Création d’applications ADOBE® AIR®

Informations juridiques

Vous trouverez des informations juridiques à l'adresse http://help.adobe.com/fr_FR/legalnotices/index.html.

Dernière mise à jour le 31/5/2017

Sommaire

Chapitre 1 : A propos d'Adobe AIR

Adobe® AIR® est un moteur d'exécution gérant plusieurs systèmes d'exploitation et écrans, qui permet d'exploiter vos compétences en matière de développement Web pour développer et déployer des applications Internet enrichies (RIA) destinées aux ordinateurs de bureau et périphériques mobiles. Vous pouvez créer des applications AIR de bureau, pour télévisions et périphériques mobiles en ActionScript 3.0 à l'aide d'Adobe® Flex et d'Adobe® Flash® (type SWF). Vous pouvez également créer des applications AIR de bureau par le biais de HTML, JavaScript® et Ajax (type HTML).

Pour plus d'informations sur la prise en main et l'utilisation d'Adobe AIR, voir le site Adobe AIR Developer Connection (http://www.adobe.com/devnet/air/).

AIR permet de travailler dans des environnements qui vous sont familiers, ainsi que d'exploiter les outils et approches qui vous conviennent. La prise en charge de Flex, HTML, JavaScript et Ajax assure la création d'une solution optimale qui répond à vos besoins.

Vous pouvez par exemple développer des applications qui font appel à l'une des technologies suivantes ou à une combinaison de celles-ci :

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

Les utilisateurs manipulent les applications AIR comme toute application native. Le moteur d'exécution est installé une seule fois sur l'ordinateur ou le périphérique de l'utilisateur. Il est alors possible d'installer et d'exécuter les applications AIR à l'instar de toute autre application de bureau. (Sous iOS, étant donné que chaque application AIR est autonome, aucun moteur d'exécution d'AIR n'est installé.)

Quel que soit le système d'exploitation utilisé, le moteur d'exécution propose une plate-forme et une structure uniformes de déploiement d'applications. La cohérence des fonctionnalités et interactions rend ainsi superflus les tests dans plusieurs navigateurs. Au lieu de développer une application pour un système d'exploitation déterminé, vous ciblez le moteur d'exécution. Cette approche offre les avantages suivants :

- Les applications développées pour AIR s'exécutent sur divers systèmes d'exploitation sans nécessiter d'intervention supplémentaire de votre part. Le moteur d'exécution assure une présentation et des interactions cohérentes et prévisibles sur tous les systèmes d'exploitation pris en charge par AIR.
- Grâce à l'exploitation de technologies Web et de modèles de conception existants, la création d'applications est plus rapide. Vous pouvez assurer la migration des applications Web vers le bureau sans avoir à apprendre les technologies traditionnelles de développement pour le bureau ou un code natif complexe.
- Parce qu'il ne fait pas appel à des langages de niveau inférieur tels que C et C++, le développement d'applications s'en trouve simplifié. Il est ainsi inutile de gérer les API complexes de bas niveau propres à chaque système d'exploitation.

Développer des applications pour AIR vous permet d'exploiter un riche ensemble de structures et d'API :

- API propres à AIR proposées par le moteur d'exécution et la structure AIR
- API d'ActionScript utilisées par les fichiers SWF et la structure Flex (ainsi que qu'autres bibliothèques et structures basées sur ActionScript)
- HTML, CSS et JavaScript
- La plupart des structures Ajax

- Les extensions natives pour Adobe AIR fournissent des API ActionScript permettant d'accéder à des fonctionnalités propres à la plate-forme programmées en code natif. Les extensions natives peuvent également permettre d'accéder au code natif hérité, ainsi qu'au code natif qui fournit de meilleures performances.

AIR a un impact considérable sur la création, le déploiement et l'utilisation des applications. Il vous assure un contrôle créatif accru et permet la migration des applications Flash, Flex, HTML et Ajax vers le bureau, les périphériques mobiles et les télévisions.

Pour plus d'informations sur le contenu de chaque nouvelle mise à jour d'AIR, voir les notes de mise à jour sur Adobe AIR (http://www.adobe.com/go/learn_air_relnotes_fr).

Chapitre 2 : Installation d'Adobe AIR

Le moteur d'exécution d'Adobe® AIR® permet d'exécuter des applications AIR. Pour installer le moteur d'exécution, procédez comme suit, au choix :

- Installez le moteur d'exécution séparément (sans installer d'application AIR).
- Installez une application AIR pour la première fois par le biais d'un « badge » d'installation de page Web (vous êtes également invité à installer le moteur d'exécution).
- Créez un programme d'installation personnalisé qui installe votre application et le moteur d'exécution. Vous devez demander l'autorisation de distribuer le moteur d'exécution AIR à Adobe. Pour demander l'autorisation, consulter la page Distribution du moteur d'exécution Adobe AIR. Notez qu'Adobe ne fournit aucun outil pour créer un tel programme d'installation. De nombreux kits d'outils tiers sont néanmoins disponibles.
- Installez une application AIR qui intègre AIR en tant que moteur d'exécution captif. Un moteur d'exécution natif est utilisé uniquement par l'application de mise en paquet. Il n'est pas utilisé pour exécuter d'autres applications AIR. Le regroupement du moteur d'exécution est possible sur Mac et Windows. Sur iOS, toutes les applications incluent un moteur d'exécution intégré. A partir d'AIR 3.7, les applications Android incluent un moteur d'exécution intégré par défaut (même si vous pouvez utiliser un moteur d'exécution distinct).
- Configurez un environnement de développement AIR tel que le kit SDK AIR, Adobe® Flash® Builder™ ou le kit SDK Adobe Flex® (qui contient les outils de développement de ligne de commande AIR). Le moteur d'exécution intégré au kit SDK est réservé au débogage d'applications, et non à l'exécution d'applications AIR installées.

Vous trouverez la configuration requise pour l'installation d'AIR et l'exécution d'applications AIR dans le document : Adobe AIR : Configuration requise (http://www.adobe.com/fr/products/air/systemreqs/).

Le programme d'installation du moteur d'exécution et celui de l'application AIR créent des fichiers journaux lors de l'installation, la mise à jour ou la suppression de l'application ou du moteur d'exécution. Vous pouvez consulter ces journaux pour déterminer la cause de tout problème d'installation. Voir Installation logs.

Installation d'Adobe AIR

Pour installer ou mettre à jour le moteur d'exécution, l'utilisateur doit disposer de privilèges d'administration sur l'ordinateur.

Installation du moteur d'exécution sur un ordinateur Windows

1. Téléchargez le fichier d'installation du moteur d'exécution à partir de http://get.adobe.com/fr/air.
2. Double-cliquez sur le fichier d'installation du moteur d'exécution.
3. Dans la fenêtre d'installation, suivez les invites pour achever l'installation.

Installation du moteur d'exécution sur un ordinateur Mac

1. Téléchargez le fichier d'installation du moteur d'exécution à partir de http://get.adobe.com/fr/air.
2. Double-cliquez sur le fichier d'installation du moteur d'exécution.
3. Dans la fenêtre d'installation, suivez les invites pour achever l'installation.
4. Si le programme d'installation affiche une fenêtre d'authentification, entrez votre nom d'utilisateur et votre mot de passe Mac OS.

Installation du moteur d'exécution sur un ordinateur Linux

Remarque : *actuellement, AIR 2.7 et les versions ultérieures ne sont pas prises en charge sur Linux. Les applications AIR déployées sur Linux doivent continuer d'utiliser le kit SDK d'AIR 2.6.*

Utilisation du programme d'installation binaire :

1. Accédez au fichier d'installation binaire à l'adresse http://kb2.adobe.com/cps/853/cpsid_85304.html et téléchargez-le.
2. Définissez les autorisations du fichier de sorte que le programme d'installation puisse être exécuté : Vous pouvez définir les autorisations sur une ligne de commande, comme suit :

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

   Certaines versions de Linux vous permettent de définir les autorisations des fichiers dans la boîte de dialogue Propriétés, ouverte par l'intermédiaire d'un menu contextuel.
3. Exécutez le programme d'installation à partir de la ligne de commande ou en double-cliquant sur le fichier d'installation du moteur d'exécution.
4. Dans la fenêtre d'installation, suivez les invites pour achever l'installation.

Adobe AIR est installé sous forme de package natif. En d'autres termes, en tant que rpm sur une distribution de type rpm et en tant que deb sur une distribution Debian. AIR ne prend actuellement en charge aucun autre format de package.

Utilisation des programmes d'installation de package :

1. Accédez au fichier du package AIR à l'adresse http://kb2.adobe.com/cps/853/cpsid_85304.html. Téléchargez le package rpm ou Debian, selon le format pris en charge par votre système.
2. Si nécessaire, double-cliquez sur le fichier de package d'AIR pour installer le package.

   Vous pouvez également effectuer l'installation à partir la ligne de commande :

   a. Sur un système Debian :

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

   b. Sur un système rpm :

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

      Ou si vous mettez à jour une version existante (AIR 1.5.3 ou ultérieur) :

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

Pour installer AIR 2 et les applications AIR, vous devez disposer de privilèges de niveau administrateur sur l'ordinateur.

Adobe AIR est installé à l'emplacement suivant : /opt/Adobe AIR/Versions/1.0

AIR enregistre le type mime « application/vnd.adobe.air-application-installer-package+zip », ce qui signifie que les fichiers .air correspondent à ce type mime et sont donc associés au moteur d'exécution d'AIR.

Installation du moteur d'exécution sur un périphérique Android

Vous pouvez installer la version la plus récente du moteur d'exécution d'AIR à partir d'Android Market.

Vous pouvez installer des versions de développement du moteur d'exécution d'AIR à partir d'un lien de page Web ou à l'aide de la commande ADT `-installRuntime`. Il est possible d'installer une seule version du moteur d'exécution d'AIR à la fois. En d'autres termes, une version validée ne peut pas cohabiter avec une version de développement.

Pour plus d'informations, voir « Commande ADT installRuntime » à la page 186.

Installation du moteur d'exécution sur un périphérique iOS

Le code du moteur d'exécution d'AIR requis est intégré à chaque application pour périphériques iPhone, iTouch et iPad. Il est inutile d'installer un composant d'exécution distinct.

Voir aussi

« AIR for iOS » à la page 73

Désinstallation d'Adobe AIR

Après l'installation du moteur d'exécution, les procédures suivantes permettent de le désinstaller.

Désinstallation du moteur d'exécution sur un ordinateur Windows

1. Dans le menu Démarrer de Windows, sélectionnez Paramètres > Panneau de configuration.
2. Selon la version de Windows dont vous disposez, ouvrez l'élément du Panneau de configuration intitulé Programmes, Programmes et fonctionnalités ou Ajout ou suppression de programmes.
3. Pour désinstaller le moteur d'exécution, sélectionnez « Adobe AIR ».
4. Cliquez sur le bouton Modifier/Supprimer.

Désinstallation du moteur d'exécution sur un ordinateur Mac

- Double-cliquez sur le programme de désinstallation d'Adobe AIR, situé dans le dossier /Applications/Utilities.

Désinstallation du moteur d'exécution sur un ordinateur Linux

Effectuez l'une des opérations suivantes :

- Sélectionnez la commande « Programme de désinstallation d'Adobe AIR » dans le menu Applications.
- Exécutez le programme d'installation binaire d'AIR avec l'option `-uninstall`.
- Supprimez les packages AIR (`adobeair` et `adobecerts`) avec le gestionnaire de package.

Suppression du moteur d'exécution sur un périphérique Android

1. Ouvrez l'application Réglages sur le périphérique.
2. Touchez l'entrée Adobe AIR sous Applications > Gérer les applications.
3. Touchez le bouton Désinstaller.

Vous disposez également de la commande ADT `-uninstallRuntime`. Pour plus d'informations, voir « Commande ADT uninstallRuntime » à la page 188.

Suppression d'un moteur d'exécution intégré

Pour supprimer un programme d'exécution captif intégré, vous devez supprimer l'application avec laquelle il est installé. Notez que les moteurs d'exécution captifs sont utilisés uniquement pour exécuter l'application d'installation.

Installation et exécution des exemples d'application AIR

Pour installer ou mettre à jour une application AIR, l'utilisateur doit disposer de privilèges d'administration sur l'ordinateur.

Certains exemples d'application illustrent des fonctionnalités AIR. Procédez comme suit pour y accéder et les installer :

1 Téléchargez et exécutez les exemples d'application AIR. Les applications compilées et le code source sont disponibles.

2 Pour télécharger et exécuter un exemple d'application, cliquez sur le bouton Installer maintenant proposé. Vous êtes invité à installer et exécuter l'application.

3 Si vous décidez de télécharger des exemples d'application et de les exécuter ultérieurement, sélectionnez les liens de téléchargement. Vous pouvez à tout moment exécuter une application AIR en procédant comme suit :

- Sous Windows, double-cliquez sur l'icône de l'application sur le bureau ou sélectionnez-la dans le menu Démarrer de Windows.
- Sous Mac OS, double-cliquez sur l'icône de l'application, qui est installée par défaut dans le dossier Applications de votre répertoire utilisateur (tel que Macintosh HD/Users/Jean/Applications/).

Remarque : *pour vérifier si ces instructions ont été mises à jour, voir les Notes de parution d'AIR, qui résident à l'adresse suivante : http://www.adobe.com/go/learn_air_relnotes_fr.*

Mises à jour d'Adobe Air

Adobe met régulièrement Adobe AIR à jour pour ajouter de nouvelles fonctionnalités ou corriger des problèmes mineurs. La fonction de notification et de mise à jour automatiques permet à Adobe d'avertir automatiquement les utilisateurs de la disponibilité d'une version mise à jour d'Adobe AIR.

Les mises à jour d'Adobe AIR garantissent que ce produit fonctionne correctement et contiennent souvent d'importantes modifications liées à la sécurité. Adobe recommande d'effectuer la mise à jour vers toute nouvelle version disponible d'Adobe AIR, surtout lorsqu'elle affecte la sécurité.

Au lancement d'une application AIR, le moteur d'exécution vérifie par défaut la disponibilité d'une mise à jour. Il effectue cette opération si la dernière vérification remonte à plus de deux semaines. Si une mise à jour est disponible, AIR la télécharge en arrière-plan.

Il est possible de désactiver la fonction de mise à jour automatique à l'aide de l'application AIR SettingsManager. Cette application peut être téléchargée à partir de http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air.

Le processus d'installation standard d'Adobe AIR se connecte à http://airinstall.adobe.com pour envoyer des informations de base sur l'environnement d'installation, notamment la version et la langue du système d'exploitation. Ces informations sont transmises une seule fois par installation et permettent à Adobe de confirmer que l'installation a abouti. Aucune information personnelle identifiable n'est collectée ou transmise.

Mise à jour des moteurs d'exécution captifs

Si vous distribuez votre application avec un paquet de moteur d'exécution captif, le moteur d'exécution captif n'est pas mis à jour automatiquement. Pour assurer la sécurité des utilisateurs, vous devez gérer les mises à jour publiées par Adobe et mettre à jour votre application avec la nouvelle version du moteur d'exécution lors de la publication d'une modification de sécurité importante.

Chapitre 3 : Utilisation des API AIR

Adobe® AIR® comprend des fonctionnalités dont ne dispose pas un contenu SWF qui s'exécute dans Adobe® Flash® Player.

Développeurs ActionScript 3.0
Les API Adobe AIR sont décrites dans les deux documents suivants :

- Guide du développeur ActionScript 3.0
- Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash

Développeurs HTML
Si vous créez des applications AIR de type HTML, les API disponibles en JavaScript via le fichier AIRAliases.js (voir Accès aux classes de l'API AIR à partir de JavaScript) sont décrites dans les deux documents suivants :

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

Classes ActionScript 3.0 propres à AIR

Le tableau suivant contient les classes d'exécution propres à Adobe AIR. Elles ne sont pas disponibles pour le contenu SWF s'exécutant dans Adobe® Flash® Player dans le navigateur.

Développeurs HTML
Les classes disponibles en JavaScript via le fichier AIRAliases.js sont décrites dans Adobe AIR API Reference for HTML Developers (disponible en anglais uniquement).

Classe	Package ActionScript 3.0	Ajoutée à la version 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
CameraUI	flash.media	2.5

Classe	Package ActionScript 3.0	Ajoutée à la version AIR
CertificateStatus	flash.security	2.0
CompressionAlgorithm	flash.utils	1.0
DatagramSocket	flash.net	2.0
DatagramSocketDataEvent	flash.events	2.0
DNSResolver	flash.net.dns	2.0
DNSResolverEvent	flash.events	2.0
DockIcon	flash.desktop	1.0
DownloadErrorEvent	air.update.events	1.5
DRMAuthenticateEvent	flash.events	1.0
DRMDeviceGroup	flash.net.drm	3.0
DRMDeviceGroupErrorEvent	flash.net.drm	3.0
DRMDeviceGroupEvent	flash.net.drm	3.0
DRMManagerError	flash.errors	1.5
EncryptedLocalStore	flash.data	1.0
ExtensionContext	flash.external	2.5
File	flash.filesystem	1.0
FileListEvent	flash.events	1.0
FileMode	flash.filesystem	1.0
FileStream	flash.filesystem	1.0
FocusDirection	flash.display	1.0
GameInput	flash.ui	3.0
GameInputControl	flash.ui	3.0
GameInputControlType	flash.ui	3.6 et versions antérieures ; abandonné à partir de la version 3.7
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 et versions antérieures ; abandonné à partir de la version 3.7
GameInputHand	flash.ui	3.6 et versions antérieures ; abandonné à partir de la version 3.7
Geolocation	flash.sensors	2.0
GeolocationEvent	flash.events	2.0
HTMLHistoryItem	flash.html	1.0
HTMLHost	flash.html	1.0

Classe	Package ActionScript 3.0	Ajoutée à la version AIR
HTMLLoader	flash.html	1.0
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

Classe	Package ActionScript 3.0	Ajoutée à la version AIR
NativeWindowRenderMode	flash.display	3.0
NativeWindowResize	flash.display	1.0
NativeWindowSystemChrome	flash.display	1.0
NativeWindowType	flash.display	1.0
NetworkInfo	flash.net	2.0
NetworkInterface	flash.net	2.0
NotificationType	flash.desktop	1.0
OutputProgressEvent	flash.events	1.0
PaperSize	flash.printing	2.0
PrintMethod	flash.printing	2.0
PrintUIOptions	flash.printing	2.0
PTRRecord	flash.net.dns	2.0
ReferencesValidationSetting	flash.security	1.0
ResourceRecord	flash.net.dns	2.0
RevocationCheckSettings	flash.security	1.0
Screen	flash.display	1.0
ScreenMouseEvent	flash.events	1.0
SecureSocket	flash.net	2.0
SecureSocketMonitor	air.net	2.0
ServerSocket	flash.net	2.0
ServerSocketConnectEvent	flash.events	2.0
ServiceMonitor	air.net	1.0
SignatureStatus	flash.security	1.0
SignerTrustSettings	flash.security	1.0
SocketMonitor	air.net	1.0
SoftKeyboardType	flash.text	3.0
SQLCollationType	flash.data	1.0
SQLColumnNameStyle	flash.data	1.0
SQLColumnSchema	flash.data	1.0
SQLConnection	flash.data	1.0
SQLError	flash.errors	1.0
SQLErrorEvent	flash.events	1.0
SQLErrorOperation	flash.errors	1.0
SQLEvent	flash.events	1.0

Classe	Package ActionScript 3.0	Ajoutée à la version AIR
SQLIndexSchema	flash.data	1.0
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 Flash Player contenant des fonctionnalités propres à AIR

Les classes suivantes sont disponibles pour les contenus SWF exécutés dans le navigateur, mais AIR offre des propriétés ou des méthodes supplémentaires :

Package	Classe	Propriété, méthode ou événement	Ajoutée à la version 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
		Événement orientationChange	2.0
		Événement 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

Package	Classe	Propriété, méthode ou événement	Ajoutée à la version 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

Package	Classe	Propriété, méthode ou événement	Ajoutée à la version AIR
flash.net	FileReference	`extension`	1.0
		Événement `httpResponseStatus`	1.0
		`uploadUnencoded()`	1.0
	NetStream	Événement `drmAuthenticate`	1.0
		Événement `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	`événement httpResponseStatus`	1.0

Package	Classe	Propriété, méthode ou événement	Ajoutée à la version AIR
flash.printing	PrintJob	`active`	2.0
		`copies`	2.0
		`firstPage`	2.0
		`isColor`	2.0
		`jobName`	2.0
		`lastPage`	2.0
		`maxPixelsPerInch`	2.0
		`paperArea`	2.0
		`printableArea`	2.0
		`printer`	2.0
		`printers`	2.0
		`selectPaperSize()`	2.0
		`showPageSetupDialog()`	2.0
		`start2()`	2.0
		`supportsPageSetupDialog`	2.0
		`terminate()`	2.0
	PrintJobOptions	`pixelsPerInch`	2.0
		`printMethod`	2.0
flash.system	Capabilities	`languages`	1.1
	LoaderContext	`allowLoadBytesCodeExecution`	1.0
	Security	`APPLICATION`	1.0
flash.ui	KeyLocation	D_PAD	2.5

La plupart de ces nouvelles propriétés et méthodes sont uniquement disponibles pour le contenu situé dans le sandbox de sécurité de l'application AIR. Toutefois, les nouveaux membres des classes URLRequest sont également disponibles pour le contenu exécuté dans d'autres sandbox.

Les méthodes `ByteArray.compress()` et `ByteArray.uncompress()` comprennent chacune un nouveau paramètre `algorithm` permettant de choisir entre les compressions deflate et zlib. Ce paramètre n'est disponible que pour le contenu s'exécutant dans AIR.

Composants Flex propres à AIR

Les composants MX d'Adobe® Flex™ suivants sont disponibles lors du développement d'un contenu pour Adobe AIR :

- FileEvent
- FileSystemComboBox
- FileSystemDataGrid

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

Par ailleurs, Flex 4 comprend les composants AIR spark suivants :

- Window
- WindowedApplication

Pour plus d'informations sur les composants AIR Flex, voir Utilisation des composants AIR de Flex.

Chapitre 4 : Outils de la plate-forme Adobe Flash pour le développement AIR

Vous pouvez développer des applications AIR à l'aide des outils de développement suivants de la plate-forme Adobe Flash.

Développeurs ActionScript 3.0 (Flash et Flex) :

- Adobe Flash Professional (voir Publication pour AIR)
- Kit SDK d'Adobe Flex 3.x et 4.x (voir « Configuration du kit SDK Flex » à la page 19 et « Outil AIR Developer (ADT) » à la page 174)
- Adobe Flash Builder (voir Développement d'applications AIR avec Flash Builder)

Développeurs HTML et Ajax :

- Kit de développement SDK Adobe AIR (voir « Installation du kit de développement SDK AIR » à la page 17 et « Outil AIR Developer (ADT) » à la page 174)
- Adobe Dreamweaver CS3, CS4, CS5 (voir Extension AIR pour Dreamweaver)

Installation du kit de développement SDK AIR

Le kit de développement SDK AIR contient les outils de ligne de commande suivants, qui permettent de lancer et de mettre en package des applications :

Application de débogage du lanceur AIR (ADL) Permet de lancer des applications AIR sans devoir d'abord les installer. Voir « Application de débogage du lanceur AIR (ADL) » à la page 168.

Outil AIR Developer (ADT) Met en package des applications AIR sous la forme de packages d'installation distribuables. Voir « Outil AIR Developer (ADT) » à la page 174.

Les outils de ligne de commande d'AIR nécessitent l'installation de Java sur l'ordinateur. Vous pouvez utiliser la machine virtuelle Java à partir des environnements JRE ou JDK (version 1.5 ou ultérieure). Vous pouvez télécharger ces environnements à l'adresse suivante : http://java.sun.com/.

Vous devez disposer d'au moins 2 Go de mémoire pour exécuter l'outil ADT.

Remarque : *l'utilisateur final n'a pas besoin de Java pour exécuter les applications AIR.*

Pour consulter un aperçu rapide de la création d'une application AIR à l'aide du kit SDK d'AIR, voir « Création d'une première application AIR de type HTML à l'aide du kit SDK d'AIR » à la page 34.

Téléchargement et installation du kit SDK AIR

Pour télécharger et installer le kit SDK AIR, procédez comme suit :

Installation du kit SDK AIR sous Windows

- Téléchargez le fichier d'installation du kit SDK AIR.

- Il est distribué sous la forme d'une archive de fichier standard. Pour installer AIR, extrayez le contenu du kit SDK dans un dossier sur l'ordinateur (exemple : C:\Program Files\Adobe\AIRSDK ou C:\AIRSDK).
- Les outils ADL et ADT figurent dans le dossier bin du kit SDK AIR. Ajoutez le chemin de ce dossier à la variable d'environnement PATH.

Installation du kit SDK AIR sous Mac OS X

- Téléchargez le fichier d'installation du kit SDK AIR.
- Il est distribué sous la forme d'une archive de fichier standard. Pour installer AIR, extrayez le contenu du kit SDK dans un dossier sur l'ordinateur (exemple : /Users/<nomUtilisateur>/Applications/AIRSDK).
- Les outils ADL et ADT figurent dans le dossier bin du kit SDK AIR. Ajoutez le chemin de ce dossier à la variable d'environnement PATH.

Installation du kit SDK AIR sous Linux

- Le kit SDK est disponible au format tbz2.
- Pour installer le kit SDK, créez un dossier pour le décompresser, puis entrez la commande suivante : tar -jxvf <chemin d'accès à AIR-SDK.tbz2>

Pour plus d'informations sur l'initiation aux outils du kit SDK AIR, voir Création d'une application AIR à l'aide des outils de ligne de commande.

Contenu du kit SDK AIR

Le tableau suivant décrit les fichiers que contient le kit SDK AIR :

Dossier du kit SDK	Description des fichiers/outils
bin	L'application de débogage du lanceur AIR (ADL) permet d'exécuter une application AIR sans la mettre en package et l'installer au préalable. Pour plus d'informations sur l'utilisation de cette application, voir « Application de débogage du lanceur AIR (ADL) » à la page 168. L'outil AIR Developer (ADT) met en package une application sous la forme d'un fichier AIR distribuable. Pour plus d'informations sur l'utilisation de cet outil, voir « Outil AIR Developer (ADT) » à la page 174.
frameworks	Le répertoire libs contient des bibliothèques de code destinées aux applications AIR. Le répertoire projects contient le code des bibliothèques SWF et SWC compilées.
include	Le répertoire include contient un fichier d'en-tête de langage C pour l'écriture d'extensions natives.
install	Le répertoire install contient les pilotes USB Windows associés aux périphériques Android. (Ces pilotes sont fournis par Google dans le kit SDK d'Android.)
lib	Contient le code de prise en charge des outils du kit SDK d'AIR.

Dossier du kit SDK	Description des fichiers/outils
runtimes	Moteurs d'exécution d'AIR destinés au bureau et aux périphériques mobiles. L'application de débogage du lanceur AIR (ADL) utilise le moteur d'exécution pour lancer les applications AIR avant leur mise en package ou leur installation. Vous pouvez installer les moteurs d'exécution d'AIR for Android (packages APK) sur des émulateurs ou des périphériques Android à des fins de développement ou de test. Utilisez un package APK distinct pour les périphériques et les émulateurs. (Vous pouvez télécharger le moteur d'exécution d'AIR for Android public à partir d'Android Market.)
samples	Ce dossier contient un exemple de fichier descripteur d'application, un exemple de fonction d'installation transparente (badge.swf) et les icônes d'application AIR par défaut.
templates	descriptor-template.xml : modèle du fichier descripteur d'application, que requiert chaque application AIR. Pour une description détaillée du fichier descripteur d'application, voir « Fichiers descripteurs d'applications AIR » à la page 217. Ce dossier contient également les fichiers de schéma associés à la structure XML du fichier descripteur d'application de chaque version d'AIR.

