional definem essa cadeia de código de idiomas. (Para obter detalhes, consulte "Gerenciamento de cadeias de códigos de idiomas" na página 301.)

Você pode modificar a cadeia de código de idiomas chamando o método estático `setLocaleChain()` do objeto Localizer. Por exemplo, talvez você deseje chamar esse método se o usuário indicar a preferência por um idioma específico. O método `setLocaleChain()` usa um parâmetro, `chain`, que é uma matriz de códigos de idiomas, como `["fr_FR","fr","fr_CA"]`. A ordem dos códigos de idiomas na matriz define a ordem em que a estrutura pesquisa recursos (em operações subsequentes). Se o recurso não for encontrado para o primeiro código de idiomas na cadeia, ele continua pesquisando em outros recursos de códigos de idiomas. Se o argumento `chain` estiver ausente, se não for uma matriz ou for uma matriz vazia, a função falhará e emitirá uma exceção IllegalArgumentsError.

O método estático `getLocaleChain()` do objeto Localizer retorna uma matriz que lista os códigos de idiomas na cadeia de código de idiomas atual.

O código a seguir faz a leitura da cadeia de código de idiomas atual e adiciona dois códigos de idiomas franceses ao cabeçalho da cadeia:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

O método `setLocaleChain()` despacha o evento `"change"` quando atualiza a cadeia de código de idiomas. A constante `air.Localizer.LOCALE_CHANGE` define a sequência `"change"`. O evento tem uma propriedade, `localeChain`, uma matriz de códigos de idiomas na nova cadeia de código de idiomas. O código a seguir define um ouvinte de evento para esse 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);
}
```

A propriedade estática `air.Localizer.ultimateFallbackLocale` representa o código de idiomas usado quando o aplicativo não oferece suporte a nenhuma preferência do usuário. O valor padrão é `"en"`. Você pode defini-lo como outro código de idiomas, como mostrado no código a seguir:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Obtenção de recursos para um código de idiomas específico

O método `getString()` do objeto Localizer retorna a sequência definida para um recurso em um código de idiomas específico. Não é necessário especificar o valor `locale` ao chamar o método. Nesse caso, o método procura em toda a cadeia de código de idiomas e retorna a sequência no primeiro código de idiomas que fornece o nome de recurso determinado. O método tem os seguintes parâmetros:

Parâmetro	Descrição
`bundleName`	O grupo que contém o recurso. Esse é o nome de arquivo do arquivo de propriedades sem a extensão .properties. (Por exemplo, se esse parâmetro for definido como `"alerts"`, o código Localizer pesquisará nos arquivos de localização chamados alerts.properties.
`resourceName`	O nome do recurso.
`templateArgs`	Opcional. A matriz de sequência para substituir as tags numeradas na sequência de substituição. Por exemplo, considere uma chamada para a função em que o parâmetro `templateArgs` seja `["Raúl", "4"]` e a sequência de recursos correspondente seja `"Hello, {0}. Você tem {1} mensagens novas."`. Nesse caso, a função retorna `"Hello, Raúl. Você tem 4 mensagens novas"`. . Para ignorar essa configuração, passe o valor `null`.
`locale`	Opcional. O código de idiomas (como `"en"`, `"en_us"` ou `"fr"`) que deve ser usado. Se o código de idiomas for fornecido e nenhum valor correspondente for encontrado, o método não continuará a busca por valores em outros códigos de idiomas na cadeia de código de idiomas. Se nenhum código de idiomas for especificado, a função retornará a sequência no primeiro código de idiomas na cadeia de código de idiomas que ofereça o valor para o nome de recurso determinado.

A estrutura de localização pode atualizar atributos HTML DOM marcados. No entanto, você pode usar sequências localizadas de outras maneiras. Por exemplo, você pode usar uma sequência em HTMLs gerados dinamicamente ou como valor de parâmetro em uma chamada de função. Por exemplo, o código a seguir chama a função `alert()` com a sequência definida no recurso `error114` no arquivo de propriedades padrão do código de idiomas fr_FR:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

O método `getString()` despacha o evento `resourceNotFound` quando não consegue encontrar o recurso no grupo especificado. A constante `air.Localizer.RESOURCE_NOT_FOUND` define a sequência `"resourceNotFound"`. O evento tem três propriedades: `bundleName`, `resourceName` e `locale`. A propriedade `bundleName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `resourceName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `locale` é o nome do código de idiomas em que o recurso não foi encontrado.

O método `getString()` despacha o evento `bundleNotFound` quando não consegue encontrar o grupo especificado. A constante `air.Localizer.BUNDLE_NOT_FOUND` define a sequência `"bundleNotFound"`. O evento tem duas propriedades: `bundleName` e `locale`. A propriedade `bundleName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `locale` é o nome do código de idiomas em que o recurso não foi encontrado.

O método `getString()` opera de forma assíncrona (e despacha os eventos `resourceNotFound` e `bundleNotFound` de forma assíncrona). O código a seguir define ouvintes de evento dos eventos `resourceNotFound` e `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);
}
```