Configuration du kit SDK Flex

Pour développer des applications Adobe® AIR® dans Adobe® Flex™, vous disposez des options suivantes :

- Vous pouvez télécharger et installer Adobe® Flash® Builder™, qui intègre des outils permettant de créer des projets Adobe AIR, ainsi que de tester, déboguer et mettre en package les applications AIR. Voir « Création d'une première application de bureau AIR Flex dans Flash Builder » à la page 21.
- Vous pouvez télécharger le kit SDK d'Adobe® Flex™ et développer des applications AIR Flex à l'aide de votre éditeur de texte et de vos outils de ligne de commande favoris.

Pour consulter un aperçu rapide de la création d'une application AIR à l'aide du kit SDK de Flex, voir « Création d'une première application de bureau AIR à l'aide du kit SDK de Flex » à la page 38.

Installation du kit SDK Flex

Pour pouvoir créer des applications AIR à l'aide des outils de ligne de commande, Java doit être installé sur l'ordinateur. Vous pouvez utiliser la machine virtuelle Java à partir des environnements JRE ou JDK (version 1.5 ou ultérieure). Vous pouvez télécharger ces environnements à l'adresse suivante : http://java.sun.com/.

Remarque : *l'utilisateur final n'a pas besoin de Java pour exécuter les applications AIR.*

Le kit SDK de Flex offre l'API d'AIR et les outils de ligne de commande nécessaires pour mettre en package, compiler et déboguer les applications AIR.

1. Le cas échéant, téléchargez le kit SDK Flex à partir de http://opensource.adobe.com/wiki/display/flexsdk/Downloads.
2. Placez le contenu du kit SDK dans un dossier (Flex SDK, par exemple).
3. Copiez le contenu du kit SDK d'AIR sur les fichiers du kit SDK de Flex.

Remarque : *sur un ordinateur Mac, veillez à copier ou remplacer les fichiers des dossiers du kit SDK, plutôt que des répertoires entiers. Par défaut, copier un répertoire sur un répertoire du même nom sur un ordinateur Mac supprime les fichiers existants du répertoire cible, plutôt que de fusionner le contenu des deux répertoires. Vous pouvez utiliser la commande* `ditto` *dans une fenêtre de terminal pour fusionner le kit SDK d'AIR et le kit SDK de Flex :* `ditto air_sdk_folder flex_sdk_folder`

4 Les outils de ligne de commande AIR figurent dans le dossier bin.

Configuration de kits SDK externes

Le développement d'applications pour Android et iOS nécessite de télécharger des fichiers de configuration, des kits SDK et autres outils de développement à partir des éditeurs de la plate-forme.

Pour plus d'informations sur le téléchargement et l'installation du kit SDK d'Android, voir Android Developers : Installing the SDK (disponible en anglais uniquement). Depuis AIR 2.6, il n'est plus obligatoire de télécharger le kit SDK d'Android. Le kit SDK d'AIR comprend à présent les composants de base requis pour installer et lancer des packages APK. Le kit SDK d'Android s'avère toutefois utile pour diverses tâches de développement, notamment la création et l'exécution d'émulateurs logiciels, ainsi que la capture d'écrans de périphérique.

Le développement d'application iOS ne requiert pas de kit SDK externe. Vous devez toutefois disposer de certificats et fichiers de configuration spéciaux. Pour plus d'informations, voir Obtention de fichiers de développement auprès d'Apple.

Chapitre 5 : Création d'une première application AIR

Création d'une première application de bureau AIR Flex dans Flash Builder

Pour vous familiariser rapidement avec le fonctionnement d'Adobe® AIR®, suivez les instructions ci-dessous, qui permettent de créer et de mettre en package une application AIR SWF simple, appelée « Hello World », à l'aide d'Adobe® Flash® Builder.

Le cas échéant, téléchargez et installez Flash Builder. Téléchargez et installez également la version la plus récente d'Adobe AIR à partir de l'adresse suivante : www.adobe.com/go/air_fr.

Création d'un projet AIR

Flash Builder propose des outils de développement et de mise en package d'applications AIR.

Pour créer une application AIR dans Flash Builder ou Flex Builder, vous commencez comme s'il s'agissait de tout autre projet d'application Flex, c-à-d. en définissant un nouveau projet.

1. Ouvrez Flash Builder.
2. Sélectionnez Fichier > Nouveau > Projet Flex.
3. Attribuez au projet le nom AIRHelloWorld.
4. Dans Flex, les applications AIR sont considérées comme un type d'application. Vous disposez de deux options de type :
   - Application Web qui s'exécute dans Adobe® Flash® Player
   - Application de bureau qui s'exécute dans Adobe AIR

   Sélectionnez le type d'application de bureau.
5. Cliquez sur Terminer pour créer le projet.

Les projets AIR se composent initialement de deux fichiers : le fichier MXML principal et un fichier XML d'application (ou fichier descripteur d'application). Le second fichier spécifie les propriétés d'une application.

Pour plus d'informations, voir Développement d'applications AIR avec Flash Builder.

Ecriture du code de l'application AIR

Pour écrire le code de l'application « Hello World », vous modifiez le fichier MXML de l'application, qui est ouvert dans l'éditeur. (Le cas échéant, ouvrez le fichier dans l'explorateur de projets.)

Les applications AIR Flex de bureau sont contenues dans la balise MXML WindowedApplication. Celle-ci crée une fenêtre simple qui comprend des contrôles de fenêtre de base, tels qu'une barre de titre et un bouton de fermeture.

1. Ajoutez un attribut `title` au composant WindowedApplication et affectez-lui la valeur `"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 Ajoutez un composant Label à l'application (en le plaçant dans la balise WindowedApplication). Définissez la propriété `text` du composant Label sur `"Hello AIR"` et stipulez qu'il doit être centré, comme illustré ci-dessous :

```
<?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 Ajoutez le bloc de style suivant juste après la balise WindowedApplication d'ouverture et avant la balise du composant Label que vous venez d'entrer :

```
<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>
```

Ces paramètres de style s'appliquent à l'intégralité de l'application et définissent un arrière-plan de fenêtre gris légèrement transparent.

Le code de l'application se présente à présent comme suit :

```
<?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>
```

Vous allez maintenant modifier certains paramètres dans le descripteur de l'application pour que celle-ci soit transparente :

1 Dans le panneau Navigation de Flex, recherchez le fichier descripteur d'application dans le répertoire source du projet. Si vous avez nommé le projet AIRHelloWorld, ce fichier s'appelle AIRHelloWorld-app.xml.

2 Double-cliquez sur le fichier descripteur d'application pour le modifier dans Flash Builder.

3 Dans le code XML, recherchez les lignes de commentaire des propriétés `systemChrome` et `transparent` (de la propriété `initialWindow`). Supprimez les commentaires (autrement dit, supprimez les séparateurs « `<!--` » et « `-->` »).

4 Définissez la valeur de texte de la propriété `systemChrome` sur `none`, comme indiqué ci-après :

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

5 Définissez la valeur de texte de la propriété `transparent` sur `true`, comme indiqué ci-après :

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

6 Enregistrez le fichier.

Test de l'application AIR

Pour tester le code d'application que vous venez d'écrire, exécutez-le en mode de débogage.

1 Cliquez sur le bouton de débogage dans la barre d'outils principale.

Vous pouvez aussi sélectionner la commande Run > Debug > AIRHelloWorld.

L'application AIR qui en résulte s'apparente à l'exemple suivant :

2 Grâce aux propriétés `horizontalCenter` et `verticalCenter` du contrôle Label, le texte est placé au centre de la fenêtre. Vous pouvez déplacer ou redimensionner la fenêtre comme pour toute autre application de bureau.

Remarque : *si la compilation de l'application échoue, corrigez les erreurs de syntaxe ou les fautes d'orthographe introduites par inadvertance dans le code. Dans Flash Builder, les erreurs et les avertissements s'affichent en mode Problems.*

Mise en package, signature et exécution de l'application AIR

Vous pouvez maintenant mettre l'application « Hello World » en package sous forme de fichier AIR pour la distribuer. Un fichier AIR est un fichier d'archives contenant les fichiers de l'application ; autrement dit, tous les fichiers qui figurent dans le dossier bin du projet. Dans cet exemple simple, il s'agit des fichiers SWF et XML. Vous distribuez le package AIR aux utilisateurs, qui s'en servent pour installer l'application. Dans le cadre de ce processus, il est impératif de signer le package numériquement.

1. Assurez-vous que l'application est exempte d'erreurs de compilation et s'exécute comme il se doit.
2. Sélectionnez Projet > Exporter vers une version validée.
3. Vérifiez que le projet correspond à AIRHelloWorld et l'application à AIRHelloWorld.mxml.
4. Sélectionnez l'option d'exportation en tant que package signé. Cliquez ensuite sur Suivant.
5. Si vous disposez déjà d'un certificat numérique, cliquez sur Parcourir pour y accéder, puis sélectionnez-le.
6. Si vous devez créer un certificat numérique auto-signé, sélectionnez Créer.
7. Spécifiez les informations requises et cliquez sur OK.
8. Cliquez sur Terminer pour générer le package AIR, qui s'appelle AIRHelloWorld.air.

Vous pouvez maintenant installer et exécuter l'application à partir de l'explorateur de projets dans Flash Builder ou en double-cliquant sur le fichier AIR dans le système de fichiers.

Création d'une première application de bureau AIR dans Flash Professional

Cette rubrique propose une brève démonstration pratique du fonctionnement d'Adobe® AIR® en vous aidant à créer et à mettre en package une application AIR simple nommée « Hello World » à l'aide d'Adobe® Flash® Professional.

Le cas échéant, téléchargez et installez Adobe AIR, à l'adresse suivante : www.adobe.com/go/air_fr.

Création de l'application Hello World dans Flash

La création d'une application Adobe AIR dans Flash ressemble beaucoup à la création d'un autre fichier FLA. La procédure suivante vous guide tout au long du processus de création de l'application Hello World simple avec Flash Professional.

Pour créer l'application Hello World

1. Démarrez Flash.
2. Dans l'écran de bienvenue, cliquez sur AIR pour créer un fichier FLA vide avec les paramètres de publication d'Adobe AIR.
3. Sélectionnez l'outil Texte dans le panneau Outils et créez un champ de texte statique (par défaut) au centre de la scène. Choisissez une largeur pouvant contenir 15 à 20 caractères.
4. Entrez le texte « Hello World » dans le champ.
5. Enregistrez le fichier en lui donnant un nom (HelloAIR, par exemple).

Test de l'application

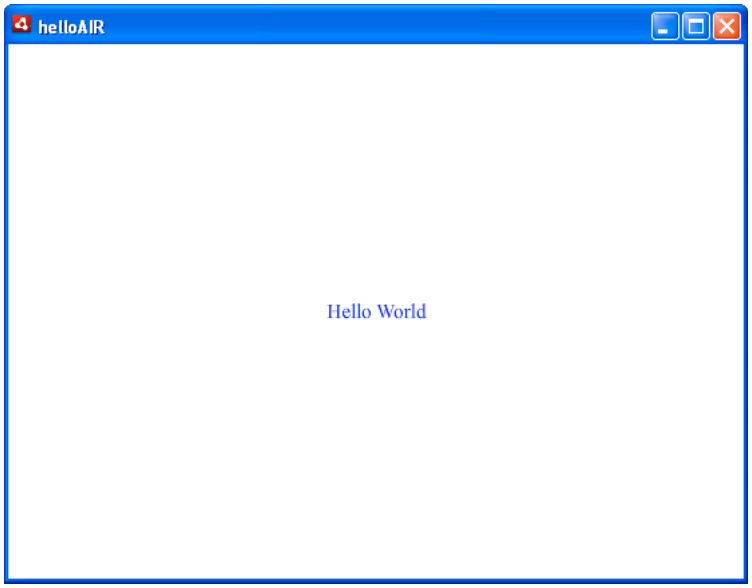
1 Appuyez sur Ctrl + Entrée ou sélectionnez Contrôle > Tester l'animation > Tester pour tester l'application dans Adobe AIR.

2 Pour utiliser la fonction Déboguer l'animation, ajoutez d'abord du code ActionScript à l'application. Vous pouvez le faire rapidement en ajoutant une instruction trace telle que :

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

3 Appuyez sur Ctrl + Maj + Entrée ou sélectionnez Déboguer > Déboguer l'animation > Déboguer pour exécuter l'application avec Déboguer l'animation.

L'application Hello World ressemble à l'illustration suivante :



Mise en package de l'application

1 Sélectionnez Fichier > Publier.

2 Signez le package Adobe AIR à l'aide d'un certificat numérique existant ou créez un certificat auto-signé en procédant comme suit :

a Cliquez sur le bouton Créer en regard du champ Certificat.

b Renseignez les entrées Nom de l'éditeur, Unité d'organisation, Nom de l'organisation, E-mail, Pays, Mot de passe et Confirmer le mot de passe.

c Spécifiez le type de certificat. L'option Type du certificat fait référence au niveau de sécurité : 1024-RSA utilise une clé 1 024 bits (moins sécurisée) et 2048-RSA une clé 2 048 bits (plus sécurisée).

d Enregistrez les informations dans un fichier de certificat en renseignant l'entrée Enregistrer sous ou en cliquant sur le bouton Parcourir... pour localiser un dossier. (Par exemple, *C:/Temp/mycert.pfx*). Lorsque vous avez terminé, cliquez sur OK.

e Flash vous renvoie à la boîte de dialogue Signature numérique. Le chemin et le nom de fichier du certificat auto-signé s'affichent dans le champ de texte Certificat. Si ce n'est pas le cas, entrez le chemin et le nom du fichier ou cliquez sur le bouton Parcourir pour le localiser et le sélectionner.

f Entrez le mot de passe défini à l'étape b dans le champ de texte Mot de passe de la boîte de dialogue Signature numérique. Pour plus d'informations sur la signature des applications Adobe AIR, voir « Signature numérique d'un fichier AIR » à la page 200.

3 Pour créer le fichier de l'application et du programme d'installation, cliquez sur le bouton Publier. (Dans Flash CS4 et CS5, cliquez sur le bouton OK.) Pour créer les fichiers SWF et application.xml, vous devez tester ou déboguer l'animation avant de créer le fichier AIR.

4 Pour installer l'application, double-cliquez sur le fichier AIR (*application.air*) dans le dossier où vous avez enregistré l'application.

5 Dans la boîte de dialogue Installation de l'application, cliquez sur le bouton Installer.

6 Vérifiez les paramètres d'emplacement et de préférences d'installation et assurez-vous que la case à cocher « Démarrer l'application à la fin de l'installation » est activée. Cliquez sur Continuer.

7 Lorsque le message Installation terminée apparaît, cliquez sur Terminer.

Création d'une première application AIR for Android dans Flash Professional

Pour développer des applications AIR for Android, vous devez télécharger l'extension Flash Professional CS5 pour Android à l'adresse suivante : Adobe Labs.

Vous devez également télécharger et installer le kit SDK d'Android à partir du site Web d'Android comme indiqué dans : Android Developers : Installing the SDK (disponible en anglais uniquement).

Création d'un projet

1 Ouvrez Flash Professional CS5.

2 Créez un projet AIR for Android.

L'écran d'accueil de Flash Professional contient un lien permettant de créer une application AIR for Android. Vous pouvez également sélectionner Fichier > Nouveau, puis le modèle AIR for Android.

3 Enregistrez le document sous le nom HelloWorld.fla.

Programmation du code

Etant donné que ce didacticiel n'est pas consacré à la programmation du code, contentez-vous de programmer « Hello, World! » sur la scène à l'aide de l'outil Texte.

Définition des propriétés de l'application

1 Sélectionnez Fichier > Paramètres d'AIR for Android.

2 Sur l'onglet Général, définissez les paramètres suivants :

- Fichier de sortie : HelloWorld.apk
- Nom de l'application : HelloWorld
- ID de l'application : HelloWorld
- Version : 0.0.1
- Format : Portrait

3 Sur l'onglet Déploiement, définissez les paramètres suivants :

- Certificat : pointez vers un certificat de signature du code AIR valide. Vous pouvez cliquer sur le bouton Créer pour créer un certificat. (Les applications Android déployées via Android Market doivent disposer de certificats valides jusqu'en 2033 au moins.) Entrez le mot de passe associé au certificat dans le champ Mot de passe.
- Type de déploiement Android : Déboguer
- Après la publication : sélectionnez les deux options.
- Entrez le chemin d'accès à l'outil ADB dans le sous-répertoire tools du kit SDK d'Android.

4 Cliquez sur OK pour fermer la boîte de dialogue Paramètres AIR for Android.

A ce stade de son développement, l'application ne requiert ni icônes, ni autorisations. La plupart des applications AIR for Android nécessitent certaines autorisations pour accéder aux fonctionnalités protégées. Contentez-vous de définir les autorisations réellement requises par l'application, car les utilisateurs risquent de la rejeter si elle demande un nombre trop élevé d'autorisations.

5 Enregistrez le fichier.

Mise en package et installation de l'application sur le périphérique Android

1 Veillez à activer le débogage USB sur le périphérique. Vous pouvez activer le débogage USB dans l'application Réglages via Applications > Développement.

2 Connectez le périphérique à l'ordinateur via un câble USB.

3 Le cas échéant, installez le moteur d'exécution d'AIR. Pour ce faire, accédez à Android Market et téléchargez Adobe AIR. (Vous pouvez également installer AIR localement à l'aide de la commande ADT « Commande ADT installRuntime » à la page 186. Le kit SDK d'Android contient les packages Android destinés aux émulateurs et périphériques Android.)

4 Sélectionnez Fichier > Publier.

Flash Professional crée le fichier APK, installe l'application sur le périphérique Android connecté et la lance.

Création d'une première application AIR for iOS

AIR 2.6 ou ultérieur, iOS 4.2 ou ultérieur

Vous pouvez programmer le code des fonctionnalités d'une application iOS, les créer et les tester par le biais d'outils Adobe uniquement. Toutefois, pour installer une application iOS sur un périphérique et la distribuer, vous devez appartenir au programme iOS Developer d'Apple (service payant). Une fois membre du programme iOS Developer, vous pouvez accéder au portail iOS Provisioning Portal, où vous pouvez obtenir d'Apple les éléments et fichiers suivants, dont vous devez disposer pour installer une application sur un périphérique à des fins de test et de distribution ultérieure :

- Certificats de développement et de distribution
- ID d'application
- Fichiers de configuration pour le développement et la distribution d'applications

Création du contenu de l'application

Créez un fichier SWF qui affiche le texte « Hello world! » Vous disposez à cet effet de Flash Professional, Flash Builder ou d'un autre IDE. Cet exemple fait simplement appel à un éditeur de texte et au compilateur SWF de ligne de commande intégré au kit SDK de Flex.

1 Créez un répertoire destiné aux fichiers de l'application à un emplacement adéquat. Créez le fichier *HelloWorld.as* et modifiez-le dans l'éditeur de code de votre choix.

2 Ajoutez le code suivant :

```
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 Compilez la classe à l'aide du compilateur amxmlc :

```
amxmlc HelloWorld.as
```

Un fichier SWF, *HelloWorld.swf*, est créé dans le même dossier.

Remarque : *cet exemple considère comme acquis que vous avez défini la variable d'environnement path de sorte à inclure le répertoire dans lequel réside amxmlc. Pour plus d'informations sur la définition de la variable path, voir* « *Variables d'environnement path* » *à la page 321*. *Vous pouvez également saisir le chemin d'accès complet à amxmlc et aux autres outils de ligne de commande utilisés dans cet exemple.*

Création des icônes et des graphiques de l'écran initial de l'application

Toutes les applications iOS contiennent des icônes affichées dans l'interface utilisateur de l'application iTunes et sur l'écran du périphérique.

1 Créez un répertoire au sein du répertoire de projet et affectez-lui le nom « icons ».

2 Créez trois fichiers PNG dans le répertoire icons, Nommez-les Icon_29.png, Icon_57.png et Icon_512.png.

3 Modifiez les fichiers PNG pour créer des graphiques adaptés à l'application. Les fichiers doivent mesurer 29 pixels sur 29 pixels, 57 pixels sur 57 pixels et 512 pixels sur 512 pixels. Dans le cadre de ce test, vous pouvez vous contenter de carrés de couleur unie.

Remarque : *lorsque vous envoyez une application à l'App Store d'Apple, vous utilisez une version JPG (et non PNG) du fichier de 512 pixels. La version PNG est réservée au test des versions de développement d'une application.*

Toute application iPhone affiche une image initiale lors de son chargement sur l'iPhone. Vous définissez cette image initiale dans un fichier PNG, comme suit :

1. Dans le répertoire de développement principal, créez un fichier PNG appelé Default.png. (Ne placez *pas* ce fichier dans le sous-répertoire icons et respectez la casse du nom du fichier.)
2. Modifiez le fichier de sorte que ses dimensions correspondent à 320 pixels de large sur 480 pixels de haut. Pour le moment, contentez-vous d'un contenu composé d'un rectangle blanc uni, que vous modifierez ultérieurement.

Pour plus d'informations sur ces graphiques, voir « Icônes d'une application » à la page 92.

Création du fichier descripteur de l'application

Créez un fichier descripteur de l'application qui spécifie les propriétés de base de cette dernière. Vous disposez à cet effet d'un IDE tel que Flash Builder ou un éditeur de texte.

1. Dans le dossier du projet qui contient HelloWorld.as, créez un fichier XML, *HelloWorld-app.xml*. Modifiez ce fichier dans l'éditeur XML de votre choix.
2. Ajoutez le code XML suivant :

```
<?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>
```

   Par souci de simplicité, cet exemple ne définit que quelques-unes des propriétés disponibles.

   Remarque : *Si vous utilisez AIR 2 ou une version antérieure, vous devez utiliser l'élément* `<version>` *et non l'élément* `<versionNumber>`.

3. Remplacez l'ID de l'application par l'ID spécifié dans le portail iOS Provisioning Portal. (N'incluez pas de préfixe au début de l'ID d'application.)
4. Testez l'application avec l'application ADL :

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

   L'application ADL devrait ouvrir une fenêtre sur le bureau dans laquelle figure le texte *Hello World!*. Si tel n'est pas le cas, vérifiez si le code source et le fichier descripteur d'application contiennent des erreurs.

Compilation du fichier IPA

Vous pouvez maintenant compiler le fichier d'installation IPA à l'aide de l'outil ADT, comme suit : Vous devez disposer de la clé privée et du certificat de développement Apple au format P12, ainsi que du profil de configuration pour le développement Apple.

Exécutez l'outil ADT en spécifiant les options suivantes (remplacez les valeurs keystore, storepass et provisioning-profile par vos propres valeurs) :

```
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
```

(Utilisez une ligne de commande unique. Les sauts de ligne qui figurent dans cet exemple ont pour unique objet de faciliter la lecture du code.)

L'outil ADT génère le fichier d'installation de l'application iOS, *HelloWorld.ipa*, dans le répertoire du projet. La compilation du fichier IPA prend parfois quelques minutes.

Installation de l'application sur un périphérique

Pour installer l'application iOS à des fins de test :

1. Ouvrez l'application iTunes.
2. Le cas échéant, ajoutez à iTunes le profil de configuration associé à l'application. Dans iTunes, sélectionnez Fichier > Ajouter le fichier à la bibliothèque. Sélectionnez ensuite le fichier du profil de configuration (dont le type de fichier correspond à mobileprovision).

   Utilisez à présent le profil de configuration pour le développement afin de tester l'application sur le périphérique de développement.

   Lors de la distribution ultérieure de l'application sur l'iTunes Store, vous utiliserez le profil de distribution. Pour procéder à une distribution ad hoc de l'application (en d'autres termes, pour la distribuer sur plusieurs périphériques sans passer par l'iTunes Store), utilisez le profil de configuration ad hoc.

   Pour plus d'informations sur les profils de configuration, voir « Configuration d'iOS » à la page 69.
3. Certaines versions d'iTunes ne remplacent pas l'application si une version identique de l'application est déjà installée. Dans ce cas de figure, supprimez l'application du périphérique et de la liste d'applications dans iTunes.
4. Double-cliquez sur le fichier IPA associé à l'application. Elle devrait apparaître dans la liste d'applications d'iTunes.
5. Connectez le périphérique au port USB de l'ordinateur.
6. Dans iTunes, vérifiez sur l'onglet Application associé au périphérique que l'application est sélectionnée dans la liste d'applications à installer.
7. Sélectionnez le périphérique dans la liste de gauche d'applications. Cliquez ensuite sur le bouton Synchroniser. Une fois la synchronisation terminée, l'application Hello World apparaît sur l'iPhone.

Si la nouvelle version n'est pas installée, supprimez-la du périphérique et de la liste d'applications dans iTunes, puis répétez la procédure. Ce cas de figure se produit parfois si la version actuellement installée utilise le même ID et le même numéro.

Modification des graphiques de l'écran initial

Avant de compiler l'application, vous avez créé le fichier Default.png (voir « Création des icônes et des graphiques de l'écran initial de l'application » à la page 28). Ce fichier PNG contient l'image de démarrage affichée lors du chargement de l'application. Lorsque vous avez testé l'application sur l'iPhone, peut-être avez-vous remarqué l'écran vide au démarrage.

Remplacez cette image par l'écran de démarrage de l'application (« Hello World! ») en procédant comme suit :

1. Ouvrez l'application sur le périphérique. Lorsque la première occurrence du texte « Hello World » apparaît, appuyez sur le bouton principal (figurant sous l'écran) et maintenez-le enfoncé. Tout en maintenant appuyé le bouton principal, appuyez sur le bouton Marche/Veille figurant dans la partie supérieure de l'iPhone. Vous effectuez ainsi une capture d'écran, qui est envoyée à Pellicule.
2. Transférez l'image sur l'ordinateur de développement via iPhoto ou toute autre application adaptée. (Sous Mac OS, vous disposez également de l'application Transfert d'images.)

   Vous pouvez aussi envoyer la photo par E-mail à l'ordinateur de développement, comme suit :

   - Ouvrez l'application Photos.
   - Ouvrez Pellicule.
   - Ouvrez la capture d'écran que vous avez effectuée.
   - Touchez l'image, puis le bouton fléché figurant dans l'angle inférieur gauche. Cliquez ensuite sur le bouton Envoyer par courrier et envoyez l'image à votre propre adresse électronique.
3. Remplacez le fichier Default.png, qui réside dans le répertoire de développement, par une version PNG de la capture d'écran effectuée.
4. Recompilez l'application (voir « Compilation du fichier IPA » à la page 30) et installez-la à nouveau sur le périphérique.

L'application utilise à présent le nouvel écran de démarrage au chargement.

Remarque : *sous réserve de respecter les dimensions requises (320 x 480 pixels), libre à vous de créer n'importe quel graphique pour le fichier Default.png. Il est toutefois préférable que l'image issue du fichier Default.png corresponde à l'état initial de l'application.*

Création d'une première application AIR de type HTML dans Dreamweaver

Pour vous familiariser rapidement avec le fonctionnement d'Adobe® AIR®, suivez les instructions ci-dessous, qui permettent de créer et de mettre en package une application AIR HTML simple, appelée « Hello World » à l'aide de l'extension Adobe® AIR® pour Dreamweaver®.

Le cas échéant, téléchargez et installez Adobe AIR, à l'adresse suivante : www.adobe.com/go/air_fr.

Pour plus d'informations sur l'installation de l'extension Adobe AIR pour Dreamweaver, voir Installation de l'extension Adobe AIR pour Dreamweaver.

Pour une présentation de l'extension, y compris la configuration système requise, voir Extension AIR pour Dreamweaver.

Remarque : *vous ne pouvez développer des applications AIR de type HTML que pour les profils desktop et extendedDesktop. Le profil mobile n'est pas pris en charge.*

Préparation des fichiers de l'application

La première page et toutes les pages connexes de l'application Adobe AIR doivent être définies dans un site Dreamweaver, comme suit :

1. Démarrez Dreamweaver et assurez-vous qu'un site est défini.
2. Pour ouvrir une nouvelle page HTML, sélectionnez Fichier > Nouveau, choisissez HTML dans la colonne Type de page et Sans dans la colonne Mise en forme, puis cliquez sur Créer.
3. Dans la nouvelle page, tapez **Hello World!**

   Cet exemple est extrêmement simple, mais vous avez tout loisir d'appliquer un style au texte, d'ajouter du contenu supplémentaire à la page, de lier d'autres pages à cette première page, etc.
4. Enregistrez la page (Fichier > Enregistrer) sous le nom hello_world.html. Veillez à enregistrer la page dans un site Dreamweaver.

Pour plus d'informations sur les sites Dreamweaver, voir l'aide de Dreamweaver.

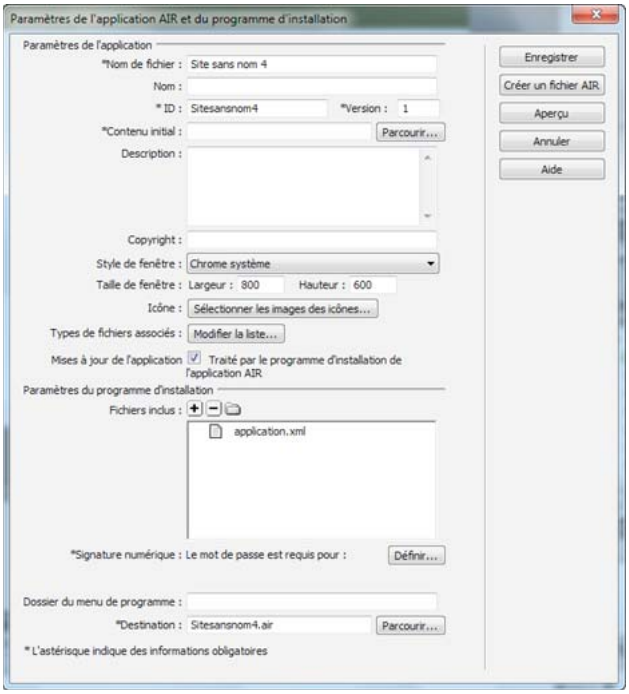
Création de l'application Adobe AIR

1. Assurez-vous que la page hello_world.html est ouverte dans la fenêtre de document Dreamweaver (voir la section précédente pour plus d'informations sur la procédure de création de cette page).
2. Sélectionnez Site > Paramètres de l'application AIR.

   La plupart des paramètres obligatoires de la boîte de dialogue Paramètres de l'application AIR et du programme d'installation sont automatiquement renseignés. Vous devez cependant sélectionner le contenu initial (première page) de votre application.
3. Cliquez sur le bouton Parcourir en regard de l'option Contenu initial, recherchez la page hello_world.html et sélectionnez-la.
4. En regard de l'option Signature numérique, cliquez sur le bouton Définir.

   Une signature numérique garantit que le code d'une application n'a pas été altéré ni endommagé depuis sa création par le développeur, et il est obligatoire d'en définir une pour toutes les applications Adobe AIR.
5. Dans la boîte de dialogue Signature numérique, sélectionnez l'option Signer le package AIR avec un certificat numérique et cliquez sur le bouton Créer (si vous avez déjà accès à un certificat numérique, vous pouvez cliquer sur le bouton Parcourir pour le sélectionner).
6. Renseignez les champs obligatoires de la boîte de dialogue Certificat numérique auto-signé. Vous devez indiquer votre nom, entrez et confirmez un mot de passe et attribuez un nom au fichier de certificat numérique. Dreamweaver enregistre le certificat numérique dans la racine de votre site.
7. Cliquez sur OK pour revenir à boîte de dialogue Signature numérique.
8. Dans la boîte de dialogue Signature numérique, entrez le mot de passe spécifié pour le certificat numérique et cliquez sur OK.

La boîte de dialogue Paramètres de l'application AIR et du programme d'installation se présente maintenant comme suit, par exemple :



Pour plus d'informations sur toutes les options de la boîte de dialogue et leur modification, voir Création d'une application AIR dans Dreamweaver.

9 Cliquez sur le bouton Créer un fichier AIR.

Dreamweaver crée le fichier de l'application Adobe AIR et l'enregistre dans le dossier racine du site. Il crée également le fichier application.xml au même emplacement. Ce fichier sert de manifeste et définit différentes propriétés de l'application.

Installation de l'application sur un bureau

Maintenant que vous avez créé le fichier de l'application, vous pouvez l'installer sur un bureau.

1 Transférez le fichier de l'application Adobe AIR du site Dreamweaver vers votre bureau ou tout autre bureau.

Cette étape est facultative. Si vous le souhaitez, vous pouvez installer la nouvelle application sur l'ordinateur à partir du répertoire du site Dreamweaver.

2 Double-cliquez sur le fichier exécutable de l'application (fichier .air) pour installer celle-ci.

Aperçu de l'application Adobe AIR

Vous pouvez afficher un aperçu des pages constituant une application AIR à tout moment. Vous n'êtes donc pas obligé de mettre l'application en package pour voir comment elle se présentera une fois installée.

1 Le cas échéant, ouvrez la page hello_world.html dans la fenêtre de document Dreamweaver.

2 Sur la barre d'outils Document, cliquez sur le bouton Aperçu/Débogage dans le navigateur, puis sélectionnez Aperçu dans AIR.

Vous pouvez aussi appuyer sur Ctrl+Maj+F12 (Windows) ou Cmd+Maj+F12 (Macintosh).

Lorsque vous affichez un aperçu de cette page, son apparence est identique à ce que voit l'utilisateur une fois qu'il a installé l'application sur un bureau.

Création d'une première application AIR de type HTML à l'aide du kit SDK d'AIR

Pour vous familiariser rapidement avec le fonctionnement d'Adobe® AIR®, suivez les instructions ci-dessous, qui permettent de créer et de mettre en package une application AIR HTML simple, appelée « Hello World ».

Avant de commencer, vous devez installer le moteur d'exécution et configurer le kit SDK AIR. Dans ce didacticiel, vous allez utiliser l'*application de débogage du lanceur AIR* et l'*outil AIR Developer* (ADT). Les outils ADL et ADT sont des programmes de ligne de commande et résident dans le répertoire `bin` du kit SDK AIR (voir « Installation du kit de développement SDK AIR » à la page 17). Dans ce didacticiel, il est considéré comme acquis que vous savez exécuter des programmes à partir de la ligne de commande et configurer les variables d'environnement path requises par le système d'exploitation.

Remarque : *si vous utilisez Adobe® Dreamweaver®, voir « Création d'une première application AIR de type HTML dans Dreamweaver » à la page 31.*

Remarque : *vous ne pouvez développer des applications AIR de type HTML que pour les profils desktop et extendedDesktop. Le profil mobile n'est pas pris en charge.*

Création des fichiers du projet

Chaque projet AIR HTML doit contenir les deux fichiers suivants : un fichier descripteur d'application, qui définit les métadonnées de l'application, et une page HTML de niveau supérieur. Outre ces fichiers obligatoires, le projet comprend également un fichier de code JavaScript, `AIRAliases.js`, qui définit des variables d'alias pratiques pour les classes des API AIR.

1. Créez le répertoire `HelloWorld` qui contiendra les fichiers du projet.
2. Créez le fichier XML `HelloWorld-app.xml`.
3. Créez le fichier HTML `HelloWorld.html`.
4. Copiez `AIRAliases.js` du dossier frameworks du kit SDK AIR vers le répertoire du projet.

Création du fichier descripteur d'application AIR

Pour commencer à programmer l'application AIR, créez un fichier descripteur d'application XML contenant la structure suivante :

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

1 Ouvrez le fichier `HelloWorld-app.xml` en vue de le modifier.

2 Ajoutez l'élément `<application>` racine avec pour attribut l'espace de noms AIR :

<application xmlns="http://ns.adobe.com/air/application/2.7"> Le dernier segment de l'espace de noms, « 2.7 », indique la version du moteur d'exécution requis par l'application.

3 Ajoutez l'élément `<id>` :

<id>examples.html.HelloWorld</id> L'ID identifie l'application de manière unique, en conjonction avec l'ID d'éditeur (dérivé par AIR du certificat de signature du package d'application). L'ID d'application assure l'installation, l'accès au répertoire de stockage du système de fichiers de l'application privé, l'accès aux emplacements de stockage chiffrés privés et la communication entre les applications.

4 Ajoutez l'élément `<versionNumber>` :

<versionNumber>0.1</versionNumber> Permet à l'utilisateur de déterminer la version de l'application qu'il installe.

Remarque : *si vous utilisez AIR 2 ou une version antérieure, vous devez utiliser l'élément `<version>` et non l'élément `<versionNumber>`.*

5 Ajoutez l'élément `<filename>` :

<filename>HelloWorld</filename> Nom utilisé pour le fichier exécutable et le répertoire d'installation de l'application, ainsi que pour les autres références à cette dernière dans le système d'exploitation.

6 Ajoutez l'élément `<initialWindow>`, qui contient les éléments enfants suivants, afin de spécifier les propriétés de la fenêtre initiale de l'application :

<content>HelloWorld.html</content> Identifie le fichier HTML racine que doit charger AIR.

<visible>true</visible> Indique que la fenêtre est immédiatement visible.

<width>400</width> Définit la largeur de la fenêtre (en pixels).

<height>200</height> Définit la hauteur de la fenêtre.

7 Enregistrez le fichier. Le fichier descripteur d'application terminé se présente comme suit :

```
<?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>
```

Cet exemple se contente de définir quelques-unes des propriétés d'application disponibles. Voir « Fichiers descripteurs d'applications AIR » à la page 217 pour consulter la liste complète des propriétés de l'application, qui permettent de définir des paramètres tels que le chrome, la taille et la transparence des fenêtres, le répertoire d'installation par défaut, les types de fichier associés et les icônes de l'application.

Création de la page HTML de l'application

Vous devez à présent créer une page HTML simple qui constituera le fichier principal de l'application AIR.

1 Ouvrez le fichier `HelloWorld.html` en vue de le modifier. Ajoutez le code HTML suivant :

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

2 Dans la section `<head>` du code HTML, importez le fichier `AIRAliases.js` :

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

AIR définit une propriété appelée `runtime` sur l'objet de fenêtre HTML. Cette propriété permet d'accéder aux classes AIR intégrées, à l'aide du nom de package complet de la classe. Pour créer un objet File AIR, par exemple, vous pourriez ajouter l'instruction suivante en JavaScript :

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

Le fichier `AIRAliases.js` définit des alias pratiques pour les API AIR les plus utiles. Grâce à lui, vous pouvez par exemple raccourcir la référence à la classe File comme suit :

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

3 Sous la balise script d'AIRAliases, ajoutez une autre balise script contenant une fonction JavaScript pour gérer l'événement `onLoad` :

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

La fonction `appLoad()` appelle simplement la fonction `air.trace()`. Lorsque vous exécutez l'application à l'aide d'ADL, le message de trace s'imprime sur la console de commande. Les instructions de trace peuvent être très utiles pour le débogage.

4 Enregistrez le fichier.

Le fichier `HelloWorld.html` doit maintenant se présenter comme suit :

```
<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>
```

Test de l'application

Pour exécuter et tester l'application à partir de la ligne de commande, faites appel à l'application de débogage du lanceur AIR (ADL). Le fichier exécutable ADL se trouve dans le répertoire `bin` du kit SDK AIR. Si vous n'avez pas encore installé le kit SDK AIR, voir « Installation du kit de développement SDK AIR » à la page 17.

1 Ouvrez une console ou un shell de commande. Passez dans le répertoire créé pour ce projet.

2 Exécutez la commande suivante :

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

Une fenêtre AIR contenant l'application s'affiche. La fenêtre de la console présente également le message résultant de l'appel `air.trace()`.

Pour plus d'informations, voir « Fichiers descripteurs d'applications AIR » à la page 217.

Création du fichier d'installation AIR

Lorsque l'application s'exécute correctement, vous pouvez, à l'aide de l'outil ADT, la mettre en package sous forme de fichier d'installation AIR, c-à-d. un fichier d'archives contenant tous les fichiers de l'application, que vous pouvez distribuer à vos utilisateurs. Pour pouvoir installer un fichier AIR mis en package, vous devez installer Adobe AIR.

Pour garantir la sécurité des applications, tous les fichiers d'installation AIR doivent être signés numériquement. A des fins de développement, vous pouvez générer un certificat auto-signé de base à l'aide de l'outil ADT ou d'un autre outil de génération de certificats. Libre à vous également d'acheter un certificat de signature de code commercial auprès d'une autorité de certification telle que VeriSign ou Thawte. Lorsque les utilisateurs installent un fichier AIR auto-signé, l'éditeur porte la mention « Inconnu » pendant le processus d'installation. En effet, un certificat auto-signé garantit uniquement que le fichier AIR n'a pas été modifié depuis sa création. Rien n'empêche une tierce personne de présenter un faux fichier AIR auto-signé comme étant votre application. Il est fortement recommandé de signer les fichiers AIR que vous publiez au moyen d'un certificat commercial vérifiable. Pour une présentation de la sécurité AIR, voir Sécurité AIR (développeurs ActionScript) ou Sécurité AIR (développeurs HTML).

Génération d'un certificat auto-signé et d'une paire de clés

❖ A l'invite de commande, entrez la commande suivante (le fichier exécutable ADT se trouve dans le répertoire `bin` du kit SDK AIR) :

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

L'outil ADT génère le fichier de magasin de clés *sampleCert.pfx*, qui contient un certificat et la clé privée correspondante.

Cet exemple utilise le nombre minimal d'attributs qu'il est possible de définir pour un certificat. Le type de clé doit correspondre à *1024-RSA* ou *2048-RSA* (voir « Signature d'applications AIR » à la page 200).

Création du fichier d'installation AIR

❖ A l'invite de commande, entrez la commande suivante (sur une même ligne) :

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

Vous êtes invité à entrer le mot de passe du fichier de magasin de clés.

L'argument HelloWorld.air correspond au fichier AIR généré par l'outil ADT. HelloWorld-app.xml est le fichier descripteur d'application. Les autres arguments représentent les fichiers utilisés par l'application. Cet exemple n'utilise que deux fichiers, mais vous pouvez inclure tout nombre de fichiers et de répertoires. L'outil ADT vérifie que le fichier de contenu principal, HelloWorld.html, figure dans le package, mais si vous omettez le fichier AIRAliases.js, l'application ne fonctionne pas.

Une fois le package AIR créé, vous pouvez double-cliquer dessus pour installer et exécuter l'application. Libre à vous également d'entrer le nom de fichier AIR en tant que commande dans une fenêtre de shell ou de commande.

Etapes suivantes

Dans AIR, le code HTML et JavaScript se comporte généralement comme s'il se trouvait dans un navigateur Web ordinaire. (De fait, AIR utilise le même moteur de rendu WebKit que le navigateur Web Safari.) Cependant, vous devez connaître quelques différences notoires lorsque vous développez des applications HTML dans AIR. Pour plus d'informations sur ces différences et d'autres sujets importants, voir Programming HTML and JavaScript (disponible en anglais uniquement).

Création d'une première application de bureau AIR à l'aide du kit SDK de Flex

A titre d'illustration rapide et pratique du fonctionnement d'Adobe® AIR®, suivez ces instructions pour créer une application AIR simple basée sur SWF, « Hello World », par le biais du kit SDK Flex. Ce didacticiel illustre la compilation, le test et la mise en package d'une application AIR à l'aide des outils de ligne de commande intégrés au kit SDK de Flex (ce dernier contient le kit SDK d'AIR).

Avant de commencer, vous devez installer le moteur d'exécution et configurer Adobe® Flex™. Ce didacticiel fait appel au compilateur *AMXMLC*, à l'*application de débogage du lanceur AIR* (ADL) et à l'*outil AIR Developer* (ADT). Ces programmes résident dans le répertoire `bin` du kit SDK Flex (voir « Configuration du kit SDK Flex » à la page 19).

Création du fichier descripteur d'application AIR

Cette section est consacrée à la création du fichier descripteur d'application, à savoir un fichier XML dont la structure est la suivante :

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

1 Créez un fichier XML appelé `HelloWorld-app.xml` et enregistrez-le dans le répertoire du projet.

2 Ajoutez l'élément `<application>` avec pour attribut l'espace de noms AIR :

<application xmlns="http://ns.adobe.com/air/application/2.7"> Le dernier segment de l'espace de noms, « 2.7 », indique la version du moteur d'exécution requis par l'application.

3 Ajoutez l'élément `<id>` :

<id>samples.flex.HelloWorld</id> L'ID identifie l'application de manière unique, en conjonction avec l'ID d'éditeur (dérivé par AIR du certificat de signature du package d'application). Le format recommandé correspond à une chaîne de style DNS inversé dont les éléments sont séparés par un point, telle que « `com.company.AppName` ». L'ID d'application assure l'installation, l'accès au répertoire de stockage du système de fichiers de l'application privé, l'accès aux emplacements de stockage chiffrés privés et la communication entre les applications.

4 Ajoutez l'élément `<versionNumber>` :

<versionNumber>1.0</versionNumber> Permet à l'utilisateur de déterminer la version de l'application qu'il installe.

Remarque : *si vous utilisez AIR 2 ou une version antérieure, vous devez utiliser l'élément `<version>` et non l'élément `<versionNumber>`.*

5 Ajoutez l'élément `<filename>` :

<filename>HelloWorld</filename> Nom du fichier exécutable et du répertoire d'installation de l'application, ainsi que des autres références à cette dernière dans le système d'exploitation.

6 Ajoutez l'élément `<initialWindow>`, qui contient les éléments enfants suivants, afin de spécifier les propriétés de la fenêtre initiale de l'application :

<content>HelloWorld.swf</content> Identifie le fichier SWF racine que doit charger AIR.

<visible>true</visible> Indique que la fenêtre est immédiatement visible.

<width>400</width> Définit la largeur de la fenêtre (en pixels).

<height>200</height> Définit la hauteur de la fenêtre.

7 Enregistrez le fichier. Le format du fichier descripteur d'application complet est le suivant :

```
<?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>
```

Cet exemple se contente de définir quelques-unes des propriétés d'application disponibles. Voir « Fichiers descripteurs d'applications AIR » à la page 217 pour consulter la liste complète des propriétés de l'application, qui permettent de définir des paramètres tels que le chrome, la taille et la transparence des fenêtres, le répertoire d'installation par défaut, les types de fichier associés et les icônes de l'application.

Programmation du code de l'application

Remarque : *les applications AIR basées sur SWF peuvent utiliser une classe principale définie avec MXML ou Adobe® ActionScript® 3.0. Cet exemple définit sa classe principale par le biais d'un fichier MXML. Le processus de création d'une application AIR avec une classe ActionScript principale est similaire. Au lieu de compiler un fichier MXML pour obtenir le fichier SWF, vous compilez le fichier de classe ActionScript. Si vous utilisez ActionScript, la classe principale doit étendre flash.display.Sprite.*

A l'instar de toutes les applications Flex, les applications AIR basées sur la structure Flex contiennent un fichier MXML principal. Les applications de bureau AIR font appel au composant *WindowedApplication* en tant qu'élément racine au lieu du composant Application. Le composant WindowedApplication fournit les propriétés, les méthodes et les événements de contrôle de l'application et de sa fenêtre initiale. Pour les plates-formes et profils pour lesquels AIR ne prend pas en charge les fenêtres multiples, continuez à utiliser le composant Application. Dans les applications mobiles Flex, vous disposez également des composants View ou TabbedViewNavigatorApplication.

La procédure suivante permet de créer l'application Hello World :

1 Créez un fichier appelé `HelloWorld.mxml` dans un éditeur de texte et ajoutez le code MXML suivant :

```
<?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 Ajoutez ensuite un composant Label à l'application (placez-le dans la balise WindowedApplication).

3 Définissez la propriété `text` du composant Label sur `« Hello AIR »`.

4 Définissez les contraintes de mise en forme de sorte qu'il reste centré.

L'exemple suivant illustre le code dans son état actuel :

```
<?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>
```

Compilation de l'application

Avant d'exécuter et de déboguer l'application, compilez le code MXML dans un fichier SWF par le biais du compilateur amxmlc. Le compilateur amxmlc réside dans le répertoire `bin` du kit SDK Flex. Le cas échéant, vous pouvez définir la variable d'environnement path de l'ordinateur de sorte à inclure le répertoire bin du kit SDK Flex. Définir cette variable d'environnement simplifie l'exécution des utilitaires de ligne de commande.

1 Ouvrez une interface de commande ou un terminal et accédez au dossier de projet de l'application AIR.

2 Entrez la commande suivante :

```
amxmlc HelloWorld.mxml
```

L'exécution de `amxmlc` produit `HelloWorld.swf`, qui contient le code compilé de l'application.

Remarque : *s'il est impossible de compiler l'application, corrigez les fautes d'orthographe ou les erreurs de syntaxe. Les erreurs et les avertissements sont affichés dans la fenêtre de console d'exécution du compilateur amxmlc.*

Pour plus d'informations, voir « Compilation de fichiers sources MXML et ActionScript pour AIR » à la page 164.

Test de l'application

Pour exécuter et tester l'application à partir de la ligne de commande, faites appel à l'application de débogage du lanceur AIR (ADL) afin de lancer l'application par le biais du fichier descripteur correspondant. (L'application ADL réside dans le répertoire bin du kit SDK Flex.)

❖ A l'invite de commande, entrez la commande suivante :

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

L'application AIR résultante ressemble à l'illustration suivante :

Grâce aux propriétés horizontalCenter et verticalCenter du contrôle Label, le texte est placé au centre de la fenêtre. Vous pouvez déplacer ou redimensionner la fenêtre comme pour toute autre application de bureau.

Pour plus d'informations, voir « Application de débogage du lanceur AIR (ADL) » à la page 168.

Création du fichier d'installation AIR

Lorsque l'application s'exécute correctement, vous pouvez, à l'aide de l'outil ADT, la mettre en package sous forme de fichier d'installation AIR, c-à-d. un fichier d'archives contenant tous les fichiers de l'application, que vous pouvez distribuer à vos utilisateurs. Pour pouvoir installer un fichier AIR mis en package, vous devez installer Adobe AIR.

Pour garantir la sécurité des applications, tous les fichiers d'installation AIR doivent être signés numériquement. A des fins de développement, vous pouvez générer un certificat auto-signé de base à l'aide de l'outil ADT ou d'un autre outil de génération de certificats. Vous pouvez également acheter un certificat développeur auprès d'une autorité de certification commerciale. Lorsque les utilisateurs installent un fichier AIR auto-signé, l'éditeur porte la mention « Inconnu » pendant le processus d'installation. En effet, un certificat auto-signé garantit uniquement que le fichier AIR n'a pas été modifié depuis sa création. Rien n'empêche une tierce personne de présenter un faux fichier AIR auto-signé comme étant votre application. Il est fortement recommandé de signer les fichiers AIR que vous publiez au moyen d'un certificat commercial vérifiable. Pour une présentation de la sécurité AIR, voir Sécurité AIR (développeurs ActionScript) ou Sécurité AIR (développeurs HTML).

Génération d'un certificat auto-signé et d'une paire de clés

❖ A l'invite de commande, entrez la commande suivante (le fichier exécutable ADT réside dans le répertoire `bin` du kit SDK Flex) :

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

Cet exemple utilise le nombre minimal d'attributs qu'il est possible de définir pour un certificat. Le type de clé doit correspondre à *1024-RSA* ou *2048-RSA* (voir « Signature d'applications AIR » à la page 200).

Création d'un package AIR

❖ A l'invite de commande, entrez la commande suivante (sur une même ligne) :

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

Vous êtes invité à entrer le mot de passe du fichier de magasin de clés. Saisissez le mot de passe et appuyez sur Entrée. Les caractères saisis ne sont pas affichés par mesure de sécurité.

L'argument HelloWorld.air correspond au fichier AIR généré par l'outil ADT. HelloWorld-app.xml est le fichier descripteur d'application. Les autres arguments représentent les fichiers utilisés par l'application. Cet exemple n'utilise que trois fichiers, mais vous pouvez inclure un nombre illimité de fichiers et de répertoires.

Une fois le package AIR créé, vous pouvez double-cliquer dessus pour installer et exécuter l'application. Libre à vous également d'entrer le nom de fichier AIR en tant que commande dans une fenêtre de shell ou de commande.

Pour plus d'informations, voir « Mise en package d'un fichier d'installation AIR de bureau » à la page 55.

Création d'une première application AIR for Android à l'aide du kit SDK de Flex

Installez et configurez au préalable les kits SDK d'AIR et de Flex. Ce didacticiel fait appel au compilateur *AMXMLC* du kit SDK de Flex, ainsi qu'à l'*application de débogage du lanceur AIR* (ADL) et à l'*outil AIR Developer* (ADT) du kit SDK d'AIR. Voir « Configuration du kit SDK Flex » à la page 19.

Vous devez également télécharger et installer le kit SDK d'Android à partir du site Web d'Android comme indiqué dans : Android Developers : Installing the SDK (disponible en anglais uniquement).

Remarque : *pour plus d'informations sur le développement d'applications iPhone, voir Création d'une application iPhone Hello World dans Flash Professional CS5.*

Création du fichier descripteur d'application AIR

Cette section est consacrée à la création du fichier descripteur d'application, à savoir un fichier XML dont la structure est la suivante :

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

1 Créez un fichier XML appelé `HelloWorld-app.xml` et enregistrez-le dans le répertoire du projet.

2 Ajoutez l'élément `<application>` avec pour attribut l'espace de noms AIR :

<application xmlns="http://ns.adobe.com/air/application/2.7"> Le dernier segment de l'espace de noms, « 2.7 », indique la version du moteur d'exécution requis par l'application.

3 Ajoutez l'élément `<id>` :

<id>samples.android.HelloWorld</id> L'ID identifie l'application de manière unique, en conjonction avec l'ID d'éditeur (extrait par AIR du certificat de signature du package d'application). Le format recommandé est une chaîne de type DNS inversé, dont les valeurs sont séparées par un point, telle que « `com.société.NomApp` ».

4 Ajoutez l'élément `<versionNumber>` :

<versionNumber>0.0.1</versionNumber> Permet à l'utilisateur de déterminer la version de l'application qu'il installe.

5 Ajoutez l'élément `<filename>` :

<filename>HelloWorld</filename> Nom du fichier exécutable et du répertoire d'installation de l'application, ainsi que des autres références à cette dernière dans le système d'exploitation.

6 Ajoutez l'élément `<initialWindow>`, qui contient les éléments enfants suivants, afin de spécifier les propriétés de la fenêtre initiale de l'application :

<content>HelloWorld.swf</content> Identifie le fichier HTML racine que doit charger AIR.

7 Ajoutez l'élément `<supportedProfiles>`.

<supportedProfiles>mobileDevice</supportedProfiles> Indique que l'application s'exécute dans le profil mobile uniquement.

8 Enregistrez le fichier. Le format du fichier descripteur d'application complet est le suivant :