O método `getResourceBundle()` do objeto Localizador retorna um grupo especificado para um determinado local. O valor de retorno do método é um objeto com propriedades correspondentes às chaves do grupo. (Se o aplicativo não puder localizar o grupo especificado, o método retornará `null`.)

O método assume dois parâmetros - `locale` e `bundleName`.

Parâmetro	Descrição
`locale`	O local (como `"fr"`).
`bundleName`	O nome do grupo.

Por exemplo, o código a seguir chama o método `document.write()` para carregar o grupo padrão para o local fr. Em seguida, ele chama o método `document.write()` com valores das chaves `str1` e `str2` do grupo:

```
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/>");
```

O método `getResourceBundle()` despacha o evento `bundleNotFound` quando não consegue encontrar o grupo especificado. A constante `air.Localizer.BUNDLE_NOT_FOUND` define a sequência `"bundleNotFound"`. O evento tem duas propriedades: `bundleName` e `locale`. A propriedade `bundleName` é o nome do grupo em que o recurso não foi encontrado. A propriedade `locale` é o nome do código de idiomas em que o recurso não foi encontrado.

O método `getFile()` do objeto Localizer retorna o conteúdo de um grupo, como uma sequência, para um código de idiomas determinado. O arquivo de compactação é interpretado como arquivo UTF-8. O método inclui os seguintes parâmetros:

Parâmetro	Descrição
`resourceFileName`	Nome de arquivo do arquivo de recurso (como `"about.html"`).

`templateArgs`	Opcional. A matriz de sequência para substituir as tags numeradas na sequência de substituição. Por exemplo, considere uma chamada para a função em que o parâmetro `templateArgs` seja `["Raúl", "4"]` e o arquivo de recursos correspondente contenha duas linhas:
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	Nesse caso, a função retorna uma sequência com duas linhas:
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

`locale`	O código de idiomas, como `"en_GB"`, que deve ser usado. Se o código de idiomas for fornecido e nenhum arquivo correspondente for encontrado, o método não continuará a busca em outros códigos de idiomas na cadeia de código de idiomas. Se o código de idiomas *no* for especificado, a função retornará o texto no primeiro código de idiomas na cadeia de código de idiomas com um arquivo que corresponda a `resourceFileName`.

Por exemplo, o código a seguir chama o método `document.write()` usando o conteúdo do arquivo about.html do código de idiomas fr:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

O método `getFile()` despacha o evento `fileNotFound` quando não consegue encontrar um recurso na cadeia de código de idiomas. A constante `air.Localizer.FILE_NOT_FOUND` define a sequência `"resourceNotFound"`. O método `getFile()` funciona de forma assíncrona (e despacha o evento `fileNotFound` de forma assíncrona). O evento tem duas propriedades: `fileName` e `locale`. A propriedade `fileName` é o nome do arquivo não encontrado. A propriedade `locale` é o nome do código de idiomas em que o recurso não foi encontrado. O código a seguir define um ouvinte de evento para esse 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);
}
```

Mais tópicos da Ajuda

Construção de um aplicativo multilíngue baseado no HTML

Capítulo 21: Variáveis de ambiente do caminho

O AIR SDK contém alguns programas que podem ser ativados a partir de uma linha de comando ou terminal. A execução destes programas pode muitas vezes ser mais conveniente quando o caminho para o diretório bin do SDK está incluído na variável de ambiente do caminho.

A informação aqui apresentada descreve como definir o caminho no Windows, Mac e Linux e deve servir como guia prático. No entanto, as configurações de computadores variam muito, de modo que o procedimento pode não funcionar para todos os sistemas. Nestes casos, você pode encontrar as informações necessárias na documentação do seu sistema operacional ou na Internet.

Configuração do PATH no Linux e Mac OS usando o shell Bash

Ao digitar um comando em uma janela de terminal (o shell, um programa que lê o que você digitou e tenta responder de forma adequada), deve primeiro localizar o programa de comando no seu sistema de arquivos. O shell procura por comandos em uma lista de diretórios armazenada em uma variável de ambiente chamada $PATH. Para ver o que está listado no caminho, digite:

```
echo $PATH
```

Isso retorna uma lista separada por dois pontos de diretórios que deve ser algo parecido com isto:

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

O objetivo é acrescentar o caminho para o diretório bin AIR SDK à lista para que o shell possa encontrar as ferramentas ADT e ADL. Supondo que você tenha colocado o AIR SDK em `/Users/fred/SDKs/AIR`, o seguinte comando adicionará os diretórios necessários para o caminho:

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Nota: *Se o caminho contiver caracteres de espaço em branco, faça o escape com uma barra invertida, como a seguir:*

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

Você pode usar o comando `echo` novamente para ter certeza de que funcionou:

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDK
s/android/tools
```

Até aqui tudo bem. Agora você deve ser capaz de escrever os seguintes comandos e obter uma resposta encorajadora:

```
adt -version
```

Se você modificou a variável $PATH corretamente, o comando deve informar a versão do ADT.

Há ainda um problema, no entanto; na próxima vez que você abrir uma nova janela de terminal, vai notar que as novas entradas no caminho não estão mais lá. O comando para definir o caminho deve ser executado sempre que iniciar um novo terminal.

Uma solução comum para este problema é adicionar o comando de um dos scripts de inicialização usado pelo shell. No Mac OS, você pode criar o arquivo .bash_profile, no diretório ~/username e este será executado cada vez que uma nova janela de terminal for aberta. No Ubuntu, o script de inicialização será executado quando uma nova janela .bashrc for aberta. Outras distribuições Linux e programas de shell têm convenções parecidas.

Para adicionar o comando para o script de inicialização do shell:

1. Mude para o diretório home:

   ```
   cd
   ```

2. Crie o perfil de configuração do shell (se necessário) e redirecione o texto que você digitar para o final do arquivo com "`cat >>`". Use o arquivo adequado para seu sistema operacional e shell. Você pode usar `.bash_profile` no Mac OS e `.bashrc` no Ubuntu, por exemplo.

   ```
   cat >> .bash_profile
   ```

3. Digite o texto para adicionar o arquivo:

   ```
   export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
   ```

4. Termine o redirecionamento texto pressionando `CTRL-SHIFT-D` no teclado.

5. Exiba o arquivo para se certificar que tudo está bem:

   ```
   cat .bash_profile
   ```

6. Abra uma nova janela de terminal para verificar o caminho:

   ```
   echo $PATH
   ```

   As inclusões de caminhos devem ser listadas.

Se posteriormente criar uma nova versão de um dos SDKs em diretório diferente, não se esqueça de atualizar o comando do caminho do arquivo de configuração. Caso contrário, o shell continuará a utilizar a versão antiga.

Configuração do caminho no Windows

Quando você abre uma janela de comando no Windows, a janela herda as variáveis de ambiente global definidas nas propriedades do sistema. Uma das variáveis importantes é o caminho, que é a lista de diretórios que o programa de comando buscas quando você digita o nome de um programa para ser executado. Para ver o que está incluído no caminho quando você estiver usando uma janela de comando, digite:

```
set path
```

Isto mostrará uma lista separada por vírgulas de diretórios que se parece com isso:

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

O objetivo é acrescentar o caminho para o diretório bin AIR SDK à lista de maneira que o programa de comando possa encontrar as ferramentas ADT e ADL. Supondo que tenha colocado o AIR SDK em `C:\SDKs\AIR`, você pode adicionar a entrada do caminho correto com o seguinte procedimento:

1. Abra a caixa de diálogo Propriedades do Sistema no Painel de Controle ou clique no ícone Meu Computador e escolha Propriedades no menu.
2. Na aba Avançado, clique no botão Variáveis de Ambiente.
3. Selecione a Entrada de caminho na seção de Variáveis do sistema da caixa de diálogo Variáveis de Ambiente
4. Clique em Editar.

5 Vá até o final do texto no campo de valor Variável.

6 Digite o seguinte texto no final do valor atual:

```
;C:\SDKs\AIR\bin
```

7 Clique em OK em todas as caixas de diálogo para salvar o caminho.

Se você tiver alguma janela de comando aberta, observe que os ambientes não serão atualizados. Abra uma nova janela de comando e digite o seguinte comando para verificar se os caminhos estão configurados corretamente:

```
adt -version
```

Se você alterar o local do AIR SDK, ou adicionar uma nova versão, lembre-se de atualizar a variável do caminho.