```
<?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>
```

Cet exemple se contente de définir quelques-unes des propriétés d'application disponibles. Vous disposez d'autres paramètres dans le fichier descripteur de l'application. Vous pouvez, par exemple, ajouter <fullScreen>true</fullScreen> à l'élément initialWindow pour créer une application en plein écran. Pour activer le débogage à distance et les fonctionnalités à accès contrôlé sous Android, vous devez également ajouter des autorisations Android sur le fichier descripteur de l'application. Etant donné que cette application simple ne requiert aucune autorisation, il est inutile d'en ajouter maintenant.

Pour plus d'informations, voir « Définition des propriétés d'une application mobile » à la page 74.

Programmation du code de l'application

Créez le fichier HelloWorld.as et ajoutez le code suivant à l'aide d'un éditeur de texte :

```
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 );
        }
    }
}
```

Compilation de l'application

Avant d'exécuter et de déboguer l'application, compilez le code MXML dans un fichier SWF par le biais du compilateur amxmlc. Le compilateur amxmlc réside dans le répertoire `bin` du kit SDK Flex. Le cas échéant, vous pouvez définir la variable d'environnement path de l'ordinateur de sorte à inclure le répertoire bin du kit SDK Flex. Définir cette variable d'environnement simplifie l'exécution des utilitaires de ligne de commande.

1. Ouvrez une interface de commande ou un terminal et accédez au dossier de projet de l'application AIR.
2. Entrez la commande suivante :

```
amxmlc HelloWorld.as
```

L'exécution de `amxmlc` produit `HelloWorld.swf`, qui contient le code compilé de l'application.

Remarque : *s'il est impossible de compiler l'application, corrigez les fautes d'orthographe ou les erreurs de syntaxe. Les erreurs et les avertissements sont affichés dans la fenêtre de console d'exécution du compilateur amxmlc.*

Pour plus d'informations, voir « Compilation de fichiers sources MXML et ActionScript pour AIR » à la page 164.

Test de l'application

Pour exécuter et tester l'application à partir de la ligne de commande, faites appel à l'application de débogage du lanceur AIR (ADL) afin de lancer l'application par le biais du fichier descripteur correspondant. (L'application ADL réside dans le répertoire bin des kits SDK d'AIR et de Flex.)

❖ A l'invite de commande, entrez la commande suivante :

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

Pour plus d'informations, voir « Simulation de périphérique à l'aide de l'application ADL » à la page 106.

Création du fichier de package APK

Lorsque l'application s'exécute correctement, vous pouvez la mettre en package sous forme de fichier APK à l'aide de l'outil ADT. Un fichier de package APK correspond au format de fichier d'application Android natif, que vous pouvez distribuer aux utilisateurs.

Toutes les applications Android doivent être signées. A l'encontre des fichiers AIR, une application Android est généralement signée à l'aide d'un certificat auto-signé. Le système d'exploitation Android ne tente pas d'identifier le développeur de l'application. Vous pouvez signer un package Android à l'aide d'un certificat généré par l'outil ADT. Les certificats associés aux applications destinées au marché Android doivent être valides pendant 25 ans au moins.

Génération d'un certificat auto-signé et d'une paire de clés

❖ A l'invite de commande, entrez la commande suivante (le fichier exécutable ADT réside dans le répertoire `bin` du kit SDK Flex) :

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

Cet exemple utilise le nombre minimal d'attributs qu'il est possible de définir pour un certificat. Le type de clé doit être soit *1024-RSA*, soit *2048-RSA* (voir « Commande ADT certificate » à la page 183).

Création d'un package AIR

❖ A l'invite de commande, entrez la commande suivante (sur une même ligne) :

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

Vous êtes invité à entrer le mot de passe du fichier de magasin de clés. Saisissez le mot de passe et appuyez sur Entrée.

Pour plus d'informations, voir « Mise en package d'une application AIR mobile » à la page 98.

Installation du moteur d'exécution d'AIR

Vous pouvez installer la version la plus récente du moteur d'exécution d'AIR sur le périphérique à partir d'Android Market. Vous pouvez également installer le moteur d'exécution intégré au kit SDK sur un périphérique ou un émulateur Android.

❖ A l'invite de commande, entrez la commande suivante (sur une même ligne) :

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

Définissez l'indicateur `-platformsdk` sur le répertoire qui stocke le kit SDK d'Android (indiquez le parent du dossier tools).

L'outil ADT installe le fichier Runtime.apk intégré au kit SDK.

Pour plus d'informations, voir « Installation du moteur d'exécution et des applications AIR à des fins de développement » à la page 115.

Installation de l'application AIR

❖ A l'invite de commande, entrez la commande suivante (sur une même ligne) :

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

Définissez l'indicateur `-platformsdk` sur le répertoire qui stocke le kit SDK d'Android (indiquez le parent du dossier tools).

Pour plus d'informations, voir « Installation du moteur d'exécution et des applications AIR à des fins de développement » à la page 115.

Vous pouvez lancer l'application en touchant l'icône correspondante sur l'écran du périphérique ou de l'émulateur.

Chapitre 6 : Développement d'applications de bureau AIR

Flux de travail de développement d'une application de bureau AIR

Le flux de travail de base de développement d'une application AIR s'apparente à la plupart des modèles de développement : code, compilation, test, puis, en fin de cycle, mise en package dans un fichier de programme d'installation.

Vous pouvez programmer le code de l'application en Flash, Flex, ou ActionScript et le compiler dans Flash Professional, Flash Builder ou les compilateurs de ligne de commande mxmlc et compc. Vous pouvez également programmer le code de l'application en HTML et JavaScript et sauter l'étape de compilation.

Vous pouvez tester des applications de bureau AIR par le biais de l'application ADL, qui exécute une application sans mise en package et installation préalables. Flash Professional, Flash Builder, Dreamweaver et l'IDE Aptana s'intègrent tous au débogueur Flash. Vous pouvez également lancer manuellement l'outil de débogage, FDB, lorsque vous utilisez l'application ADL à partir de la ligne de commande. L'application ADL affiche quant à elle des messages d'erreur et des instructions trace.

Toute application AIR doit être mise en package dans un fichier d'installation. Le format du fichier AIR multiplateformes est recommandé, sauf dans les cas suivants :

- Vous devez accéder à des API qui dépendent de la plate-forme, telles que la classe NativeProcess.
- Votre application fait appel à des extensions natives.

Dans ce cas de figure, vous pouvez mettre en package une application AIR en tant que fichier de programme d'installation natif propre à une plate-forme donnée.

Applications de type SWF

1. Programmez le code MXML ou ActionScript.
2. Créez les actifs requis, tels que les fichiers de bitmap d'icône.
3. Créez le fichier descripteur de l'application.
4. Compilez le code ActionScript.
5. Testez l'application.
6. Mettez en package l'application et signez-la en tant que fichier AIR à l'aide de la cible *air*.

Applications de type HTML

1. Programmez le code HTML et JavaScript.
2. Créez les actifs requis, tels que les fichiers de bitmap d'icône.
3. Créez le fichier descripteur de l'application.
4. Testez l'application.

5 Mettez en package l'application et signez-la en tant que fichier AIR à l'aide de la cible *air*.

Création de programmes d'installation natifs destinés aux applications AIR

1 Programmez le code (ActionScript ou HTML et JavaScript).

2 Créez les actifs requis, tels que les fichiers de bitmap d'icône.

3 Créez le descripteur de l'application en spécifiant le profil *extendedDesktop*.

4 Compilez tout code ActionScript.

5 Testez l'application.

6 Mettez en package l'application sur chaque plate-forme cible à l'aide de la cible *native*.

Remarque : *le programme d'installation natif correspondant à une plate-forme cible doit être créé sur cette plate-forme. Vous ne pouvez pas, par exemple, créer un programme d'installation Windows sur un ordinateur Mac. Vous pouvez utiliser une machine virtuelle, telle que VMWare, pour exécuter plusieurs plates-formes sur le même ordinateur.*

Création d'applications AIR à l'aide d'un paquet de moteur d'exécution captif

1 Programmez le code (ActionScript ou HTML et JavaScript).

2 Créez les actifs requis, tels que les fichiers de bitmap d'icône.

3 Créez le descripteur de l'application en spécifiant le profil *extendedDesktop*.

4 Compilez tout code ActionScript.

5 Testez l'application.

6 Mettez en package l'application sur chaque plate-forme cible à l'aide de la cible *bundle*.

7 Créez un programme d'installation à l'aide des fichiers du paquet. (Le kit SDK d'AIR ne fournit aucun outil pour créer un tel programme d'installation, mais de nombreux kits d'outils tiers sont disponibles.)

Remarque : *le paquet correspondant à une plate-forme cible doit être créé sur cette plate-forme. Vous ne pouvez pas, par exemple, créer un paquet Windows sur un ordinateur Mac. Vous pouvez utiliser une machine virtuelle, telle que VMWare, pour exécuter plusieurs plates-formes sur le même ordinateur.*

Définition des propriétés d'une application de bureau

Définissez les propriétés de base de l'application dans le fichier descripteur correspondant. Cette section est consacrée aux propriétés des applications de bureau AIR. Les éléments du fichier descripteur d'application font l'objet d'une description détaillée dans « Fichiers descripteurs d'applications AIR » à la page 217.

Version du moteur d'exécution d'AIR requise

Indiquez la version du moteur d'exécution d'AIR requise par l'application à l'aide de l'espace de noms du fichier descripteur de l'application.

Assigné dans l'élément `application`, l'espace de noms détermine globalement les fonctionnalités dont dispose l'application. Ainsi, si l'application utilise l'espace de noms d'AIR 1.5 et qu'AIR 3.0 est installé sur l'ordinateur ou le périphérique de l'utilisateur, elle assimile le comportement d'AIR à celui de la version 1.5, même s'il a été modifié dans AIR 3.0. Pour que l'application accède au nouveau comportement et aux nouvelles fonctionnalités, vous devez au préalable modifier l'espace de noms et publier une mise à jour. Les modifications associées à la sécurité et à WebKit constituent les principales exceptions à la règle.

Spécifiez l'espace de noms à l'aide de l'attribut xmlns de l'élément racine `application` :

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

Voir aussi

« application » à la page 222

Identité d'une application

Plusieurs paramètres devraient être propres à chaque application publiée, à savoir l'ID, le nom et le nom de fichier.

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

Voir aussi

« id » à la page 239

« filename » à la page 234

« name » à la page 247

Version d'une application

Dans les versions d'AIR antérieures à 2.5, spécifiez l'application dans l'élément `version`. Vous pouvez utiliser n'importe quelle chaîne. Le moteur d'exécution d'AIR n'interprète pas la chaîne. En d'autres termes, « 2.0 » n'est pas considéré comme une version ultérieure à « 1.0 ».

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

Dans AIR 2.5 et ultérieur, spécifiez la version de l'application dans l'élément `versionNumber`. L'élément `version` est à présent obsolète. Si vous définissez une valeur dans `versionNumber`, vous devez utiliser une séquence de trois nombres au plus, séparés par un point (« 0.1.2 », par exemple). Chaque segment du numéro de la version se compose de trois chiffres au plus. (En d'autres termes, la version la plus longue autorisée correspond à « 999.999.999 ».) Vous ne devez pas obligatoirement inclure trois segments dans le nombre. Les numéros de version « 1 » et « 1.0 » sont également valides.

Vous pouvez aussi définir le libellé de la version à l'aide de l'élément `versionLabel`. Si vous ajoutez un libellé de version, il remplace le numéro de version dans les boîtes de dialogue du programme d'installation de l'application AIR, par exemple.

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

Voir aussi

« version » à la page 255

« versionLabel » à la page 255

« versionNumber » à la page 256

Propriétés de la fenêtre principale

Lorsqu'AIR démarre une application à partir du bureau, il crée une fenêtre et charge dans celle-ci la page HTML ou le fichier SWF principal. AIR fait appel aux éléments enfants de l'élément `initialWindow` pour contrôler l'aspect et le comportement initiaux de la fenêtre initiale de l'application.

- **content** : fichier SWF principal de l'application dans l'élément enfant `content` de l'élément `initalWindow`. Si vous ciblez des périphériques dans le profil de bureau, vous pouvez utiliser un fichier SWF ou HTML.

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

  Vous devez inclure ce fichier dans le package AIR (à l'aide de l'outil ADT ou de l'IDE). Se contenter de faire référence au nom dans le descripteur d'application n'entraîne pas l'inclusion automatique du fichier dans le package.

- **depthAndStencil** : indique s'il est nécessaire d'utiliser le tampon de profondeur ou le tampon de modèle. Ces tampons s'utilisent normalement avec du contenu 3D.

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

- **height** : hauteur de la fenêtre initiale.
- **maximizable** : indique si le chrome système d'agrandissement de la fenêtre est affiché.
- **maxSize** : taille maximale autorisée.
- **minimizable** : indique si le chrome système de réduction de la fenêtre en icône est affiché.
- **minSize** : taille minimale autorisée.
- **renderMode** : dans AIR 3 ou les versions ultérieures, le mode de rendu peut être défini sur *auto*, *cpu*, *direct* ou *gpu* pour les applications de bureau. Dans les versions antérieures d'AIR, ce paramètre est ignoré sur les plates-formes de bureau. Il est impossible de modifier le paramètre renderMode au moment de l'exécution.
  - auto : pratiquement identique au mode cpu.
  - cpu : les objets d'affichage sont rendus et copiés pour afficher la mémoire dans le logiciel. StageVideo est disponible uniquement lorsqu'une fenêtre est en mode plein écran. Stage3D fait appel au moteur de rendu du logiciel.
  - direct : les objets d'affichage sont rendus par le logiciel du moteur d'exécution, mais la copie de l'image rendue pour afficher la mémoire (blitting) est effectuée par accélération matérielle. StageVideo est disponible. Stage3D fait appel à l'accélération matérielle, lorsque cela est possible. Si la transparence de la fenêtre est définie sur true, la fenêtre revient au rendu logiciel et au blitting.

    Remarque : *pour tirer profit de l'accélération par processeur graphique du contenu Flash sur les plates-formes AIR mobiles, Adobe recommande d'utiliser renderMode="direct" (c'est-à-dire, Stage3D) plutôt que renderMode="gpu". Adobe prend officiellement en charge les structures d'application basées sur Stage3D suivantes et recommande leur utilisation : Starling (2D) et Away3D (3D). Pour plus d'informations sur Stage3D et Starling/Away3D, voir http://gaming.adobe.com/getstarted/.*
  - gpu : l'accélération matérielle est utilisée, le cas échéant.

- **requestedDisplayResolution** : indique si l'application doit utiliser le mode de résolution *standard* ou *high* (élevé) sur les ordinateurs MacBook Pro dotés d'écrans haute résolution. Sur toutes les autres plates-formes, cette valeur est ignorée. Si sa valeur est *standard*, chaque pixel de la scène est rendu par quatre pixels à l'écran. Si la valeur est *high*, chaque pixel correspond à un seul pixel physique à l'écran. La valeur spécifiée est utilisée pour toutes les fenêtres d'application. L'utilisation de l'élément `requestedDisplayResolution` pour les applications de bureau AIR (en tant qu'enfant de l'élément `intialWindow`) est disponible dans AIR 3.6 et ultérieur.
- **resizable** : indique si le chrome système de redimensionnement de la fenêtre est affiché.
- **systemChrome** : indique si les caractéristiques standard du système d'exploitation sont utilisées. Il est impossible de modifier le paramètre systemChrome d'une fenêtre à l'exécution.
- **title** : titre de la fenêtre.
- **transparent** : indique si la fenêtre est semi-transparente par rapport à l'arrière-plan. Si la transparence est activée, la fenêtre ne peut pas utiliser le chrome système. Il est impossible de modifier le paramètre transparent d'une fenêtre à l'exécution.
- **visible** : indique si la fenêtre est visible dès sa création. La fenêtre est initialement invisible par défaut, afin que l'application puisse dessiner son contenu avant d'activer elle-même sa visibilité.
- **width** : largeur de la fenêtre.
- **x** : position horizontale de la fenêtre.
- **y** : position verticale de la fenêtre.

Voir aussi

« content » à la page 227

« depthAndStencil » à la page 229

« height » à la page 238

« maximizable » à la page 245

« maxSize » à la page 246

« minimizable » à la page 246

« minimizable » à la page 246

« minSize » à la page 246

« renderMode » à la page 249

« requestedDisplayResolution » à la page 250

« resizable » à la page 251

« systemChrome » à la page 253

« title » à la page 254

« transparent » à la page 255

« visible » à la page 256

« width » à la page 257

« x » à la page 257

« y » à la page 257

Fonctionnalités associées au bureau

Les éléments suivants contrôlent les fonctionnalités d'installation sur le bureau et de mise à jour.

- customUpdateUI : permet d'utiliser vos propres boîtes de dialogue lors de la mise à jour d'une application. Si cet élément est défini sur `false` (valeur par défaut), les boîtes de dialogue AIR standard sont utilisées.
- fileTypes : indique les types de fichiers ouverts par défaut dans l'application. Si une autre application est déjà associée par défaut à un type de fichier, AIR ne remplace pas l'enregistrement existant. L'application peut toutefois remplacer l'enregistrement à l'exécution par le biais de la méthode `setAsDefaultApplication()` de l'objet NativeApplication. Il est recommandé de demander à l'utilisateur s'il autorise le remplacement d'associations de types de fichiers existantes.

  Remarque : *l'enregistrement du type de fichier est ignoré lors de la mise en package d'une application en tant que paquet de moteur d'exécution captif (à l'aide de la cible `-bundle`). Pour enregistrer un type de fichier donné, vous devez créer un programme d'installation qui effectue lui-même l'enregistrement.*
- installFolder : indique un chemin relatif au dossier d'installation standard de l'application. Ce paramètre permet de définir un nom de dossier personnalisé, ainsi que de grouper plusieurs applications dans un dossier commun.
- programMenuFolder : indique l'arborescence de menus associée au menu Tous les programmes de Windows. Ce paramètre permet de grouper plusieurs applications au sein d'un menu commun. Si vous ne spécifiez pas de dossier de menus, le raccourci associé à l'application est directement ajouté au menu principal.

Voir aussi

« customUpdateUI » à la page 229

« fileTypes » à la page 235

« installFolder » à la page 242

« programMenuFolder » à la page 248

Profils pris en charge

Si l'application ne peut s'exécuter que sur le bureau, vous pouvez interdire son installation sur des périphériques issus d'un autre profil en excluant celui-ci de la liste des profils pris en charge. Si l'application utilise la classe NativeProcess ou des extensions natives, vous devez prendre en charge le profil `extendedDesktop`.

Si l'élément `supportedProfile` ne figure pas dans le descripteur d'application, il est considéré comme acquis que l'application prend en charge tous les profils définis. Pour restreindre l'application à une liste déterminée de profils, recensez ces derniers en séparant chaque profil par un espace blanc :

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

Pour consulter la liste des classes ActionScript prises en charge par les profils `desktop` et `extendedDesktop`, voir « Fonctionnalités des différents profils » à la page 260.

Voir aussi

« supportedProfiles » à la page 252

Extensions natives requises

Les applications qui prennent en charge le profil `extendedDesktop` peuvent recourir aux extensions natives.

Déclarez toutes les extensions natives auxquelles fait appel l'application AIR dans le descripteur de l'application. L'exemple suivant illustre la syntaxe permettant de spécifier deux extensions natives requises :

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

La valeur de l'élément `extensionID` est identique à celle de l'élément `id` dans le fichier descripteur de l'extension. Le fichier descripteur de l'extension est un fichier XML appelé extension.xml. Il est mis en package dans le fichier ANE fourni par le développeur de l'extension native.

Icônes d'une application

Les icônes spécifiées dans le descripteur d'application sont associées sur le bureau au fichier d'application, au raccourci et au menu du programme. L'icône de l'application doit être fournie sous la forme d'un ensemble d'images PNG de 16x16, 32x32, 48x48 et 128x128 pixels. Spécifiez le chemin d'accès aux fichiers d'icône dans l'élément icon du fichier descripteur de l'application :

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

Si vous ne fournissez pas d'icône de la taille indiquée, la taille suivante est utilisée et l'image est mise à l'échelle en conséquence. Si vous ne fournissez pas d'icône, une icône système par défaut est utilisée.

Voir aussi

« icon » à la page 238

« imageNxN » à la page 239

Paramètres ignorés

Les applications installées sur le bureau ignorent les paramètres d'application qui s'appliquent aux fonctionnalités d'un profil mobile. Les paramètres ignorés sont les suivants :

- android
- aspectRatio
- autoOrients
- fullScreen
- iPhone
- renderMode (versions antérieures à AIR 3)
- requestedDisplayResolution
- softKeyboardBehavior

Débogage d'une application de bureau AIR

Si vous développez une application par le biais d'un IDE tel que Flash Builder, Flash Professional ou Dreamweaver, les outils de débogage sont généralement intégrés. Pour déboguer une application, il suffit de la lancer en mode de débogage. Si vous n'utilisez pas d'IDE qui prend en charge le débogage direct, vous disposez de l'application de débogage du lanceur AIR (ADL) et du programme Flash Debugger (FDB) pour déboguer l'application.

Voir aussi

De Monsters: Monster Debugger (disponible en anglais uniquement)

« Débogage à l'aide de l'outil AIR HTML Introspector » à la page 298

Exécution d'une application à l'aide de l'application ADL

L'utilisation d'ADL permet d'exécuter une application AIR sans la mettre en package et l'installer. Transmettez le fichier descripteur de l'application à l'application ADL sous forme de paramètre, comme illustré par l'exemple suivant (compilez au préalable le code ActionScript de l'application) :

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

ADL imprime les instructions trace, les exceptions d'exécution et les erreurs d'analyse HTML dans la fenêtre du terminal. Si un processus FDB attend une connexion entrante, l'application ADL se connecte au débogueur.

Vous pouvez également utiliser ADL pour déboguer une application AIR faisant appel à des extensions natives. Exemple :

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

Voir aussi

« Application de débogage du lanceur AIR (ADL) » à la page 168

Impression d'instructions trace

Pour imprimer des instructions trace sur la console utilisée pour exécuter ADL, ajoutez des instructions trace au code à l'aide de la fonction `trace()` :

Remarque : *si vos instructions* `trace()` *ne s'affichent pas sur la console, assurez-vous de ne pas avoir spécifié* `ErrorReportingEnable` *ou* `TraceOutputFileEnable` *dans le fichier mm.cfg. Pour plus d'informations sur l'emplacement propre à la plate-forme de ce fichier, voir Modification du fichier mm.cfg.*

Exemple ActionScript :

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

Exemple JavaScript :

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

Dans le code JavaScript, vous pouvez utiliser les fonctions `alert()` et `confirm()` afin d'afficher les messages de débogage provenant de l'application. En outre, les numéros de ligne correspondant aux erreurs de syntaxe, de même que toutes les exceptions JavaScript non interceptées sont imprimées sur la console.

Remarque : *pour pouvoir utiliser le préfixe air illustré dans l'exemple JavaScript, vous devez importer le fichier AIRAliases.js dans la page. Ce fichier réside dans le répertoire frameworks du kit SDK d'AIR.*

Connexion au programme Flash Debugger (FDB)

Pour déboguer une application AIR à l'aide de Flash Debugger, ouvrez une session FDB, puis lancez l'application à l'aide d'ADL.

Remarque : *dans les applications AIR de type SWF, il est nécessaire de compiler les fichiers sources ActionScript à l'aide de l'indicateur* `-debug`*. (Dans Flash Professional, cochez l'option Autoriser le débogage dans la boîte de dialogue Paramètres de publication.)*

1. Lancez le programme FDB. Le programme FDB est disponible dans le répertoire `bin` du kit SDK Flex.

   La console affiche l'invite de FDB : `<fdb>`

2. Exécutez la commande `run` : `<fdb>run [Entrée]`.

3. Dans une console de shell ou de commande différente, lancez une version de débogage de l'application :

   ```
   adl myApp.xml
   ```

4. A l'aide des commandes de FDB, définissez les points d'arrêt souhaités.

5. Saisissez : `continue [Entrée]`

Si l'application AIR est de type SWF, le débogueur ne contrôle que l'exécution du code ActionScript. Si l'application AIR est de type HTML, le débogueur ne contrôle que l'exécution du code JavaScript.

Pour exécuter ADL sans vous connecter au débogueur, ajoutez l'option `-nodebug` :

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

Pour obtenir des informations de base sur les commandes FDB, exécutez la commande `help` :

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

Pour plus d'informations sur les commandes FDB, voir Utilisation des commandes du débogueur de ligne de commande dans la documentation Flex.

Mise en package d'un fichier d'installation AIR de bureau

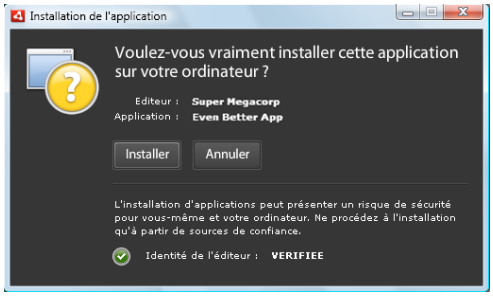
Chaque application AIR doit disposer d'au moins un fichier descripteur d'application et d'un fichier SWF ou HTML principal. Il est également impératif de mettre en package dans le fichier AIR tout autre actif à installer avec l'application.

Cet article est consacré à la mise en package d'une application AIR à l'aide des outils de ligne de commande intégrés au kit SDK. Pour plus d'informations sur la mise en package d'une application par le biais de l'un des outils de création d'Adobe, voir :

- Adobe® Flex® Builder™, voir Packaging AIR applications with Flex Builder (disponible en anglais uniquement)
- Adobe® Flash® Builder™, voir Création de packages d'application AIR avec Flash Builder
- Adobe® Flash® Professional, voir Publication pour Adobe AIR
- Adobe® Dreamweaver®, voir Création d'une application AIR dans Dreamweaver

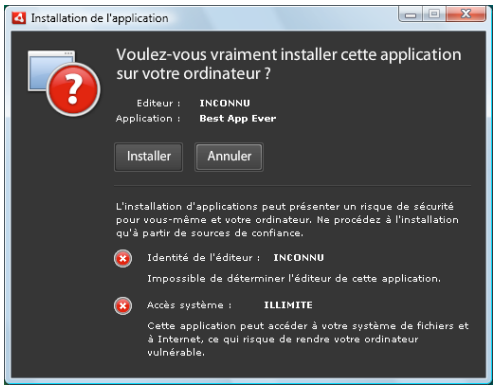
Tous les fichiers du programme d'installation AIR doivent être signés à l'aide d'un certificat numérique. Le programme d'installation AIR fait appel à la signature pour vérifier que le fichier de l'application n'a pas été modifié depuis que vous y avez apposé votre signature. Vous pouvez utiliser un certificat de signature de code provenant d'une autorité de certification ou un certificat auto-signé.

Un certificat émis par une autorité de certification approuvée offre aux utilisateurs de l'application une certaine garantie de votre identité en tant qu'éditeur. La boîte de dialogue d'installation reflète le fait que votre identité est vérifiée par une autorité de certification :



Boîte de dialogue de confirmation de l'installation d'une application signée par un certificat approuvé

Lorsque vous utilisez un certificat auto-signé, les utilisateurs n'ont aucun moyen de vérifier que vous êtes véritablement le signataire. Par ailleurs, un tel certificat ne garantit en aucune façon que le package n'a pas été modifié (il se peut en effet qu'un faux soit substitué au fichier d'installation légitime avant que l'utilisateur ne l'ait en sa possession). La boîte de dialogue d'installation reflète le fait que l'identité de l'éditeur ne peut être avérée. Les risques sécuritaires liés à l'installation de l'application sont donc plus importants :



Boîte de dialogue de confirmation de l'installation d'une application signée par un certificat auto-signé

Vous pouvez créer et signer un package de fichier AIR au cours de la même opération grâce à la commande `-package` d'ADT. Vous avez également la possibilité de créer un package non signé intermédiaire à l'aide de la commande `-prepare`. Vous pouvez ensuite signer ce package intermédiaire au cours d'une étape distincte en utilisant la commande `-sign`.

Remarque : *la version 1.5 et les versions ultérieures de Java ne prennent pas en charge les caractères ASCII étendus dans les mots de passe de protection des fichiers de certificats PKCS12. Lorsque vous créez ou exportez le fichier de certificat de signature du code, utilisez uniquement des caractères ASCII ordinaires dans le mot de passe.*

Lors de la signature du package d'installation, l'outil ADT contacte automatiquement un serveur d'autorité d'horodatage chargé de vérifier l'heure. Les informations d'horodatage sont incluses dans le fichier AIR. Un fichier AIR comprenant un horodatage vérifié peut être installé à tout moment par la suite. Si l'outil ADT ne parvient pas à se connecter au serveur d'horodatage, la création du package est annulée. Vous pouvez ignorer l'option d'horodatage, mais sans horodatage, il devient impossible d'installer une application AIR une fois que le certificat utilisé pour signer le fichier d'installation est arrivé à expiration.

Vous devez utiliser le certificat d'origine pour signer un package créé pour mettre à jour une application AIR existante. Si vous avez renouvelé ce certificat, qu'il a expiré au cours des 180 derniers jours ou que vous souhaitez le remplacer, vous pouvez apposer une signature de migration. Dans ce cas, le fichier AIR est signé à l'aide des deux certificats, l'ancien et le nouveau. La commande `-migrate` permet d'appliquer la signature de migration, comme indiqué à la section « Commande ADT migrate » à la page 182.

Important : *vous disposez d'un délai de 180 jours au maximum pour apposer la signature de migration une fois le certificat d'origine périmé. Sans cette signature, les utilisateurs existants doivent désinstaller leur application existante pour pouvoir installer la nouvelle version. Le délai ne s'applique qu'aux applications pour lesquelles l'espace de noms stipule AIR 1.5.3 ou une version ultérieure dans le fichier descripteur. Lorsque vous ciblez des versions antérieures du moteur d'exécution d'AIR, vous ne disposez d'aucun délai.*

Les signatures de migration n'étaient pas prises en charge préalablement à AIR 1.1. Pour apposer une signature de migration, vous devez mettre en package une application à l'aide d'un kit de développement SDK version 1.1 ou ultérieure.

Les applications déployées à l'aide de fichiers AIR font partie du profil de bureau. Il est impossible d'utiliser l'outil ADT pour mettre en package un programme d'installation natif destiné à une application AIR si le fichier descripteur correspondant ne prend pas en charge le profil de bureau. Vous pouvez restreindre l'utilisation de ce profil à l'aide de l'élément `supportedProfiles` dans le fichier descripteur de l'application. Voir « Profils de périphérique » à la page 259 et « supportedProfiles » à la page 252.

Remarque : *les paramètres du fichier descripteur d'application déterminent l'identité d'une application AIR et son chemin d'installation par défaut. Voir « Fichiers descripteurs d'applications AIR » à la page 217.*

Identifiants d'éditeur

Depuis AIR 1.5.3, les identifiants d'éditeur sont obsolètes. Les nouvelles applications (publiées à l'origine avec AIR 1.5.3 ou version ultérieure) ne requièrent pas d'identifiant d'éditeur et ne doivent pas en spécifier.

Lors de la mise à jour d'applications publiées à l'aide des versions antérieures d'AIR, vous devez spécifier l'identifiant d'éditeur d'origine dans le fichier descripteur d'application, sans quoi la version installée de l'application et la version mise à jour sont traitées comme des applications différentes. Si vous utilisez un autre identifiant d'éditeur ou omettez la balise publisherID, l'utilisateur est contraint de désinstaller la version antérieure pour pouvoir installer la nouvelle.

Pour déterminer l'identifiant d'éditeur original, recherchez le fichier `publisherid` dans le sous-répertoire META-INF/AIR du répertoire d'installation de l'application originale. La chaîne que contient ce fichier correspond à l'identifiant d'éditeur. Le fichier descripteur de l'application doit stipuler le moteur d'exécution d'AIR 1.5.3 (ou version ultérieure) dans la déclaration d'espace de noms pour que vous puissiez stipuler manuellement l'identifiant d'éditeur.

Dans le cas des applications publiées avant AIR 1.5.3 (ou de celles publiées à l'aide du kit SDK AIR 1.5.3, mais dont l'espace de noms du descripteur spécifie une version antérieure d'AIR), un identifiant d'éditeur est calculé à partir du certificat de signature. Allié à l'identifiant d'application, l'identifiant d'éditeur permet de déterminer l'identité d'une application. S'il existe, l'identifiant d'éditeur est utilisé comme suit :

- Pour vérifier que le fichier AIR est une mise à jour et non une nouvelle application à installer
- Dans la clé de chiffrement destinée au magasin local chiffré
- Dans le chemin du répertoire de stockage d'application
- Dans la chaîne de connexion associée aux connexions locales
- Dans la chaîne d'identité destinée à appeler une application par le biais de l'API intégrée au navigateur d'AIR
- Dans l'OSID (définition d'interface de service ouverte) utilisée lors de la création de programmes d'installation/de désinstallation personnalisés

Préalablement à AIR 1.5.3, l'identifiant d'éditeur d'une application pouvait changer si vous aviez signé la mise à jour de celle-ci par le biais d'une signature de migration utilisant un certificat nouveau ou renouvelé. Toute modification de l'identifiant d'éditeur entraîne un changement de comportement de toute fonctionnalité AIR basée sur celui-ci. Il devient, par exemple, impossible d'accéder aux données stockées dans le magasin local chiffré et la chaîne de connexion associée à toute occurrence de Flash ou d'AIR qui crée une connexion locale à l'application doit contenir le nouvel identifiant.

Dans AIR 1.5.3 ou ultérieur, l'identifiant d'éditeur n'est pas fondé sur le certificat de signature et est uniquement affecté si la balise publisherID figure dans le descripteur d'application. Il est impossible de mettre à jour une application si l'identifiant d'éditeur spécifié pour le package AIR de mise à jour ne correspond pas à l'identifiant d'éditeur actif.

Mise en package avec l'outil ADT

L'outil de ligne de commande ADT d'AIR permet de mettre en package une application AIR. Avant la mise en package, il est nécessaire de compiler tout code ActionScript, MXML et code d'extension éventuel. Vous devez également disposer d'un certificat de signature du code.

Pour consulter des informations de référence détaillées sur les commandes et options de l'outil ADT, voir « Outil AIR Developer (ADT) » à la page 174.

Création d'un package AIR

Pour créer un package AIR, utilisez la commande ADT package et définissez le type de cible sur *air* pour les versions validées.

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

L'exemple part du principe que le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Pour plus d'informations, voir « Variables d'environnement path » à la page 321.)

Vous devez exécuter la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf et un répertoire d'icônes.

Si vous exécutez la commande comme indiqué, l'outil ADT vous invite à entrer le mot de passe associé au keystore. (Les caractères du mot de passe saisis ne sont pas toujours affichés. Appuyez simplement sur Entrée une fois la saisie terminée.)

Création d'un package AIR à partir d'un fichier AIRI

Créer et signer un fichier AIRI permet de créer un package AIR à installer :

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

Mise en package d'un programme d'installation natif de bureau

Depuis AIR 2, vous pouvez créer, à l'aide de l'outil ADT, des programmes d'installation d'application natifs destinés à distribuer des applications AIR. Vous pouvez, par exemple, créer un fichier d'installation EXE pour distribuer une application AIR sous Windows, un fichier d'installation DMG pour distribuer une application AIR sur Mac OS et, dans AIR 2.5 et AIR 2.6, un fichier d'installation DEB ou RPM pour distribuer une application AIR sous Linux.

Les applications installées par le biais d'un programme d'installation natif font partie du profil de bureau étendu. Il est impossible d'utiliser l'outil ADT pour mettre en package un programme d'installation natif destiné à une application AIR si le fichier descripteur correspondant ne prend pas en charge le profil étendu de bureau. Vous pouvez restreindre l'utilisation de ce profil à l'aide de l'élément `supportedProfiles` dans le fichier descripteur de l'application. Voir « Profils de périphérique » à la page 259 et « supportedProfiles » à la page 252.

Vous disposez de deux techniques de base pour créer une version du programme d'installation natif de l'application AIR :

- Vous pouvez créer le programme d'installation natif à partir du fichier descripteur de l'application et d'autres fichiers sources (à savoir, des fichiers SWF, des fichiers HTML et d'autres actifs).
- Vous pouvez fonder le programme d'installation natif sur un fichier AIR ou un fichier AIRI.

Vous devez utiliser l'outil ADT sur le même système d'exploitation que celui du programme d'installation à créer. Par conséquent, pour créer un fichier EXE pour Windows, exécutez l'outil ADT sous Windows. Pour créer un fichier DMG pour Mac OS, exécutez l'outil ADT sur Mac OS. Pour créer un fichier DEB ou RPG pour Linux, exécutez l'outil ADT à partir du kit SDK d'AIR 2.6 sous Linux.

Lorsque vous créez un programme d'installation natif pour distribuer une application AIR, celle-ci possède les fonctionnalités suivantes :

- Elle peut lancer des processus natifs et communiquer avec eux, par le biais de la classe NativeProcess. Pour plus d'informations, voir :
  - Communication avec les processus natifs dans AIR (développeurs ActionScript)
  - Communication avec les processus natifs dans AIR (développeurs HTML)
- Elle peut faire appel à des extensions natives.
- Elle peut ouvrir tout fichier dans l'application système définie par défaut à cet effet à l'aide de la méthode `File.openWithDefaultApplication()`, quel que soit le type du fichier. (Les applications qui *ne sont pas* installées à l'aide d'un programme d'installation natif sont soumises à des restrictions. Pour plus d'informations, voir l'entrée `File.openWithDefaultApplication()` dans la référence du langage.)

La mise en package sous forme de programme d'installation natif entraîne toutefois la perte de certains avantages qui caractérisent le format de fichier AIR. Il est alors impossible de distribuer un fichier unique sur tous les ordinateurs de bureau. La fonction de mise à jour intégrée (ainsi que la structure du programme de mise à jour) ne fonctionnent pas.

Lorsque l'utilisateur double-clique sur le fichier du programme d'installation natif, l'application AIR est installée. Si la version requise d'Adobe AIR n'est pas déjà installée sur la machine, le programme d'installation la télécharge du réseau et l'installe. En l'absence d'une connexion réseau permettant d'obtenir la version appropriée d'Adobe AIR (si nécessaire), l'installation échoue. L'installation échoue également si le système d'exploitation n'est pas pris en charge dans Adobe AIR 2.

Remarque : *si vous souhaitez qu'un fichier puisse être exécuté dans l'application installée, assurez-vous qu'il est exécutable sur le système de fichiers lors de la mise en package de l'application. (Sous Mac et Linux, vous disposez de chmod pour définir l'indicateur exécutable, le cas échéant.)*

Création d'un programme d'installation natif à partir des fichiers sources de l'application

Pour créer un programme d'installation natif à partir des fichiers sources de l'application, utilisez la commande `-package` en respectant la syntaxe suivante (sur une même ligne de commande) :

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

La syntaxe est identique à celle permettant de mettre en package un fichier AIR (sans programme d'installation natif), à quelques exceptions près :

- Vous ajoutez l'option `-target native` à la commande. (Si vous spécifiez `-target air`, l'outil ADT génère un fichier AIR plutôt qu'un fichier d'installation natif.)
- Vous spécifiez le fichier DMG ou EXE cible en tant que fichier de programme d'installation (`installer_file`).
- Sous Windows, vous pouvez ajouter un deuxième jeu d'options de signature, indiquées sous la forme `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` dans la syntaxe. Cette étape est facultative. Sous Windows, outre la signature du fichier AIR, vous pouvez signer le fichier d'installation. Utilisez la même syntaxe pour définir le type de certificat et l'option de signature que pour signer le fichier AIR (voir « Options de signature du code de l'outil ADT » à la page 189). Vous pouvez signer le fichier AIR et le fichier d'installation à l'aide d'un même certificat ou de certificats différents. Lorsque l'utilisateur télécharge un fichier d'installation Windows signé du Web, Windows se fonde sur le certificat pour identifier la source du fichier.

Pour plus d'informations sur les options ADT autres que `-target`, voir « Outil AIR Developer (ADT) » à la page 174.

L'exemple suivant crée un fichier DMG (fichier d'installation natif réservé à Mac OS) :

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

L'exemple suivant crée un fichier EXE (fichier d'installation natif réservé à Windows) :

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

L'exemple suivant crée un fichier EXE et le signe :

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

Création d'un programme d'installation natif pour une application faisant appel à des extensions natives

Vous pouvez créer un programme d'installation natif à partir des fichiers sources de l'application et des packages d'extensions natives dont l'application a besoin. Utilisez la commande `-package` en respectant la syntaxe suivante (sur une seule ligne de commande) :

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

Cette syntaxe est identique à celle utilisée pour la mise en package d'un programme d'installation natif, avec deux options supplémentaires. Utilisez l'option `-extdir` *`extension-directory`* en vue de spécifier le répertoire contenant les fichiers ANE (extensions natives) qu'utilise l'application. Utilisez l'indicateur facultatif `-migrate` avec les paramètres `MIGRATION_SIGNING_OPTIONS` en vue de signer la mise à jour d'une application avec une signature de migration lorsque le certificat principal de signature de code est différent de celui utilisé par la version précédente. Pour plus d'informations, voir « Signature d'une version mise à jour d'une application AIR » à la page 211.

Pour plus d'informations sur les options d'ADT, voir « Outil AIR Developer (ADT) » à la page 174.

L'exemple suivant crée un fichier DMG (fichier de programme d'installation pour Mac OS) pour une application faisant appel à des extensions natives :

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

Création d'un programme d'installation natif à partir d'un fichier AIR ou AIRI

L'outil ADT permet de générer un fichier d'installation natif à partir d'un fichier AIR ou AIRI. Pour créer un fichier d'installation natif à partir d'un fichier AIR, utilisez la commande `-package` de l'outil ADT en respectant la syntaxe suivante (sur une même ligne de commande) :

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

Cette syntaxe est identique à celle permettant de créer un fichier d'installation natif à partir des fichiers sources de l'application AIR, à quelques exceptions près :

- Comme source, vous spécifiez un fichier AIR, et non un fichier descripteur d'application et d'autres fichiers sources de l'application AIR.
- Ne spécifiez pas d'options de signature du fichier AIR car il est déjà signé.

Pour créer un fichier d'installation natif à partir d'un fichier *AIRI*, utilisez la commande `-package` de l'outil ADT en respectant la syntaxe suivante (sur une même ligne de commande) :

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

Cette syntaxe est identique à celle permettant de créer un fichier d'installation natif à partir d'un fichier AIR, à quelques exceptions près :

- Vous spécifiez un fichier AIRI comme source.
- Vous spécifiez des options de signature pour l'application AIR cible.

L'exemple suivant crée un fichier DMG (fichier d'installation natif sur Mac OS) à partir d'un fichier AIR :

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

L'exemple suivant crée un fichier EXE (fichier d'installation natif sous Windows) à partir d'un fichier AIR :

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

L'exemple suivant crée un fichier EXE (à partir d'un fichier AIR) et le signe :

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

L'exemple suivant crée un fichier DMG (fichier d'installation natif sur Mac OS) à partir d'un fichier AIRI :

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

L'exemple suivant crée un fichier EXE (fichier d'installation natif sous Windows) à partir d'un fichier AIRI :

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

L'exemple suivant crée un fichier EXE (basé sur un fichier AIRI) et applique une signature AIR et une signature Windows native :

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

Mise en package d'un paquet de moteur d'exécution captif pour des ordinateurs de bureau

Un paquet de moteur d'exécution captif est un package qui inclut le code de votre application, ainsi qu'une version dédiée du moteur d'exécution. Une application mise en package de cette manière fait appel au moteur d'exécution du paquet et non au moteur d'exécution partagé installé sur l'ordinateur d'un utilisateur.

Le paquet produit est un dossier autonome de fichiers d'application sur Windows et un paquet .app sur Mac OS. Vous devez produire le paquet correspondant à un système d'exploitation cible lors de l'exécution de ce système d'exploitation. (Il est possible d'utiliser une machine virtuelle, telle que VMWare, pour exécuter plusieurs systèmes d'exploitation sur un même ordinateur.)

Vous pouvez exécuter l'application à partir de ce dossier ou paquet sans installation.

Avantages

- Création d'une application autonome
- Aucun accès Internet requis pour l'installation

- Indépendance de l'application par rapport aux mises à jour du moteur d'exécution
- Possibilité aux entreprises de certifier l'application spécifique et l'association de moteurs d'exécution
- Prise en charge du modèle traditionnel de déploiement de logiciels
- Aucune redistribution individuelle du moteur d'exécution requise
- Possibilité d'utiliser l'API NativeProcess
- Possibilité d'utiliser des extensions natives
- Possibilité d'utiliser la fonction `File.openWithDefaultApplication()` sans aucune restriction
- Possibilité d'exécution à partir d'un périphérique USB ou d'un disque optique sans aucune installation

Inconvénients

- Principaux correctifs de sécurité publiés par Adobe non automatiquement disponibles aux utilisateurs
- Impossibilité d'utiliser le format de fichier .air
- Création nécessaire d'un programme d'installation propre, le cas échéant
- API et structure de mise à jour AIR non prises en charge
- API propre au navigateur AIR permettant d'installer et de lancer une application AIR à partir d'une page Web non prise en charge
- Sous Windows, l'enregistrement du fichier doit être géré par le programme d'installation de l'utilisateur
- Plus grande empreinte disque de l'application

Création d'un paquet de moteur d'exécution captif sous Windows

Pour créer un paquet de moteur d'exécution captif sous Windows, vous devez mettre en package l'application lors de l'exécution du système d'exploitation Windows. Mettez en package l'application à l'aide de la cible *bundle* de l'outil ADT :

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

Cette commande crée le paquet dans un répertoire appelé myApp. Ce répertoire contient les fichiers pour votre application, ainsi que les fichiers du moteur d'exécution. Vous pouvez exécuter le programme directement à partir du dossier. Néanmoins, pour créer une entrée de menu de programme, ou enregistrer les types de fichiers ou les gestionnaires de modèles d'URI, vous devez créer un programme d'installation qui définit les entrées de registre requises. Le kit SDK d'AIR n'inclut aucun outil permettant de créer ces programmes d'installation, mais diverses options tierces sont disponibles, notamment des kits d'outils de programmes d'installation Open Source gratuits.

Vous pouvez signer l'exécutable natif sous Windows en spécifiant un deuxième ensemble d'options de signature après l'entrée `-target bundle` sur la ligne de commande. Ces options de signature identifient la clé privée et le certificat associé à utiliser lors de l'application de la signature Windows native. (Il est possible d'utiliser un certificat de signature de code AIR.) Seul l'exécutable principal est signé. Tout exécutable supplémentaire mis en package avec votre application n'est pas signé par ce processus.

Association de types de fichiers

Pour associer votre application à des types de fichiers publics ou personnalisés sous Windows, votre programme d'installation doit définir les entrées de registre appropriées. Les types de fichiers doivent également être répertoriés dans l'élément fileTypes du fichier descripteur de l'application.

Pour plus d'informations sur les types de fichiers Windows, voir MSDN Library: File Types and File Associations (disponible en anglais uniquement).

Enregistrement du gestionnaire d'URI

Afin que votre application puisse prendre en charge le lancement d'une URL via un modèle d'URI donné, votre programme d'installation doit définir les entrées de registre requises.

Pour plus d'informations sur l'enregistrement d'une application en vue de gérer un modèle d'URI, voir MSDN Library: Registering an Application to a URL Protocol (disponible en anglais uniquement).

Création d'un paquet de moteur d'exécution captif sous Mac OS X

Pour créer un paquet de moteur d'exécution captif sous Mac OS X, vous devez mettre en package l'application lors de l'exécution du système d'exploitation Macintosh. Mettez en package l'application à l'aide de la cible *bundle* de l'outil ADT :

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

Cette commande crée le paquet d'application appelé myApp.app. Ce paquet contient les fichiers pour votre application, ainsi que les fichiers du moteur d'exécution. Vous pouvez exécuter l'application en cliquant deux fois sur l'icône myApp.app, puis l'installer en la faisant glisser vers l'emplacement approprié, par exemple vers le dossier Applications. Néanmoins, pour enregistrer les types de fichiers ou les gestionnaires de modèles d'URI, vous devez modifier le fichier de liste de propriétés à l'intérieur du package d'application.

Pour la distribution, vous pouvez créer un fichier d'image de disque (.dmg). Le kit SDK d'Adobe AIR ne fournit aucun outil permettant de créer un fichier dmg pour un paquet de moteur d'exécution captif.

Association de types de fichiers

Pour associer votre application à des types de fichiers publics ou personnalisés sous Mac OS X, vous devez modifier le fichier info.plist dans le paquet afin de définir la propriété CFBundleDocumentTypes. Voir Mac OS X Developer Library: Information Property List Key Reference, CFBundleURLTypes (disponible en anglais uniquement).

Enregistrement du gestionnaire d'URI

Afin que votre application puisse prendre en charge le lancement d'une URL via un modèle d'URI donné, vous devez modifier le fichier info.plist file dans le paquet en vue de définir la propriété CFBundleURLTypes. Voir Mac OS X Developer Library: Information Property List Key Reference, CFBundleDocumentTypes (disponible en anglais uniquement).

Distribution d'un package AIR pour ordinateur de bureau

Vous pouvez distribuer une application AIR sous forme de package AIR contenant le code de l'application et tous ses actifs. Vous pouvez distribuer ce package via les méthodes standard : téléchargement, courrier électronique ou support physique comme le CD-ROM. Les utilisateurs peuvent installer l'application en double-cliquant sur le fichier AIR. L'API d'AIR intégrée au navigateur (bibliothèque ActionScript Web) permet aux utilisateurs d'installer l'application AIR (ainsi qu'Adobe® AIR®, le cas échéant) par le biais d'un lien unique sur une page Web.

Il est également possible de mettre en package et distribuer une application AIR sous forme de programme d'installation natif (en d'autres termes, fichier EXE sous Windows, fichier DMG sous Mac et fichier DEB ou RPM sous Linux). Vous pouvez distribuer et installer les packages d'installation natifs conformément aux conventions appropriées de la plate-forme. En distribuant l'application sous forme de package natif, vous perdez toutefois certains avantages du format de fichier AIR. Il devient ainsi impossible de faire appel à la structure de mise à jour d'AIR et à l'API intégrée au navigateur ou d'utiliser un fichier d'installation unique sur la plupart des plates-formes.

Installation et exécution d'une application AIR à partir du bureau

Il vous suffit d'envoyer le fichier AIR au destinataire. Vous pouvez par exemple l'envoyer sous forme de pièce jointe dans un courrier électronique, ou sous forme de lien dans une page Web.

Une fois que l'utilisateur a téléchargé l'application AIR, il suit les instructions suivantes pour l'installer :

1 Double-cliquez sur le fichier AIR.

Le logiciel Adobe AIR doit être déjà installé sur l'ordinateur.

2 Dans la fenêtre d'installation, conservez les paramètres par défaut sélectionnés, puis cliquez sur Continuer.

Sous Windows, AIR effectue automatiquement les opérations suivantes :

- Installation de l'application dans le répertoire Program Files
- Création d'un raccourci sur le bureau pour ouvrir l'application
- Création d'un raccourci dans le menu Démarrer
- Ajout d'une entrée dans l'application Ajout/Suppression de programmes du Panneau de configuration

Sous Mac OS, l'application est automatiquement ajoutée dans le répertoire Applications.

Si l'application existe déjà, le programme d'installation donne à l'utilisateur le choix d'ouvrir la version existante de l'application ou de mettre celle-ci à jour grâce au fichier AIR téléchargé. Le programme d'installation identifie l'application au moyen de l'ID d'application et de l'ID d'éditeur qui sont contenus dans le fichier AIR.

3 Une fois que vous avez terminé l'installation, cliquez sur Terminer.

S'il utilise Mac OS, l'utilisateur a besoin des privilèges système appropriés pour installer une version de mise à jour d'une application dans le répertoire de l'application. S'il utilise Windows ou Linux, l'utilisateur doit disposer de privilèges d'administrateur.

Une application peut également installer une nouvelle version via ActionScript ou JavaScript. Pour plus d'informations, voir « Mise à jour des applications AIR » à la page 273.

Une fois l'application AIR installée, l'utilisateur se contente de double-cliquer sur l'icône de l'application pour l'exécuter, à l'instar de n'importe quelle application de bureau.

- Sous Windows, double-cliquez sur l'icône de l'application (installée sur le Bureau ou dans un dossier) ou sélectionnez l'application à partir du menu Démarrer.

- Sous Linux, double-cliquez sur l'icône de l'application (installée sur le Bureau ou dans un dossier) ou sélectionnez l'application dans le menu.
- Sous Mac OS, double-cliquez sur l'application dans le dossier où elle a été installée. Le répertoire d'installation par défaut est /Applications.

Remarque : *Seules les applications AIR développées pour AIR 2.6 ou les versions antérieures peuvent être installées sous Linux.*

La fonctionnalité d'*installation transparente* d'AIR permet à un utilisateur d'installer une application AIR en cliquant sur un lien dans une page Web. La fonctionnalité d'*appel du navigateur* d'AIR permet à un utilisateur d'exécuter une application AIR installée en cliquant sur un lien dans une page Web. Ces fonctionnalités sont décrites dans la section suivante.

Installation et exécution d'une application de bureau AIR à partir d'une page Web

L'API AIR intégrée au navigateur permet d'installer et d'exécuter une application AIR à partir d'une page Web. Elle réside dans une bibliothèque SWF, *air.swf*, qui est hébergée par Adobe. Le kit SDK d'AIR contient un exemple d'application « badge » qui utilise cette bibliothèque pour installer, mettre à jour ou lancer une application AIR (ainsi que le moteur d'exécution, le cas échéant). Vous pouvez modifier cet exemple d'application ou créer une application Web badge qui utilise directement la bibliothèque en ligne air.swf.

Il est possible d'installer toute application AIR par le biais d'un badge de page Web. Toutefois, seules les applications dont le fichier descripteur contient l'élément `<allowBrowserInvocation>true</allowBrowserInvocation>` peuvent être lancées par un badge Web.

Voir aussi

« API intégrée au navigateur et stockée dans le fichier AIR.SWF » à la page 264

Déploiement en entreprise sur des ordinateurs de bureau

Les administrateurs peuvent installer le moteur d'exécution d'Adobe AIR et les applications AIR en toute transparence à l'aide d'outils de déploiement de postes de travail standard. Ces administrateurs peuvent exécuter les tâches suivantes :

- Installation en mode silencieux du moteur d'exécution d'Adobe AIR à l'aide d'outils tels que Microsoft SMS, Tivoli d'IBM ou de tout autre outil de déploiement permettant des installations automatiques au moyen d'un programme d'amorçage
- Installation en mode silencieux de l'application AIR à l'aide des mêmes outils utilisés pour déployer le moteur d'exécution

Pour plus d'informations, voir le *Guide de l'administrateur Adobe AIR (http://www.adobe.com/go/learn_air_admin_guide_fr).*

Journaux d'installation sur des ordinateurs de bureau

Des journaux d'installation sont enregistrés lors de l'installation du moteur d'exécution d'AIR ou d'une application AIR. Vous pouvez les consulter pour déterminer la cause de tout problème d'installation ou de mise à jour.

Les fichiers journaux sont créés aux emplacements suivants :

- Mac : journal système standard (`/private/var/log/system.log`)

  Pour afficher le journal système Mac, ouvrez l'application Console (qui réside généralement dans le dossier Utilitaires).

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

Remarque : *ces fichiers n'étaient pas créés dans les versions d'AIR antérieures à AIR 2.*

Chapitre 7 : Développement d'applications AIR pour périphériques mobiles

Les applications AIR pour périphériques mobiles sont déployées en tant qu'applications natives. Elles utilisent le format applicatif du périphérique, plutôt que le format AIR. AIR prend actuellement en charge les packages APK d'Android et IPA d'iOS. Une fois la version commerciale du package de l'application créée, vous pouvez distribuer l'application par le biais du mécanisme standard de la plate-forme. Pour Android, il s'agit généralement d'Android Market, pour iOS, de l'App Store d'Apple.

Vous disposez du kit SDK d'AIR et de Flash Professional, Flash Builder ou d'un autre outil de développement ActionScript pour créer des applications AIR pour périphériques mobiles. Les applications AIR de type HTML ne sont actuellement pas prises en charge.

Remarque : *la tablette Research In Motion (RIM) BlackBerry Playbook fournit son propre kit SDK pour le développement d'AIR. Pour plus d'informations sur le développement de Playbook, voir RIM: BlackBerry Tablet OS Development (disponible en anglais uniquement).*

Remarque : *Ce document décrit la procédure de développement d'applications iOS par le biais du kit SDK d'AIR 2.6 ou d'une version ultérieure. Il est possible d'installer les applications créées avec AIR 2.6 et les versions ultérieures sur les dispositifs iPhone 3G, iPhone 4 et iPad exécutant iOS 4 ou une version ultérieure. Pour développer une application AIR destinée aux versions antérieures d'iOS, vous devez utiliser l'outil Packager for iPhone AIR 2, comme décrit dans le manuel Création d'applications iPhone.*

Pour plus d'informations sur les meilleures pratiques en matière de confidentialité, voir Guide de confidentialité du SDK Adobe AIR.

Pour plus d'informations sur la configuration système requise pour exécuter les applications AIR, voir Configuration requise.

Configuration de l'environnement de développement

Parallèlement à la configuration de l'environnement de développement AIR, Flex et Flash standard, il est nécessaire de configurer les plates-formes mobiles. (Pour plus d'informations sur la configuration de l'environnement de développement AIR de base, voir « Outils de la plate-forme Adobe Flash pour le développement AIR » à la page 17.)

Configuration d'Android

Aucune configuration spéciale n'est normalement requise pour Android dans AIR 2.6+. L'outil ADB d'Android est inclus dans le kit SDK d'AIR (dans le dossier lib/android/bin). Le kit SDK d'AIR fait appel à l'outil ADB pour installer, désinstaller et exécuter un package d'application sur un périphérique. L'outil ADB permet également d'afficher les journaux système. Pour créer et exécuter un émulateur Android, vous devez télécharger le kit SDK d'Android distinct.

Si votre application ajoute des éléments à l'élément `<manifestAdditions>` dans le descripteur d'application que la version actuelle d'AIR ne reconnaît pas comme valides, vous devez installer une version plus récente du SDK d'Android. Définissez la variable d'environnement AIR_ANDROID_SDK_HOME ou le paramètre de la ligne de commande `-platformsdk` sur le chemin de fichier du kit SDK. L'outil de mise en package d'AIR, ADT, fait appel à ce kit SDK pour valider les entrées dans l'élément `<manifestAdditions>`.

Dans AIR 2.5, vous devez télécharger un exemplaire distinct du kit SDK d'Android à partir de Google. Vous pouvez définir la variable d'environnement AIR_ANDROID_SDK_HOME de sorte à référencer le dossier du kit SDK d'Android. Si vous ne définissez pas cette variable d'environnement, vous devez spécifier le chemin d'accès au kit SDK d'Android dans l'argument `-platformsdk` sur la ligne de commande ADT.

Voir aussi

« Variables d'environnement ADT » à la page 198

« Variables d'environnement path » à la page 321

Configuration d'iOS

Pour installer et tester une application iOS sur un périphérique et la distribuer, vous devez appartenir au programme iOS Developer d'Apple (service payant). Une fois membre du programme iOS Developer, vous pouvez accéder au portail iOS Provisioning Portal, où vous pouvez obtenir d'Apple les éléments et fichiers suivants, dont vous devez disposer pour installer une application sur un périphérique à des fins de test et de distribution ultérieure :

- Certificats de développement et de distribution
- ID d'application
- Fichiers de configuration pour le développement et la distribution d'applications

Considérations liées à la conception d'applications mobiles

Le contexte d'exploitation et les caractéristiques physiques des périphériques mobiles nécessitent de porter une attention toute particulière au codage et à la conception. Il est, par exemple, déterminant de rationaliser le code pour assurer une exécution aussi rapide que possible. Il est bien évident que l'optimisation du code a ses limites. Une conception intelligente qui prend en compte les restrictions du périphérique permet également d'éviter que la présentation visuelle ne sollicite trop le système de rendu.

Code

Bien qu'une exécution plus rapide du code constitue toujours un avantage, la vitesse réduite du processeur de la plupart des périphériques mobiles rationalise l'investissement en temps que représente l'écriture de code minimaliste. Par ailleurs, les périphériques mobiles sont quasiment toujours alimentés par batterie. Obtenir un résultat identique en réduisant la puissance de traitement requise utilise moins de batterie.

Création

La taille d'écran réduite, le mode d'interaction par écran tactile, voire l'environnement en évolution constante d'un utilisateur mobile sont autant de facteurs à prendre en compte lors de la conception de l'expérience utilisateur de l'application.

Code et création conjoints

Si l'application fait appel à l'animation, l'optimisation du rendu joue un rôle déterminant. Dans certains cas, l'optimisation du code n'est toutefois pas suffisante. Vous devez concevoir les aspects visuels de l'application de sorte à assurer un rendu performant par le code.

D'importantes techniques d'optimisation sont passées en revue dans le manuel Optimisation de contenu mobile pour la plate-forme Adobe Flash. Les techniques décrites dans ce manuel s'appliquent à tout contenu Flash et AIR, mais jouent un rôle déterminant dans le développement d'applications pour périphériques mobiles performantes.

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

Cycle de vie d'une application

Lorsque la cible d'action passe d'une application à une autre, AIR fait passer la cadence à 4 images/s et désactive le rendu des graphiques. Une cadence inférieure se traduit souvent par l'interruption des connexions socket et du réseau de diffusion en continu. Si l'application n'utilise pas ces types de connexion, vous pouvez réduire plus encore la cadence.

Au moment opportun, vous devez arrêter la lecture de l'audio et supprimer les écouteurs associés aux capteurs de géolocalisation et de l'accéléromètre. L'objet NativeApplication AIR distribue des événements d'activation et de désactivation. Ces événements permettent de gérer la transition entre l'état actif et l'état d'arrière-plan.

La plupart des systèmes d'exploitation mobiles arrêtent les applications qui s'exécutent en arrière-plan sans avertissement. Grâce à l'enregistrement régulier de l'état d'une application, celle-ci devrait pouvoir restaurer elle-même un état satisfaisant, qu'il s'agisse de la réactivation de l'état actif à partir d'une exécution en arrière-plan ou d'un nouveau lancement.

Densité des informations

En dépit d'une densité (nombre de pixels par pouce) supérieure, la taille physique de l'écran des périphériques mobiles est inférieure à celle d'un ordinateur de bureau. Une même taille de police produit des lettres de taille physique inférieure sur un écran de périphérique mobile que sur un ordinateur de bureau. Par souci de lisibilité, il est souvent nécessaire d'utiliser une taille de police supérieure. En règle générale, 14 points est la plus petite taille de police qu'un utilisateur puisse facilement lire.

Les périphériques mobiles sont souvent utilisés en cours de déplacement et dans des environnements mal éclairés. Faites preuve de réalisme pour évaluer le volume d'informations qu'il est possible d'afficher sans perte de lisibilité. Cette valeur est probablement inférieure à celle d'un écran d'ordinateur de bureau aux dimensions identiques en pixels.

Gardez également à l'esprit le fait que lorsqu'un utilisateur touche l'écran, il masque du doigt ou de la main une partie de ce dernier. Placez les éléments interactifs sur les côtés et au bas de l'écran si l'utilisateur ne se contente pas de les toucher momentanément.

Saisie de texte

La plupart des périphériques gèrent la saisie de texte par le biais d'un clavier virtuel. Souvent peu pratiques à utiliser, les claviers virtuels masquent une partie de l'écran. A l'exception des touches programmables, évitez de faire appel aux événements de clavier.

Envisagez de substituer d'autres solutions aux champs de texte de saisie. Il est, par exemple, inutile d'inviter l'utilisateur à entrer une valeur numérique dans un champ de texte. Proposez deux boutons pour augmenter ou réduire la valeur.

Touches programmables

Les périphériques mobiles comportent un nombre divers de touches programmables, c'est-à-dire des boutons que vous pouvez programmer pour leur affecter différentes fonctions. Respectez les conventions de la plate-forme lorsque vous programmez ces touches dans l'application.

Changements d'orientation de l'écran

Un contenu mobile peut être visualisé en mode portrait ou paysage. Considérez comment l'application gère les changements d'orientation de l'écran. Pour plus d'informations, voir Orientation de la scène.

Baisse de la luminosité de l'écran

AIR ne désactive pas automatiquement la baisse de la luminosité de l'écran en cours de lecture vidéo. Vous disposez de la propriété `systemIdleMode` de l'objet NativeApplication AIR pour contrôler l'activation du mode d'économie d'énergie du périphérique. (Sur certaines plates-formes, vous devez demander les autorisations appropriées pour que cette fonctionnalité fonctionne.)

Appels téléphoniques entrants

Le moteur d'exécution d'AIR coupe automatiquement le son lorsque l'utilisateur effectue ou reçoit un appel téléphonique. Sous Android, définissez l'autorisation Android READ_PHONE_STATE dans le descripteur de l'application si celle-ci lit l'audio lorsqu'elle s'exécute en arrière-plan. Android interdit sinon au moteur d'exécution de détecter les appels téléphoniques et de couper le son automatiquement. Voir « Autorisations Android » à la page 80.

Cibles tactiles

Tenez compte de la taille des cibles tactiles lorsque vous concevez les boutons et autres éléments de l'interface utilisateur touchés par l'utilisateur. Assurez-vous que la taille de ces éléments est suffisamment élevée pour que l'utilisateur puisse facilement les activer du doigt sur un écran tactile. Veillez également à ménager un espace suffisant entre cibles tactiles. Pour un écran de téléphone à résolution élevée standard, la superficie d'une cible tactile doit mesurer environ 44 pixels sur 57 pixels.

Taille d'installation d'un package d'application

Les périphériques mobiles disposent généralement d'un espace de stockage considérablement plus réduit que les ordinateurs de bureau pour installer applications et données. Réduisez la taille d'un package en supprimant les actifs et bibliothèques inutilisés.

Sous Android, le package d'une application n'est pas extrait dans des fichiers distincts lors de l'installation de cette dernière. Les actifs sont décompressés dans un espace de stockage temporaire lorsqu'il est nécessaire d'y accéder. Pour réduire l'encombrement mémoire des actifs décompressés, fermez les flux d'URL et de fichier, une fois le chargement des actifs terminé.

Accès au système de fichiers

Les restrictions relatives au système de fichiers varient selon le système d'exploitation mobile. Elles diffèrent souvent des restrictions imposées par les systèmes d'exploitation de bureau. L'emplacement d'enregistrement des fichiers et données risque par conséquent de varier selon la plate-forme.

Les variations en matière de système de fichiers ont pour conséquence que les raccourcis associés aux répertoires communs fournis par la classe File d'AIR ne sont pas toujours disponibles. Le tableau suivant indique les raccourcis disponibles sous Android et iOS :

	Android	**iOS**
File.applicationDirectory	Lecture seule via une URL (et non le chemin natif)	Lecture seule
File.applicationStorageDirectory	Disponible	Disponible
File.cacheDirectory	Disponible	Disponible
File.desktopDirectory	Racine de la carte SD	Non disponible
File.documentsDirectory	Racine de la carte SD	Disponible
File.userDirectory	Racine de la carte SD	Non disponible
File.createTempDirectory()	Disponible	Disponible
File.createTempFile()	Disponible	Disponible

Les directives d'Apple concernant les applications iOS définissent des règles spécifiques concernant les emplacements de stockage à utiliser pour les fichiers dans différentes situations. Par exemple, l'une des directives est que seuls les fichiers contenant des données saisies par l'utilisateur ou des données qui ne peuvent pas être générées ou téléchargées à nouveau doivent être stockées dans le répertoire désigné pour la sauvegarde à distance. Pour plus d'informations sur la conformité aux directives d'Apple sur la sauvegarde des fichiers et la mise en cache, voir Contrôle de la sauvegarde et de la mise en cache des fichiers.

Composants de l'interface utilisateur

Adobe a développé une version de la structure d'application Flex optimisée pour les plates-formes mobiles. Pour plus d'informations, voir Développement d'applications mobiles avec Flex et Flash Builder.

Des projets de composants à usage communautaire adaptés aux applications mobiles sont également disponibles. Parmi ces API figurent :

- Josh Tynjala : Feathers UI controls for Starling (en anglais uniquement)
- Skinnable version of Minimal Comps (Derrick Grigg) (disponible en anglais uniquement)
- as3flobile components (Todd Anderson) (disponible en anglais uniquement)

Rendu des graphiques par accélération matérielle via l'API Stage3D

A partir d'AIR 3.2, AIR for Mobile prend en charge le rendu des graphiques par accélération matérielle via l'API Stage 3D. Les API ActionScript Stage3D sont un ensemble d'API à accélération matérielle par GPU permettant d'utiliser des fonctionnalités 2D et 3D avancées. Ces API de bas niveau permettent aux développeurs de tirer profit de l'accélération matérielle par GPU pour augmenter les performances de façon significative. Il est également possible d'utiliser des moteurs de jeu prenant en charge les API ActionScript Stage3D.

Pour plus d'informations, voir Moteurs de jeu, 3D et Stage 3D.

Lissage vidéo

Afin d'améliorer les performances, le lissage vidéo est désactivé sur AIR.

Fonctions natives

AIR 3.0+

De nombreuses plates-formes mobiles proposent des fonctions qui ne sont pas encore accessibles via l'API AIR standard. A partir d'AIR 3, vous pouvez étendre AIR avec vos propres bibliothèques de code natives. Ces bibliothèques d'extensions natives peuvent accéder aux fonctions disponibles sur le système d'exploitation, voire même aux fonctions propres à un périphérique donné. Il est possible d'écrire les extensions natives en langage C sur iOS, et en langage Java ou C sur Android. Pour plus d'informations sur le développement d'extensions natives, voir Présentation des extensions natives pour Adobe AIR.

Flux de travail de création d'une application AIR pour périphériques mobiles

Le flux de travail de création d'une application AIR pour périphériques mobiles (ou autres types de périphériques) s'apparente généralement au flux de travail de création d'applications de bureau. Les principales différences sont liées aux phases de mise en package, de débogage et d'installation d'une application. Les applications AIR for Android utilisent, par exemple, le format de package APK natif d'Android au lieu du format de package d'AIR. Par conséquent, elles font également appel aux mécanismes d'installation et de mise à jour standard d'Android.

AIR for Android

Les étapes ci-dessous décrivent une procédure de développement typique d'une application AIR for Android :

- Programmez le code en ActionScript ou MXML.
- Créez un fichier descripteur d'application AIR (utilisez l'espace de noms 2.5 ou ultérieur).
- Compilez l'application.
- Mettez en package l'application au format Android (.apk).
- Installez le moteur d'exécution AIR sur le périphérique ou l'émulateur Android (si vous utilisez un moteur d'exécution externe ; le moteur d'exécution captif est la valeur par défaut dans AIR 3.7 et versions ultérieures).
- Installez l'application sur le périphérique (ou l'émulateur Android).
- Lancez l'application sur le périphérique.

Pour exécuter cette procédure, vous disposez d'Adobe Flash Builder, d'Adobe Flash Professional CS5 ou des outils de ligne de commande.

Une fois l'application AIR terminée et mise en package au format APK, vous pouvez la proposer à Android Market ou la distribuer par un autre moyen.

AIR for iOS

Les étapes ci-dessous décrivent une procédure de développement typique d'une application AIR for iOS :

- Installez iTunes.

- Générez les ID et fichiers de développement requis sur le portail iOS Provisioning Portal d'Apple, à savoir :
  - Certificat de développement
  - ID d'application
  - Profil de configuration

  Vous devez recenser l'ID de tous les périphériques de test sur lesquels vous envisagez d'installer l'application lors de la création du profil de configuration.
- Convertissez le certificat de développement et la clé privée en fichier de keystore P12.
- Programmez le code de l'application en ActionScript ou MXML.
- Compilez l'application par le biais d'un compilateur ActionScript ou MXML.
- Créez les icônes et graphiques de l'écran initial de l'application.
- Créez le fichier descripteur de l'application (utilisez l'espace de noms 2.6 ou ultérieur).
- Mettez en package le fichier IPA à l'aide de l'outil ADT.
- Placez le profil de configuration sur le périphérique de test via iTunes.
- Installez et testez l'application sur le périphérique iOS. Vous pouvez installer le fichier IPA avec iTunes ou l'outil ADT via USB (prise en charge d'USB sur AIR 3.4 et les versions ultérieures).

Une fois l'application AIR terminée, vous pouvez la mettre à nouveau en package avec un certificat de distribution et un profil de configuration. Vous pouvez alors l'envoyer à l'App Store d'Apple.

Définition des propriétés d'une application mobile

A l'instar des autres applications AIR, vous définissez les propriétés de base d'une application dans le fichier descripteur correspondant. Les applications mobiles ignorent certaines propriétés propres au bureau, telles que la transparence et la taille des fenêtres. Elles peuvent également utiliser les propriétés propres à leur plate-forme. Vous pouvez, par exemple, inclure un élément `android` pour les applications Android et un élément `iPhone` pour les applications iOS.

Paramètres standard

Divers paramètres de descripteur d'application jouent un rôle important dans toutes les applications pour périphériques mobiles.

Version du moteur d'exécution d'AIR requise

Indiquez la version du moteur d'exécution d'AIR requise par l'application à l'aide de l'espace de noms du fichier descripteur de l'application.

Assigné dans l'élément `application`, l'espace de noms détermine globalement les fonctionnalités dont dispose l'application. Si, par exemple, l'application utilise l'espace de noms AIR 2.7 et que l'utilisateur a installé une version ultérieure, elle continue à assimiler le comportement d'AIR à celui de la version 2.7, même si celui-ci a été modifié dans la version ultérieure. Pour que l'application accède au nouveau comportement et aux nouvelles fonctionnalités, vous devez au préalable modifier l'espace de noms et publier une mise à jour. Les correctifs de sécurité constituent une exception notable à la règle.

Sur les périphériques qui utilisent un moteur d'exécution distinct de l'application, tel Android sur AIR 3.6 et versions antérieures, l'utilisateur est invité à installer ou mettre à niveau AIR s'il ne dispose pas de la version requise. Sur les périphériques qui intègrent le moteur d'exécution, tel iPhone, ce cas de figure ne se produit pas, étant donné que la version requise est mise en package avec l'application.

Remarque : *(AIR 3.7 et versions ultérieures) Par défaut, ADT compresse le moteur d'exécution pour les applications Android.*

Spécifiez l'espace de noms à l'aide de l'attribut xmlns de l'élément racine `application`. Les espaces de noms suivants sont adaptés aux applications mobiles (selon la plate-forme mobile ciblée) :

```
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">
```

Remarque : *la prise en charge des périphériques iOS 3 est assurée par le kit SDK de Packager for iPhone, basé sur le kit SDK d'AIR 2.0. Pour plus d'informations sur la création d'applications AIR pour iOS 3, voir Création d'applications iPhone. Le kit SDK d'AIR 2.6 (et versions ultérieures) prend en charge iOS 4 et les versions ultérieures sur les dispositifs iPhone 3G, iPhone 4 et iPad.*

Voir aussi

« application » à la page 222

Identité d'une application

Plusieurs paramètres devraient être propres à chaque application publiée, à savoir l'ID, le nom et le nom de fichier.

ID d'une application Android

Sous Android, l'ID est converti en nom de package AIR en ajoutant le préfixe « air » à l'ID AIR. Par conséquent, si l'AID AIR est *com.exemple.MonApp*, le nom du package Android correspond à *air.com.exemple.MonApp*.

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

En outre, si l'ID n'est pas un nom de package géré par le système d'exploitation Android, il est converti en nom valide. Les tirets sont remplacés par des traits de soulignement et les chiffres de début de tout composant ID sont précédés d'un « A » majuscule. Exemple : l'ID *3-goats.1-boat* est converti comme suit : *air.A3_goats.A1_boat*.

Remarque : *le préfixe ajouté à l'ID d'application permet d'identifier les applications AIR dans Android Market. Si vous ne souhaitez pas que l'application soit identifiée comme une application AIR en raison du préfixe, vous devez extraire du package le fichier APK, modifier l'ID de l'application, puis remettre celle-ci en package comme indiqué dans Opt-out of AIR application analytics for Android (disponible en anglais uniquement).*

ID d'une application iOS

Définissez l'ID de l'application AIR sur l'ID d'application créée dans le portail iOS Provisioning Portal d'Apple.

Les ID d'application iOS contiennent un identifiant d'application (préfixe), suivi d'un identifiant d'application (suffixe). L'identifiant d'application (préfixe) est une chaîne de caractères, telle que 5RM86Z4DJM, affectée par Apple à l'ID d'application. L'identifiant d'application (suffixe) contient un nom de type domaine inversé que vous sélectionnez. Il se termine parfois par un astérisque (*), qui représente un ID d'application à caractère générique. S'il se termine par un caractère générique, vous pouvez remplacer ce dernier par toute chaîne valide.

Exemple :

- Si l'ID de l'application Apple est `5RM86Z4DJM.com.example.helloWorld`, vous devez utiliser `com.example.helloWorld` dans le descripteur d'application.
- Si l'ID d'application Apple est `96LPVWEASL.com.example.*` (ID d'application à caractère générique), vous pourriez utiliser `com.example.helloWorld`, `com.example.anotherApp` ou tout autre ID qui commence par `com.example`.
- Enfin, si l'ID de l'application Apple ne contient qu'un identifiant d'application (préfixe) et un caractère générique, tel que : `38JE93KJL.*`, vous pouvez utiliser tout ID d'application dans AIR.

Lorsque vous spécifiez l'ID de l'application, ignorez la partie identifiant d'application (préfixe) de l'ID d'application.

Voir aussi

« id » à la page 239

« filename » à la page 234

« name » à la page 247

Version d'une application

Dans AIR 2.5 et ultérieur, spécifiez la version de l'application dans l'élément `versionNumber`. L'élément `version` est à présent obsolète. Si vous définissez une valeur dans `versionNumber`, vous devez utiliser une séquence de trois nombres au plus, séparés par un point (« 0.1.2 », par exemple). Chaque segment du numéro de la version se compose de trois chiffres au plus. (En d'autres termes, la version la plus longue autorisée correspond à « 999.999.999 ».) Vous ne devez pas obligatoirement inclure trois segments dans le nombre. Les numéros de version « 1 » et « 1.0 » sont également valides.

Vous pouvez aussi définir le libellé de la version à l'aide de l'élément `versionLabel`. Si vous ajoutez un libellé de version, il remplace le numéro de version dans l'écran d'information de l'application Android, par exemple. Vous devez stipuler un libellé de version si une application est distribuée via Android Market. Si vous ne spécifiez pas de valeur `versionLabel` dans le descripteur de l'application AIR, la valeur `versionNumber` est affectée au champ du libellé de version Android.

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

Sous Android, l'élément `versionNumber` AIR est converti en entier Android `versionCode` à l'aide de la formule : `a*1000000 + b*1000 + c`, où a, b et c correspondent aux composants du numéro de version AIR : `a.b.c`.

Voir aussi

« version » à la page 255

« versionLabel » à la page 255

« versionNumber » à la page 256

Fichier SWF principal d'une application

Spécifiez le fichier SWF principal de l'application dans l'élément enfant `content` de l'élément `initalWindow`. Si vous ciblez des périphériques associés au profil mobile, vous devez utiliser un fichier SWF (les applications de type HTML n'étant pas prises en charge).

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

Vous devez inclure ce fichier dans le package AIR (à l'aide de l'outil ADT ou de l'IDE). Se contenter de faire référence au nom dans le descripteur d'application n'entraîne pas l'inclusion automatique du fichier dans le package.

Propriétés de l'écran principal

Plusieurs éléments enfants de l'élément initialWindow contrôlent le comportement et l'aspect initiaux du principal écran de l'application.

- **aspectRatio** : indique si l'application doit initialement s'afficher au format *portrait* (hauteur supérieure à la largeur), au format *landscape* (hauteur inférieure à la largeur) ou au format *any* (la scène s'oriente automatiquement dans toutes les orientations).

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

- **autoOrients** : indique si la scène doit changer automatiquement d'orientation lorsque l'utilisateur fait pivoter le périphérique ou effectue un autre geste d'orientation tel que l'ouverture ou la fermeture d'un clavier coulissant. Si cet élément est défini sur *false* (valeur par défaut), la scène ne change pas d'orientation en conjonction avec le périphérique.

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

- **depthAndStencil** : indique s'il est nécessaire d'utiliser le tampon de profondeur ou le tampon de modèle. Ces tampons s'utilisent normalement avec du contenu 3D.

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

- **fullScreen** : indique si l'application doit occuper la totalité de l'écran du périphérique ou le partager avec le chrome standard du système d'exploitation, tel qu'une barre d'état système.

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

- **renderMode** : indique si le moteur d'exécution doit effectuer le rendu de l'application via le processeur graphique (GPU) ou l'unité centrale principale (UC). En règle générale, le rendu sur GPU augmente la vitesse de rendu, mais diverses fonctionnalités, telles que certains modes de fusion et filtres PixelBender, ne sont pas disponibles en mode GPU. Par ailleurs, les fonctionnalités et restrictions du processeur graphique varient selon le périphérique et le pilote de périphérique. Veillez à tester l'application sur un éventail aussi large que possible de périphériques, en particulier en mode GPU.

  Vous pouvez définir le mode de rendu sur *gpu*, *cpu*, *direct* ou *auto*. La valeur par défaut est *auto*, qui active actuellement le mode UC.

  Remarque : *pour tirer profit de l'accélération par processeur graphique du contenu Flash sur les plates-formes AIR mobiles, Adobe recommande d'utiliser renderMode="direct" (c'est-à-dire, Stage3D) plutôt que renderMode="gpu". Adobe prend officiellement en charge les structures d'application basées sur Stage3D suivantes et recommande leur utilisation : Starling (2D) et Away3D (3D). Pour plus d'informations sur Stage3D et Starling/Away3D, voir* http://gaming.adobe.com/getstarted/.

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

  Remarque : *vous ne pouvez pas utiliser renderMode="direct" pour les applications qui s'exécutent en arrière-plan.*

  Le mode de rendu sur GPU est soumis aux restrictions suivantes :

  - La structure d'application de Flex ne prend pas en charge le mode de rendu sur GPU.
  - Les filtres ne sont pas pris en charge.
  - Les remplissages et fusions PixelBender ne sont pas pris en charge.

- Les modes de fusion suivants ne sont pas pris en charge : Calque, Alpha, Effacement, Superposition, Lumière crue, Eclaircir et Obscurcir.
- Il n'est pas recommandé d'activer le mode de rendu sur GPU dans une application qui lit la vidéo.
- En mode de rendu sur GPU, les champs de texte ne sont pas correctement déplacés vers un emplacement visible lors de l'ouverture du clavier virtuel. Pour assurer la visibilité du champ de texte lorsque l'utilisateur saisit du texte, déplacez-le vers la zone visible par le biais de la propriété softKeyboardRect de la scène et des événements de clavier logiciel.
- S'il est impossible d'effectuer le rendu d'un objet d'affichage sur le GPU, il n'est pas affiché du tout. Ainsi, si un filtre est appliqué à un objet d'affichage, ce dernier n'est pas affiché.

Remarque : *l'implémentation GPU pour iOS dans AIR 2.6 et les versions ultérieure est très différente de l'implémentation utilisée dans la version précédente, AIR 2.0. Des considérations d'optimisation différentes sont prises en compte.*

Voir aussi

« aspectRatio » à la page 225

« autoOrients » à la page 226

« depthAndStencil » à la page 229

« fullScreen » à la page 237

« renderMode » à la page 249

Profils pris en charge

Vous pouvez ajouter l'élément `supportedProfiles` pour stipuler les profils de périphérique pris en charge par l'application. Réservez le profil mobileDevice aux périphériques mobiles. Lorsque vous exécutez l'application avec l'application de débogage du lanceur AIR (ADL), celle-ci active le premier profil de la liste. Vous disposez également de l'indicateur `-profile` lorsque vous exécutez l'application ADL pour sélectionner un profil donné dans la liste de profils pris en charge. Si l'application s'exécute sous tous les profils, vous pouvez omettre complètement l'élément `supportedProfiles`. Dans ce cas de figure, le profil actif par défaut de l'application ADL correspond au profil de bureau.

Pour spécifier que l'application prend en charge le profil de périphérique mobile et le profil de bureau et que vous souhaitez normalement tester l'application avec le profil mobile, ajoutez l'élément suivant :

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

Voir aussi

« supportedProfiles » à la page 252

« Profils de périphérique » à la page 259

« Application de débogage du lanceur AIR (ADL) » à la page 168

Extensions natives requises

Les applications qui prennent en charge le profil `mobileDevice` peuvent recourir aux extensions natives.

Déclarez toutes les extensions natives auxquelles fait appel l'application AIR dans le descripteur de l'application. L'exemple suivant illustre la syntaxe permettant de spécifier deux extensions natives requises :

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

La valeur de l'élément `extensionID` est identique à celle de l'élément `id` dans le fichier descripteur de l'extension. Le fichier descripteur de l'extension est un fichier XML appelé extension.xml. Il est mis en package dans le fichier ANE fourni par le développeur de l'extension native.

Comportement du clavier virtuel

Définissez l'élément `softKeyboardBehavior` sur `none` pour désactiver le comportement d'exécution d'un panoramique et de redimensionnement automatique utilisé par le moteur d'exécution pour s'assurer que le champ de texte de saisie correspondant à la cible d'action est visible après l'affichage du clavier virtuel. Si vous désactivez le comportement automatique, il incombe à l'application de s'assurer que la zone de texte de saisie ou tout autre contenu approprié est visible une fois le clavier affiché. Vous pouvez utiliser la propriété `softKeyboardRect` de la scène en conjonction avec l'élément SoftKeyboardEvent pour détecter l'ouverture du clavier et déterminer la zone masquée par ce dernier.

Pour activer le comportement automatique, définissez la valeur de l'élément sur `pan` :

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

Etant donné que `pan` est la valeur par défaut, omettre l'élément `softKeyboardBehavior` active également le comportement automatique du clavier.

Remarque : *si le mode de rendu sur GPU est également activé, le comportement d'exécution du panoramique n'est pas pris en charge.*

Voir aussi

« softKeyboardBehavior » à la page 251

Stage.softKeyboardRect

SoftKeyboardEvent

Paramètres Android

Sur la plate-forme Android, vous disposez de l'élément `android` du descripteur d'application pour ajouter des informations au manifeste d'application Android (fichier de propriétés d'application utilisé par le système d'exploitation Android). L'outil ADT génère automatiquement le fichier Manifest.xml Android lorsque vous créez le package APK. AIR définit quelques propriétés sur les valeurs requises pour assurer le fonctionnement de diverses fonctionnalités. Toute autre propriété définie dans la section android du descripteur d'application AIR est ajoutée à la section correspondante du fichier Manifest.xml.

Remarque : *pour la plupart des applications AIR, vous devez définir les autorisations Android requises par l'application dans l'élément* `android`*, mais il est généralement inutile de définir d'autres propriétés.*

Vous ne pouvez définir que les attributs qui gèrent les valeurs de type chaîne, entier ou booléen. La définition de références sur des ressources du package d'application n'est pas prise en charge.

Remarque : *Le moteur d'exécution nécessite l'installation d'une version du kit SDK égale ou ultérieure à la version 14. Si vous souhaitez créer une application uniquement pour les versions ultérieures, vous devez vous assurer que le fichier manifeste inclut* `<uses-sdk android:minSdkVersion=""></uses-sdk>` *avec la version correcte appropriée.*

Paramètres réservés du manifeste Android

AIR définit plusieurs entrées manifeste dans le document manifeste Android généré pour s'assurer que les fonctionnalités de l'application et du moteur d'exécution fonctionnent correctement. Il est impossible de définir les paramètres suivants :

Élément manifest

Il est impossible de définir les attributs suivants de l'élément manifest :

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

Élément activity

Il est impossible de définir les attributs suivants de l'élément activity principal :

- android:label
- android:icon

Élément application

Il est impossible de définir les attributs suivants de l'élément application :

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

Autorisations Android

Le modèle de sécurité Android requiert que chaque application demande l'autorisation d'utiliser les fonctionnalités dotées d'implications d'ordre sécuritaire ou confidentiel. Vous spécifiez ces autorisations lors de la mise en package de l'application et il est impossible de les modifier à l'exécution. Le système d'exploitation Android avertit l'utilisateur des autorisations demandées par une application lors de l'installation de cette dernière. Si l'application ne demande pas une autorisation associée à une fonctionnalité, le système d'exploitation risque de renvoyer une exception lorsque l'application accède à la fonctionnalité. Le renvoi de l'exception n'est toutefois pas garanti. Le moteur d'exécution transmet les exceptions à l'application. En cas d'échec silencieux, un message d'échec de l'autorisation est inséré dans le journal système d'Android.

Dans AIR, vous spécifiez les autorisations Android dans l'élément `android` du descripteur d'application. L'ajout d'autorisations fait appel au format suivant (PERMISSION_NAME correspondant au nom d'une autorisation Android) :

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

Les instructions uses-permissions intégrées à l'élément `manifest` sont directement ajoutées au document manifeste Android.

Les autorisations suivantes sont obligatoires pour utiliser diverses fonctionnalités AIR :

ACCESS_COARSE_LOCATION Permet à l'application d'accéder aux données de localisation du réseau Wi-Fi ou cellulaire via la classe Geolocation.

ACCESS_FINE_LOCATION Permet à l'application d'accéder aux données GPS via la classe Geolocation.

ACCESS_NETWORK_STATE et ACCESS_WIFI_STATE Permet à l'application d'accéder aux informations du réseau via la classe NetworkInfo.

CAMERA Permet à l'application d'accéder à la caméra.

Remarque : *lorsque vous demandez l'autorisation d'utiliser la fonctionnalité caméra, Android part du principe que l'application requiert également la caméra. Si la caméra est une fonctionnalité facultative de l'application, ajoutez un élément* `uses-feature` *relatif à la caméra au manifeste en définissant l'attribut requis sur* `false`*. Voir « Filtrage de la compatibilité Android » à la page 83.*

INTERNET Permet à l'application d'effectuer des requêtes réseau. Permet également de procéder au débogage à distance.

READ_PHONE_STATE Permet au moteur d'exécution d'AIR de couper le son lors d'appels téléphoniques. Définissez cette autorisation si l'application lit l'audio lorsqu'elle s'exécute en arrière-plan.

RECORD_AUDIO Permet à l'application d'accéder au microphone.

WAKE_LOCK et DISABLE_KEYGUARD Permet à l'application d'empêcher le périphérique d'entrer en mode de veille à l'aide des paramètres de la classe SystemIdleMode.

WRITE_EXTERNAL_STORAGE Permet à l'application d'écrire sur la carte mémoire externe du périphérique.

Ainsi, si une application nécessite toutes les autorisations, vous pourriez ajouter l'élément suivant au descripteur de l'application pour les définir :

```
<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>
```

Voir aussi

Android Security and Permissions (disponible en anglais uniquement)

Android Manifest.permission class (disponible en anglais uniquement)

Modèles personnalisés d'URI Android

Vous pouvez utiliser un modèle URI personnalisé pour lancer une application AIR à partir d'une page Web ou une application Android native. Etant donné que la prise en charge des URI personnalisés repose sur des filtres d'intention spécifiés dans le manifeste Android, il est impossible d'utiliser cette technique sur d'autres plates-formes.

Pour utiliser un URI personnalisé, ajoutez un élément intent-filter au descripteur d'application dans le bloc `<android>`. Les deux éléments `intent-filter` de l'exemple suivant sont obligatoires. Modifiez l'instruction `<data android:scheme="`*`my-customuri`*`"/>` en fonction de la chaîne URI associée au modèle personnalisé.

```
<android>
                        <manifestAdditions>
                        <![CDATA[
                        <manifest>
                        <application>
                        <activity>
                        <intent-filter>
                        <action android:name="android.intent.action.MAIN"/>
                        <category android:name="android.intent.category.LAUNCHER"/>
                        </intent-filter>
                        <intent-filter>
                        <action android:name="android.intent.action.VIEW"/>
                        <category android:name="android.intent.category.BROWSABLE"/>
                        <category android:name="android.intent.category.DEFAULT"/>
                        <data android:scheme="my-customuri"/>
                        </intent-filter>
                        </activity>
                        </application>
                        </manifest>
                        ]]>
                        </manifestAdditions>
                        </android>
```

Un filtre d'intention avertit le système d'exploitation Android que l'application est en mesure d'exécuter une action donnée. Dans le cas d'un URI personnalisé, cela signifie que l'utilisateur a cliqué sur un lien basé sur ce modèle d'URI (et le navigateur ne sait pas comment le gérer).

Si l'application est appelée par le biais d'un URI personnalisé, l'objet NativeApplication distribue un événement `invoke`. L'URL du lien, paramètres d'interrogation inclus, est placée dans le tableau `arguments` de l'objet InvokeEvent. Vous disposez d'un nombre illimité d'éléments intent-filter.

Remarque : *les liens dans une occurrence de StageWebView ne peuvent pas ouvrir les URL qui font appel à un modèle d'URI personnalisé.*

Voir aussi

Android intent filters (disponible en anglais uniquement)

Android actions and categories (disponible en anglais uniquement)

Filtrage de la compatibilité Android

Le système d'exploitation Android utilise divers éléments du fichier manifeste de l'application pour déterminer si celle-ci est compatible avec un périphérique donné. L'ajout de ces informations au fichier manifeste est facultatif. Si vous n'incluez pas ces éléments, l'application peut être installée sur tout périphérique Android. Elle risque toutefois de ne fonctionner correctement sur aucun périphérique Android. Une application pour caméra s'avère par exemple d'une utilité limitée sur un téléphone qui n'est pas équipé de caméra.

Vous disposez des balises de manifeste Android suivantes à des fins de filtrage :

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

Applications pour caméra

Si vous demandez l'autorisation caméra pour l'application, Android part du principe que l'application nécessite toutes les fonctionnalités de caméra disponibles, y compris la mise au point automatique et le flash. Si l'application ne requiert pas toutes les fonctionnalités de caméra ou si la caméra est une fonctionnalité facultative, définissez les divers éléments `uses-feature` associés à la caméra pour indiquer qu'ils sont facultatifs. Les utilisateurs dont le périphérique ne gère pas une fonctionnalité ou n'est pas équipé de caméra ne pourront sinon pas trouver l'application dans Android Market.

L'exemple suivant illustre la procédure de demande de l'autorisation caméra et indique comment rendre toutes les fonctionnalités de la caméra facultatives :

```
<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>
```

Applications d'enregistrement de l'audio

Si vous demandez l'autorisation d'enregistrer l'audio, Android considère comme acquis que l'application nécessite un microphone. Si l'enregistrement de l'audio est une fonctionnalité facultative de l'application, vous disposez d'une balise uses-feature pour spécifier que le microphone est inutile. Les utilisateurs dont le périphérique n'est pas équipé d'un microphone ne pourront sinon pas trouver l'application sur Android Market.

L'exemple suivant illustre la procédure de demande d'une autorisation d'utilisation du microphone, le matériel associé restant facultatif :

```
<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>
```

Voir aussi

Android Developers : Android Compatibility (disponible en anglais uniquement)

Android Developers : Android feature name constants (disponible en anglais uniquement)

Emplacement d'installation

Vous pouvez autoriser l'installation ou le transfert de l'application sur la carte mémoire externe en définissant l'attribut `installLocation` de l'élément Android `manifest` sur `auto` ou `preferExternal` :

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

Le système d'exploitation Android ne garantit pas que l'application soit installée sur la carte mémoire externe. Un utilisateur peut également transférer une application entre la mémoire interne et la mémoire externe par le biais de l'application Réglages du système.

Même si l'application est installée sur la carte mémoire externe, le cache d'application, les données utilisateur telles que le contenu du répertoire app-storage, les objets partagés et les fichiers temporaires continuent à être stockés en mémoire interne. Pour éviter de solliciter trop de mémoire interne, sélectionnez avec discernement les données enregistrées dans le répertoire de stockage de l'application. Enregistrez les volumes élevés de données sur la carte SD (emplacements `File.userDirectory` ou `File.documentsDirectory`, qui pointent tous deux vers la racine de la carte SD sous Android).

Activation de Flash Player et d'autres modules d'extension dans un objet StageWebView

Dans Android 3.0 et les versions ultérieures, l'application doit activer l'accélération matérielle dans l'élément de l'application Android pour afficher le contenu des modules d'extension dans un objet StageWebView. Pour activer le rendu sur le module d'extension, définissez l'attribut `android:hardwareAccelerated` de l'élément `application` sur `true` :

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

Codage des couleurs

AIR 3+

Dans AIR 3 et les versions ultérieures, le moteur d'exécution configure l'écran de façon à ce qu'il effectue le rendu des couleurs 32 bits. Dans les versions antérieures d'AIR, le moteur d'exécution utilise les couleurs 16 bits. Vous pouvez demander au moteur d'exécution d'utiliser les couleurs 16 bits à l'aide de l'élément <colorDepth> du descripteur d'application :

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

L'utilisation de la profondeur de couleurs 16 bits peut augmenter les performances de rendu, au détriment de la fidélité des couleurs.

Paramètres iOS

Les paramètres réservés aux périphériques iOS sont placés dans l'élément `<iPhone>` du descripteur de l'application. L'élément `iPhone` peut avoir un élément `InfoAdditions`, un élément `requestedDisplayResolution`, un élément `Entitlements`, un élément `externalSwfs` et un élément `forceCPURenderModeForDevices` comme enfants.

L'élément `InfoAdditions` permet de spécifier des paires clé-valeur ajoutées au fichier de paramètres Info.plist associé à l'application. Les valeurs suivantes déterminent, par exemple, le style de la barre d'état de l'application et stipulent que cette dernière ne nécessite pas un accès Wi-Fi permanent.

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

Les paramètres InfoAdditions sont entourés d'une balise `CDATA`.

L'élément `Entitlements` permet de spécifier les paires clé-valeur ajoutées au fichier de paramètres Entitlements.plist de l'application. Les paramètres Entitlements.plist permettent à certaines fonctions d'iOS, notamment aux notifications Push, d'accéder à l'application.

Pour plus d'informations sur les paramètres Info.plist et Entitlements.plist, voir la documentation Apple destinée aux développeurs.

Prise en charge de tâches en arrière-plan sous iOS

AIR 3.3

Adobe AIR 3.3 et les versions ultérieures prennent en charge les multitâches sous iOS en permettant certains comportements en arrière-plan :

- Audio
- Mises à jour de l'emplacement
- Mise en réseau
- Annulation de l'exécution d'une application en arrière-plan

Remarque : *avec les versions 21 et antérieures de SWF, AIR ne prend pas en charge l'exécution en arrière-plan sur iOS et Android lorsque le mode de rendu est défini sur direct. En raison de cette restriction, les applications basées sur Stage3D ne peuvent pas exécuter des tâches en arrière-plan telles que la lecture de l'audio, les mises à jour d'emplacement, les téléchargements vers/depuis le réseau, etc. iOS n'autorise pas les appels OpenGLES/de rendu en arrière-plan. Il interrompt ainsi les applications qui tentent de lancer des appels en arrière-plan. Android ne restreint pas les appels OpenGLES ou toute autre tâche en arrière-plan lancée par les applications (telle que la lecture d'audio). Avec les versions 22 et ultérieures de SWF, les applications mobiles AIR peuvent s'exécuter en arrière-plan lorsque renderMode est définir sur direct. Le moteur d'exécution AIR iOS entraîne une erreur ActionScript (3768 - L'API Stage3D ne peut pas être utilisée lors d'une exécution en arrière-plan) si les appels OpenGLES sont effectués en arrière-plan. Cependant, aucune erreur ne se produit sur Android car ses applications natives sont autorisées à effectuer des appels OpenGLES en arrière-plan. Pour une utilisation optimale des ressources mobiles, n'effectuez pas d'appels de rendu lorsqu'une application s'exécute en arrière-plan.*

Son en arrière-plan

Pour activer la lecture et l'enregistrement audio en arrière-plan, incluez la paire clé-valeur suivante à l'élément `InfoAdditions` :

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

Mises à jour de l'emplacement en arrière-plan

Pour activer les mises à jour de l'emplacement en arrière-plan, incluez la paire clé-valeur suivante à l'élément `InfoAdditions` :

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

Remarque : *utilisez cette fonction uniquement lorsque cela est nécessaire, car les API d'emplacement consomment énormément de batterie.*

Mise en réseau en arrière-plan

Pour exécuter de brèves tâches en arrière-plan, votre application définit la propriété `NativeApplication.nativeApplication.executeInBackground` sur `true`.

Par exemple, votre application peut lancer une opération de mise à jour d'un fichier après que l'utilisateur place au premier plan une autre application. Lorsque l'application reçoit un événement de fin de téléchargement, elle peut définir `NativeApplication.nativeApplication.executeInBackground` sur `false`.

Définir la propriété `NativeApplication.nativeApplication.executeInBackground` sur `true` ne garantit pas l'exécution indéfinie de l'application, car iOS impose une limite temporelle aux tâches en arrière-plan. Lorsque iOS arrête le traitement en arrière-plan, AIR distribue un événement `NativeApplication.suspend`.

Annulation de l'exécution en arrière-plan

Votre application peut annuler explicitement l'exécution en arrière-plan en incluant la paire clé-valeur suivante à l'élément `InfoAdditions` :

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

Paramètres InfoAdditions iOS réservés

AIR définit plusieurs entrées dans le fichier Info.plist généré pour s'assurer que les fonctionnalités de l'application et du moteur d'exécution fonctionnent correctement. Il est impossible de définir les paramètres suivants :

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

Remarque : *Vous pouvez définir MinimumOSVersion. La définition de MinimumOSVersion est prise en charge dans Air 3.3 et les versions ultérieures.*

Prise en charge de différents modèles de périphérique iOS

Pour prendre en charge l'iPad, incluez les paramètres clé-valeur appropriés de `UIDeviceFamily` dans l'élément `InfoAdditions`. Le paramètre `UIDeviceFamily` est un tableau de chaînes. Chaque chaîne définit les périphériques pris en charge. Le paramètre `<string>1</string>` définit la prise en charge de l'iPhone et de l'iPod touch. Le paramètre `<string>2</string>` définit la prise en charge de l'iPad. Le paramètre `<string>3</string>` définit la prise en charge de tvOS. Si vous spécifiez uniquement l'un de ces paramètres, seule cette famille de périphériques est prise en charge. Par exemple, le paramètre suivant limite la prise en charge à l'iPad :

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

Le paramètre suivant prend en charge les deux familles de périphériques (iPhone/iPod Touch et iPad) :

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

En outre, dans AIR 3.7 et les versions ultérieures, vous pouvez utiliser la balise `forceCPURenderModeForDevices` balise pour forcer le mode de rendu UC pour un ensemble de périphériques spécifiés et activer le mode de rendu GPU les périphériques iOS restants.

Vous ajoutez cette balise comme enfant de la balise `iPhone` et spécifiez une liste de noms de modèles séparés par des espaces. Pour obtenir une liste de noms de modèle valides, reportez-vous à « forceCPURenderModeForDevices » à la page 236.

Par exemple, pour utiliser le mode UC dans les anciens iPods, iPhones et iPads, et activer le mode GPU pour tous les autres périphériques, spécifiez ce qui suit dans le descripteur d'application :

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

Affichages à haute résolution

L'élément `requestedDisplayResolution` indique si l'application doit utiliser le mode de résolution *standard* ou *high* sur les périphériques iOS équipés d'un écran à haute résolution.

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

Le mode Haute résolution permet d'adresser individuellement chaque pixel d'un écran à haute résolution. En mode Standard, l'écran du périphérique est assimilé par l'application à un écran à résolution standard. Dessiner un pixel unique en ce mode définit la couleur de quatre pixels sur un écran à haute résolution.

Le paramètre par défaut est `standard`. Notez que, afin de cibler les périphériques iOS, vous utilisez l'élément `requestedDisplayResolution` en tant qu'enfant de l'élément `iPhone` (et non de l'élément `InfoAdditions` ou `initialWindow`).

Si vous voulez utiliser d'autres paramètres sur d'autres périphériques, spécifiez la valeur par défaut comme la valeur de l'élément `requestedDisplayResolution`. Utilisez l'attribut `excludeDevices` afin de spécifier les périphériques qui doivent utiliser la valeur opposée. Par exemple, avec le code suivant, le mode haute résolution est utilisé sur tous les périphériques qui le prennent en charge, sauf les iPad de 3e génération, qui utilisent le mode standard.

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

L'attribut `excludeDevices` est disponible dans AIR 3.6 et ultérieur.

Voir aussi

« requestedDisplayResolution » à la page 250

Renaun Erickson : Developing for both retina and non-retina iOS screens using AIR 2.6 (disponible en anglais uniquement)

Modèles d'URI personnalisés iOS

L'enregistrement d'un modèle d'URI personnalisé permet d'appeler l'application à l'aide d'un lien inséré dans une page Web ou d'une autre application native installée sur le périphérique. Pour enregistrer un modèle d'URI, ajoutez une clé CFBundleURLTypes à l'élément InfoAdditions. L'exemple suivant enregistre un modèle d'URI, *com.example.app*, afin de permettre l'appel d'une application par le biais d'URL de type : *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>
```

Si l'application est appelée par le biais d'un URI personnalisé, l'objet NativeApplication distribue un événement `invoke`. L'URL du lien, paramètres d'interrogation inclus, est placée dans le tableau `arguments` de l'objet InvokeEvent. Vous disposez d'un nombre illimité de modèles d'URI.

Remarque : *les liens dans une occurrence de StageWebView ne peuvent pas ouvrir les URL qui font appel à un modèle d'URI personnalisé.*

Remarque : *si une autre application a déjà enregistré un modèle d'URI, il est impossible d'associer votre application au modèle.*

Filtrage de la compatibilité iOS

Ajoutez des entrées à un tableau UIRequiredDeviceCapabilities dans l'élément `InfoAdditions` si l'application doit être réservée aux périphériques dotés de fonctionnalités matérielles ou logicielles déterminées. L'entrée suivante indique, par exemple, qu'une application requiert un appareil photo et un microphone :

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

Si le périphérique n'est pas équipé de ces fonctionnalités, il est impossible d'installer l'application. Les paramètres de fonctionnalités associés aux applications AIR sont les suivants :

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

AIR 2.6+ ajoute automatiquement *armv7* et *opengles-2* à la liste des fonctionnalités requises.

Remarque : *il est inutile d'inclure ces fonctionnalités dans le descripteur de l'application pour que cette dernière les utilise. Ne faites appel aux paramètres UIRequiredDeviceCapabilities que pour interdire aux utilisateurs d'installer l'application sur un périphérique sur lequel elle ne peut pas fonctionner correctement.*

Fermeture plutôt que mise en pause

Lorsqu'un utilisateur quitte par basculement une application AIR, elle s'exécute en arrière-plan et est mise en pause. Pour que l'application se ferme complètement au lieu d'être mise en pause, définissez la propriété `UIApplicationExitsOnSuspend` sur `YES` :

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

Réduire la taille des téléchargements en chargeant des fichiers SWF externes d'actifs uniquement

AIR 3.7

Vous pouvez réduire la taille de téléchargement initiale de l'application en compressant un sous-ensemble des fichiers SWF utilisées par votre application et en chargeant les fichiers SWF externes restants (actif uniquement) lors de l'exécution à l'aide de la méthode `Loader.load()`. Pour utiliser cette fonction, vous devez compresser l'application de telle sorte qu'ADT déplace tout le pseudo-code ActionScript (ABC) depuis les fichiers SWF chargés en externe vers le fichier SWF principal de l'application, laissant un fichier SWF qui contient uniquement des actifs. Cela est nécessaire pour respecter la règle de l'Apple Store qui interdit le téléchargement tout code une fois qu'une application est installée.

ADT effectue les opérations suivantes pour prendre en charge les fichiers SWF chargés en externe (également appelés fichiers SWF démunis) :

- Lit le fichier texte spécifié dans le sous-élément `<externalSwfs>` de l'élément `<iPhone>` pour accéder à la liste de fichiers SWF séparés par une ligne à charger à la prochaine exécution :

  ```
  <iPhone>
                                          ...

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

- Transfert le code ABC de chaque fichier SWF chargé en externe vers l'exécutable principal.
- Omet les fichiers SWF chargés en externe du fichier .ipa.
- Copie les fichiers SWF démunis dans le répertoire `.remoteStrippedSWFs`. Vous hébergez ces fichiers SWF sur un serveur Web et votre application les charge, au besoin, au moment de l'exécution.

Vous devez indiquer les fichiers SWF à charger au moment de l'exécution en spécifiant leurs noms, un par ligne dans un fichier texte, comme le montre l'exemple suivant :

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

Le chemin d'accès du fichier spécifié est relatif au fichier descripteur de l'application. En outre, vous devez spécifier ces fichiers SWF comme actifs dans la commande `adt`.

Remarque : *Cette fonction s'applique aux compressions standard uniquement. Pour une compression rapide (par exemple à l'aide d'un interpréteur, d'un simulateur, ou d'un débogueur) ADT ne crée pas de fichiers SWF démunis.*

Pour plus d'informations sur cette fonction, y compris un exemple de code, reportez-vous à Hébergement externe de fichiers SWF secondaires pour applications AIR sur iOS, une publication de blog par Abhinav Dhandh, ingénieur chez Adobe.

Prise en charge de la géolocalisation

Pour la prise en charge de la géolocalisation, ajoutez l'une des paires clé-valeur suivantes à l'élément `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>
```

Icônes d'une application

Le tableau suivant indique les tailles d'icônes utilisées sur chaque plate-forme mobile :

Taille d'icône	Plate-forme
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
192 x 192	Android
512 x 512	Android, iOS
1 024 x 1024	iOS

Spécifiez le chemin d'accès aux fichiers d'icône dans l'élément icon du fichier descripteur de l'application :

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

Si vous ne fournissez pas d'icône de la taille indiquée, la taille suivante est utilisée et l'image est mise à l'échelle en conséquence.

Icônes sous Android

Sous Android, les icônes spécifiées dans le descripteur d'application servent d'icône de lancement de l'application. L'icône de lancement de l'application doit être fournie sous la forme d'un ensemble d'images PNG de 36x36, 48x48, 72x72, 96x96, 144x144 et 192x192 pixels. Ces tailles d'icône correspondent respectivement aux écrans de faible densité, densité moyenne et densité élevée.

Les développeurs doivent envoyer l'icône de 512x512 pixels au moment de l'envoi de l'application sur Google Play Store.

Icônes sous iOS

Les icônes définies dans le descripteur d'application sont utilisées comme suit pour une application iOS :

- Icône de 29 par 29 pixels : icône Recherche Spotlight pour les iPhone/iPod de faible résolution et icône Réglages pour les iPad de faible résolution.
- Icône de 40 par 40 pixels : icône de recherche Spotlight pour les iPad de faible résolution.
- Icône de 48 par 48 pixels : AIR ajoute une bordure à cette image et l'utilise en tant qu'icône 50x50 pour la recherche Spotlight sur les iPad de faible résolution.
- Icône de 50 par 50 pixels : recherche Spotlight pour les iPad de faible résolution.
- Icône de 57 par 57 pixels : icône de l'application pour les iPhone/iPod de faible résolution.
- Icône de 58 par 58 pixels : icône Spotlight pour les iPhone/iPod munis d'un écran Retina et icône Réglages pour les iPad munis d'un écran Retina.
- Icône de 60 par 60 pixels : icône de l'application pour les iPhone/iPod de faible résolution.
- Icône de 72 par 72 pixels (facultatif) : icône de l'application pour les iPad de faible résolution.
- Icône de 76 par 76 pixels (facultatif) : icône de l'application pour les iPad de faible résolution.
- Icône de 80 par 80 pixels : recherche Spotlight pour les iPhone/iPod/iPad de haute résolution.
- Icône de 100 par 100 pixels : recherche Spotlight pour les iPad munis d'un écran Retina.
- Icône de 114 par 114 pixels : icône de l'application pour les iPhone/iPod munis d'un écran Retina.
- Icône de 120 par 120 pixels : icône de l'application pour les iPhone/iPod de haute résolution.
- Icône de 152 par 152 pixels : icône de l'application pour les iPad de haute résolution.
- Icône de 167 par 167 pixels : icône de l'application pour les iPad Pro de haute résolution.
- Icône de 512 par 512 pixels : icône de l'application pour les iPhone/iPod/iPad de faible résolution. iTunes affiche cette icône. Le fichier PNG de 512 pixels est réservé au test des versions de développement de l'application. Lorsque vous envoyez l'application définitive à l'App Store d'Apple, vous transmettez séparément l'image de 512 pixels au format JPG. Elle n'est pas incluse dans l'IPA.
- Icône de 1 024 par 1 024 pixels : icône de l'application pour les iPhone/iPod/iPad munis d'un écran Retina.

Etant donné qu'iOS ajoute un effet de lueur jaune à une icône, il est inutile d'appliquer l'effet à l'image source. Pour désactiver cet effet de lueur jaune activé par défaut, ajoutez le texte ci-dessous à l'élément `InfoAdditions` du fichier descripteur d'application :

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

Remarque : *sur iOS, les métadonnées de l'application sont insérées en tant que métadonnées png dans les icônes de l'application afin qu'Adobe puisse connaître le nombre d'applications AIR disponibles dans l'App Store iOS d'Apple. Si vous ne souhaitez pas que votre application soit identifiée comme une application AIR, vous devez extraire le fichier IPA du package, retirer les métadonnées de l'icône, puis remettre le fichier IPA en package. Cette procédure est décrite dans l'article Opt-out of AIR application analytics for iOS (disponible en anglais uniquement).*

Voir aussi

« icon » à la page 238

« imageNxN » à la page 239

Android Developers : Icon Design Guidelines (disponible en anglais uniquement)

iOS Human Interface Guidelines: Custom Icon and Image Creation Guidelines (disponible en anglais uniquement)

Images de lancement iOS

Outre les icônes de l'application, vous devez fournir au moins une image de lancement appelée *Default.png*. Vous pouvez, si vous le souhaitez, inclure des images de lancement distinctes selon l'orientation au démarrage, la résolution (notamment sur les écrans Retina haute résolution et les écrans au format 16:9) et le périphérique. Vous pouvez également inclure différentes images de lancement à utiliser lorsque l'utilisateur appelle l'application par le biais d'une URL.

Les fichiers d'image de lancement ne sont pas référencés dans le descripteur de l'application et doivent résider dans le répertoire racine de cette dernière. (Ne placez *pas* les fichiers dans un sous-répertoire.)

Modèle d'affectation de noms de fichiers

Attribuez un nom à l'image comme suit :

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

La partie *basename* du nom de fichier est l'unique partie requise. Elle correspond soit à *Default* (avec un D majuscule), soit au nom spécifié à l'aide de la clé `UILaunchImageFile` dans l'élément `InfoAdditions` du descripteur de l'application.

La partie *screen size modifier* désigne la taille de l'écran lorsqu'elle ne correspond à aucune taille d'écran standard. Ce modificateur s'applique uniquement aux modèles d'iPhone et d'iPod dotés d'écrans 16:9, tels que l'iPhone 5 et l'iPod touch de 5e génération. L'unique valeur prise en charge par ce modificateur est `-568h`. Etant donné que ces périphériques prennent en charge les affichages haute résolution (Retina), le modificateur de taille d'écran est toujours utilisé avec une image possédant également le modificateur d'échelle `@2x`. Le nom d'image de lancement complet par défaut pour ces périphériques est `Default-568h@2x.png`.

La partie *urischeme* correspond à la chaîne d'identification du modèle d'URI. Cette partie s'applique uniquement si votre application mobile prend en charge un ou plusieurs modèles d'URL personnalisés. Par exemple, s'il est possible d'appeler l'application par le biais d'un lien tel que `example://foo`, faites de `-example` la partie modèle du nom de fichier de l'image de lancement.

La partie *orientation* permet de spécifier plusieurs images de lancement selon l'orientation du périphérique lors du démarrage de l'application. Cette partie s'applique uniquement aux images des applications de l'iPad. Elle peut posséder l'une des valeurs suivantes, selon l'orientation du périphérique lorsque l'application démarre :

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

La partie *scale* correspond à `@2x` (pour l'iPhone 4, l'iPhone 5 et l'iPhone 6) ou `@3x` (pour l'iPhone 6 plus) pour les images de lancement réservées aux écrans haute résolution (retina). (Omettez complètement la partie scale pour les images associées aux écrans à résolution standard.) Pour les images de lancement des périphériques plus grands, tels que l'iPhone 5 et l'iPod touch de 5e génération, vous devez également spécifier le modificateur de taille d'écran `-528h` après la partie basename et avant tout autre partie.

La partie *device* est utilisée pour désigner les images de lancement des périphériques et des téléphones de poche. Cette partie est utilisée lorsque votre application est une application mobile universelle qui prend en charge les périphériques de poche et les tablettes avec une seule interface binaire-programme. La valeur doit être soit `~ipad` soit `~iphone` (pour l'iPhone et l'iPod Touch).

Pour l'iPhone, vous ne pouvez inclure que les images au format portrait. Cependant, dans le cas d'un iPhone 6 plus, les images en mode paysage peuvent également être ajoutées. Utilisez des images de 320 x 480 pixels pour les périphériques de résolution standard, des images de 640 x 960 pixels pour les périphériques haute résolution, et des images 640 x 1 136 pixels pour les périphériques 16:9 tels que l'iPhone 5 et l'iPod touch de 5e génération.

Pour l'iPad, vous pouvez inclure les images suivantes :

- AIR 3.3 et versions antérieures : images qui ne sont pas au mode plein écran : vous pouvez inclure des images au format paysage (1 024 x 748 en résolution normale, 2 048 x 1 496 en haute résolution) et au format portrait (768 x 1 004 en résolution normale, 1 536 x 2 008 en haute résolution).
- AIR 3.4 et versions ultérieures : images plein écran : vous pouvez inclure des images au format paysage (1 024 x 768 en résolution normale, 2 048 x 1 536 en haute résolution) et au format portrait (768 x 1 024 en résolution normale, 1 536 x 2 048 en haute résolution). Notez que lorsque vous mettez en package une image plein écran pour une application qui n'est pas affichée en plein écran, les 20 pixels supérieurs (les 40 pixels supérieurs pour une image haute résolution) sont masqués par la barre d'état. Evitez d'afficher des informations importantes dans cette zone.

Exemples

Le tableau suivant contient un exemple d'ensemble d'images de lancement que vous pourriez associer à une application hypothétique qui prend en charge l'éventail le plus large possible de périphériques et d'orientations. L'utilisateur peut lancer l'application par le biais d'URL basées sur le modèle `example://` :

Nom du fichier	Taille de l'image	Utilisation
Default.png	320 x 480	iPhone, résolution standard
Default@2x.png	640 x 960	iPhone, résolution élevée
Default-568h@2x.png	640 x 1 136	iPhone, haute résolution, format d'image 16:9

Nom du fichier	Taille de l'image	Utilisation
Default-Portrait.png	768 x 1 004 (AIR 3.3 et versions antérieures) 768 x 1 024 (AIR 3.4 et versions ultérieures)	iPad, orientation portrait
Default-Portrait@2x.png	1 536 x 2 008 (AIR 3.3 et versions antérieures) 1 536 x 2 048 (AIR 3.4 et versions ultérieures)	iPad, haute résolution, orientation portrait
Default-PortraitUpsideDown.png	768 x 1 004 (AIR 3.3 et versions antérieures) 768 x 1 024 (AIR 3.4 et versions ultérieures)	iPad, orientation portrait inversée
Default-PortraitUpsideDown@2x.png	1 536 x 2 008 (AIR 3.3 et versions antérieures) 1 536 x 2 048 (AIR 3.4 et versions ultérieures)	iPad, haute résolution, orientation portrait à l'envers
Default-Landscape.png	1 024 x 768	iPad, orientation paysage gauche
Default-LandscapeLeft@2x.png	2 048 x 1 536	iPad, haute résolution, orientation paysage vers la gauche
Default-LandscapeRight.png	1 024 x 768	iPad, orientation paysage droite
Default-LandscapeRight@2x.png	2 048 x 1 536	iPad, haute résolution, orientation paysage vers la droite
Default-example.png	320 x 480	example:// URL sur un iPhone standard
Default-example@2x.png	640 x 960	example:// URL sur un iPhone à résolution élevée
Default-example~ipad.png	768 x 1 004	example:// URL sur un iPad (orientations portrait)
Default-example-Landscape.png	1 024 x 768	example:// URL sur un iPad (orientations paysage)

Cet exemple illustre uniquement une approche. Vous pourriez notamment utiliser l'image `Default.png` pour l'iPad, et spécifier des images de lancement spécifiques pour l'iPhone et l'iPod avec `Default~iphone.png` et `Default@2x~iphone.png`.

Voir aussi

iOS Application Programming Guide: Application Launch Images (disponible en anglais uniquement)

Images de lancement à assembler pour les périphériques iOS

Périphériques	Résolution (pixels)	Nom de l'image de lancement	Orientation
iPhone			
iPhone 4 (non retina)	640 x 960	Default~iphone.png	Portrait
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Portrait
iPhone 5, 5c, 5s	640 x 1 136	Default-568h@2x~iphone.png	Portrait

iPhone6, iPhone7	750 x 1 334	Default-375w-667h@2x~iphone.png	Portrait
iPhone6+, iPhone 7+	1 242 x 2 208	Default-414w-736h@3x~iphone.png	Portrait
iPhone6+, iPhone7+	2 208 x 1 242	Default-Landscape-414w-736h@3x~iphone.png	Landscape
iPad			
iPad 1, 2	768 x 1 024	Default-Portrait~ipad.png	Portrait
iPad 1, 2	768 x 1 024	Default-PortraitUpsideDown~ipad.png	Portrait à l'envers
iPad 1, 2	1 024 x 768	Default-Landscape~ipad.png	Paysage gauche
iPad 1, 2	1 024 x 768	Default-LandscapeRight~ipad.png	Paysage droit
iPad 3, Air	1 536 x 2 048	Default-Portrait@2x~ipad.png	Portrait
iPad 3, Air	1 536 x 2 048	Default-PortraitUpsideDown@2x~ipad.png	Portrait à l'envers
iPad 3, Air	2 048 x 1 536	Default-LandscapeLeft@2x~ipad.png	Paysage gauche
iPad 3, Air	2 048 x 1 536	Default-LandscapeRight@2x~ipad.png	Paysage droit
iPad Pro	2048 x 2732	Default-Portrait@2x.png	Portrait
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Landscape

Directives de création

Sous réserve de respecter les dimensions requises, libre à vous de créer n'importe quel graphique pour une image de lancement. Il est toutefois préférable que l'image corresponde à l'état initial de l'application. Vous pouvez créer une telle image de lancement en capturant l'écran de démarrage de l'application :

1. Ouvrez l'application sur le périphérique iOS. Lorsque le premier écran de l'interface utilisateur apparaît, appuyez sur le bouton principal (figurant sous l'écran) et maintenez-le enfoncé. Tout en maintenant enfoncé le bouton principal, appuyez sur le bouton Marche/Veille figurant dans la partie supérieure du périphérique. Vous effectuez ainsi une capture d'écran, qui est envoyée à Pellicule.
2. Transférez l'image sur l'ordinateur de développement via iPhoto ou toute autre application adaptée.

N'incluez pas de texte dans l'image de lancement si l'application est localisée en plusieurs langues. Etant donné que l'image de lancement est statique, le texte ne serait pas traduit.

Voir aussi

iOS Human Interface Guidelines: Launch images (disponible en anglais uniquement)

Paramètres ignorés

Les applications pour périphériques mobiles ignorent les paramètres d'application relatifs aux fenêtres natives ou aux fonctionnalités du système d'exploitation de bureau. Les paramètres ignorés sont les suivants :

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder

- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

Mise en package d'une application AIR mobile

La commande ADT -package permet de créer le package d'une application AIR pour périphérique mobile. Le paramètre -target spécifie la plate-forme mobile pour laquelle est créé le package.

Packages Android
Les applications AIR for Android utilisent le format de package d'Android (APK), plutôt que le format de package d'AIR.

Le format des packages produits par l'outil ADT avec le type de cible *APK* est adapté à Android Market. Toute application doit impérativement se conformer aux conditions requises d'Android Market pour être acceptée. Passez en revue les conditions requises les plus récentes avant de créer le package final. Voir Android Developers : Publishing on the Market (disponible en anglais uniquement).

Contrairement aux applications iOS, vous pouvez signer l'application par le biais d'un certificat de signature du code AIR standard. Toutefois, pour proposer une application sur Android Market, le certificat doit respecter les règles en vigueur, qui exigent qu'il soit valide au moins jusqu'en 2033. Pour créer un certificat de ce type, utilisez la commande ADT -certificate.

Pour proposer une application à un autre site dont les applications ne doivent pas nécessiter le téléchargement d'AIR à partir d'Android Market, vous pouvez spécifier une autre URL de téléchargement à l'aide du paramètre `-airDownloadURL` de l'outil ADT. Lorsqu'un utilisateur qui ne dispose pas de la version requise du moteur d'exécution d'AIR lance l'application, il est redirigé vers l'URL indiquée. Pour plus d'informations, voir « Commande ADT package » à la page 175.

Par défaut, ADT met en package l'application Android avec le moteur d'exécution partagé. Pour exécuter l'application, l'utilisateur doit donc installer le moteur d'exécution d'AIR séparément sur le périphérique.

Remarque : *Pour forcer ADT à créer un package APK qui utilise un moteur d'exécution captif, utilisez* `target apk-captive-runtime`.

Packages iOS

Sous iOS, les applications AIR utilisent le format de package d'iOS (IPA), plutôt que le format d'AIR natif.

Le format des packages produits par l'outil ADT en définissant le type de cible *ipa-app-store* avec le certificat développeur et le profil de configuration corrects est adapté à l'App Store d'Apple. Utilisez le type de cible *ipa-ad-hoc* pour mettre en package une application à des fins de distribution ad hoc.

Signez l'application avec le certificat de développement délivré par Apple correct. Les certificats requis par la création de versions de test ne correspondent pas aux certificats utilisés pour la mise en package finale avant l'envoi de l'application.

Pour obtenir un exemple de mise en package d'une application iOS avec Ant, voir Piotr Walczyszyn : Packaging AIR application for iOS devices with ADT command and ANT script (disponible en anglais uniquement)

Mise en package avec l'outil ADT

La version 2.6 du kit SDK d'AIR et les versions ultérieures prennent en charge la mise en package d'iOS et d'Android. Avant la mise en package, il est nécessaire de compiler tout code ActionScript, MXML et code d'extension éventuel. Vous devez également disposer d'un certificat de signature du code.

Pour consulter des informations de référence détaillées sur les commandes et options de l'outil ADT, voir « Outil AIR Developer (ADT) » à la page 174.

Packages APK Android

Création d'un package APK

Pour créer un package APK, utilisez la commande ADT package et définissez le type de cible sur *apk* pour les versions validées, sur *apk-debug* pour les versions de débogage et sur *apk-emulator* pour les versions en mode de validation à exécuter sur un émulateur.

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

Saisissez la commande entière sur une ligne unique. Les sauts de ligne que contient l'exemple ci-dessus ont pour unique objet de faciliter la lecture. Par ailleurs, l'exemple part du principe que le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Pour plus d'informations, voir « Variables d'environnement path » à la page 321.)

Vous devez exécuter la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf et un répertoire d'icônes.

Si vous exécutez la commande comme indiqué, l'outil ADT vous invite à entrer le mot de passe associé au keystore. (Les caractères du mot de passe saisis ne sont pas affichés. Appuyez simplement sur Entrée une fois la saisie terminée.)

Remarque : *Par défaut, toutes les applications AIR Android disposent d'*`Air.` *préfixe du nom du package. Pour désactiver ce comportement par défaut, définissez la variable d'environnement* `AIR_NOANDROIDFLAIR` *sur true depuis votre ordinateur.*

Création d'un package APK pour une application faisant appel à des extensions natives

Pour créer un package APK pour une application ayant recours aux extensions natives, ajoutez l'option `-extdir` aux options de mise en package standard. Dans le cas de fichiers ANE qui partagent des ressources/bibliothèques, l'outil ADT sélectionne une seule ressource/bibliothèque et ignore les autres entrées en double avant de générer un avertissement. Cette option spécifie le répertoire contenant les fichiers ANE auxquels fait appel l'application. Exemple :

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

Création d'un package APK incluant sa propre version du moteur d'exécution d'AIR

Pour créer un package APK contenant l'application et une version captive du moteur d'exécution d'AIR, utilisez la cible `apk-captive-runtime`. Cette option spécifie le répertoire contenant les fichiers ANE auxquels fait appel l'application. Exemple :

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

Cette technique peut présenter les inconvénients suivants :

- Principaux correctifs de sécurité publiés par Adobe non automatiquement disponibles aux utilisateurs
- Plus grande empreinte RAM de l'application

Remarque : *lorsque vous intégrez le moteur d'exécution, ADT ajoute les autorisations `INTERNET` et `BROADCAST_STICKY` à votre application. Ces autorisations sont requises par le moteur d'exécution d'AIR.*

Création d'un package APK de débogage

Pour créer une version de l'application à utiliser avec un débogueur, utilisez la cible apk-debug et spécifiez les options de connexion :

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

L'indicateur -connect indique au moteur d'exécution d'AIR du périphérique où se connecter à un débogueur distant via le réseau. Pour effectuer un débogage via USB, vous devez spécifier l'indicateur `-listen` en stipulant le port TCP à utiliser pour la connexion de débogage :

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

Pour que la plupart des fonctionnalités de débogage fonctionnent, vous devez également compiler les fichiers SWF et SWC de l'application en activant le débogage. Voir « Options de connexion au débogueur » à la page 192 pour obtenir une description complète des indicateurs `-connect` et `-listen`.

Remarque : *Par défaut, ADT met en package une copie captive du moteur d'exécution d'AIR avec votre application Android lors de la création du package de l'application avec la cible apk-debug. Pour forcer l'outil ADT à créer un package APK qui utilise un moteur d'exécution externe, définissez la variable d'environnement* `AIR_ANDROID_SHARED_RUNTIME` *sur* `true`.

Sous Android, l'application doit également être autorisée à accéder à Internet pour pouvoir se connecter à l'ordinateur qui exécute le débogueur via le réseau. Voir « Autorisations Android » à la page 80.

Création d'un package APK à utiliser sur un émulateur Android

Vous pouvez utiliser un package de débogage APK sur un émulateur Android, mais non un package en mode de validation. Pour créer un package APK en mode de validation à utiliser sur un émulateur, faites appel à la commande ADT package et définissez le type de cible sur *apk-emulator* :

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

L'exemple part du principe que le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Pour plus d'informations, voir « Variables d'environnement path » à la page 321.)

Création d'un package APK à partir d'un fichier AIR ou AIRI

Vous pouvez créer un package APK directement à partir d'un fichier AIR ou AIRI existant :

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

Le fichier AIR doit utiliser l'espace de noms AIR 2.5 (ou ultérieur) dans le fichier descripteur d'application.

Création d'un package APK pour la plate-forme Android x86

A partir d'AIR 14, l'argument, `-arch`, peut être utilisé pour mettre en package APK pour la plate-forme Android x86. Exemple :

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

Packages iOS

Sous iOS, l'outil ADT convertit le code binaire du fichier SWF et autres fichiers sources en application iOS native.

1 Créez le fichier SWF à l'aide de Flash Builder, de Flash Professional ou d'un compilateur de ligne de commande.

2 Ouvrez une interface de commande ou un terminal et accédez au dossier de projet contenant l'application iPhone.

3 Créez ensuite le fichier IPA à l'aide de l'outil ADT en appliquant la syntaxe suivante :

```
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
```

Modifiez la référence `adt` de sorte à inclure le chemin d'accès complet à l'outil ADT. L'outil ADT est installé dans le sous-répertoire bin du kit SDK d'AIR.

Sélectionnez l'option `-target` correspondant au type d'application iPhone à créer :

- `-target ipa-test` : cette option permet de compiler rapidement une version de l'application en vue de la tester sur l'iPhone de développement. Vous pouvez par ailleurs utiliser `ipa-test-interpreter` pour une compilation encore plus rapide ou `ipa-test-interpreter-simulator` pour une exécution dans le simulateur iOS.
- `-target ipa-debug` : cette option permet de compiler une version de débogage de l'application en vue de la tester sur l'iPhone de développement. Cette option permet de recevoir la sortie `trace()` issue de l'application iPhone dans le cadre d'une session de débogage.

  Vous pouvez inclure l'une des options `-connect` suivantes (`CONNECT_OPTIONS`) pour spécifier l'adresse IP de l'ordinateur de développement qui exécute le débogueur :

  - `-connect` : l'application tentera de se connecter via wi-fi à une session de débogage sur l'ordinateur de développement utilisé pour compiler l'application.
  - `-connect IP_ADDRESS` : l'application tentera de se connecter via wi-fi à une session de débogage sur l'ordinateur dont l'adresse IP a été spécifiée. Exemple :

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

  - `-connect HOST_NAME` : l'application tentera de se connecter via wi-fi à une session de débogage sur l'ordinateur dont le nom d'hôte a été spécifié. Exemple :

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

  L'option `-connect` est facultative. Si elle n'est pas spécifiée, l'application de débogage résultante ne tentera pas de se connecter à un débogueur hébergé. Vous pouvez également spécifier `-listen` au lieu de `-connect` pour activer le débogage via USB (voir « Débogage à distance avec le programme FDB via USB » à la page 112).

  En cas d'échec d'une tentative de connexion à une session de débogage, l'application affiche une boîte de dialogue qui invite l'utilisateur à saisir l'adresse IP de l'ordinateur qui héberge le débogueur. Une tentative de connexion peut échouer si le périphérique n'est pas connecté au réseau Wi-Fi. Elle peut également échouer si le périphérique est connecté, mais pas derrière le pare-feu de l'ordinateur de débogage hôte.

  Vous pouvez également utiliser `ipa-debug-interpreter` pour une compilation plus rapide ou `ipa-debug-interpreter-simulator` pour exécuter l'application dans le simulateur iOS.

  Pour plus d'informations, voir « Débogage d'une application AIR mobile » à la page 105.

- `-target ipa-ad-hoc` : cette option permet de créer une application destinée à un déploiement ad hoc (voir le centre des développeurs iPhone d'Apple).
- `-target ipa-app-store` : cette option permet de créer une version définitive du fichier IPA à déployer sur l'App Store d'Apple.

Remplacez *`PROFILE_PATH`* par le chemin du profil de configuration de l'application. Pour plus d'informations sur les profils de configuration, voir « Configuration d'iOS » à la page 69.

Utilisez l'option `-platformsdk` pour pointer vers le kit du simulateur iOS lorsque vous créez votre application en vue de l'exécuter dans le simulateur iOS.

Remplacez l'élément *`SIGNING_OPTIONS`* de sorte à faire référence au certificat de développement iPhone et au mot de passe correspondant Utilisez la syntaxe suivante :

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

Remplacez *P12_FILE_PATH* par le chemin du fichier de certificat P12. Remplacez *PASSWORD* par le mot de passe associé au certificat (voir l'exemple ci-dessous). Pour plus d'informations sur le fichier du certificat P12, voir « Conversion d'un certificat de développement en fichier de keystore P12 » à la page 209.

Remarque : *vous pouvez utiliser un certificat auto-signé lors de la mise en package pour le simulateur iOS.*

Remplacez l'élément *`APP_DESCRIPTOR`* de sorte à faire référence au fichier descripteur de l'application.

Remplacez l'élément *`SOURCE_FILES`* de sorte à faire référence au principal fichier SWF du projet, suivi de tout autre actif à inclure. Incluez les chemins d'accès à tous les fichiers d'icône définis dans la boîte de dialogue des paramètres de l'application de Flash Professional ou dans un fichier descripteur d'application personnalisé. Ajoutez également le fichier contenant les graphiques de l'écran initial, Default.png.

Utilisez l'option `-extdir` *`extension-directory`* pour spécifier le répertoire contenant les fichiers ANE (extensions natives) qu'utilise l'application. Si l'application ne fait pas appel à des extensions natives, n'incluez pas cette option.

Important : *ne créez pas de sous-répertoire dans le répertoire de votre application appelé `Resources`. Le moteur d'exécution crée automatiquement un dossier avec ce nom pour respecter la structure du package IPA. La création de votre propre dossier de ressources entraîne un conflit fatal.*

Création d'un package iOS à des fins de débogage

Pour créer un package iOS à installer sur des périphériques de test, utilisez la commande ADT package et définissez le type de cible sur *ios-debug*. Avant d'exécuter cette commande, vous devez obtenir un profil de configuration de développement et un certificat développeur auprès d'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
```

Remarque : *Vous pouvez également utiliser `ipa-debug-interpreter` pour une compilation plus rapide ou `ipa-debug-interpreter-simulator` pour exécuter l'application dans le simulateur iOS*

Saisissez la commande entière sur une ligne unique. Les sauts de ligne que contient l'exemple ci-dessus ont pour unique objet de faciliter la lecture. Par ailleurs, l'exemple part du principe que le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Pour plus d'informations, voir « Variables d'environnement path » à la page 321.)

Vous devez exécuter la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf, un répertoire d'icônes et le fichier Default.png.

Vous devez signer l'application à l'aide du certificat de distribution correct délivré par Apple. Il est impossible d'utiliser tout autre certificat développeur.

Utilisez l'option `-connect` pour le débogage wi-fi. L'application tente de lancer une session de débogage lorsque le débogueur Flash (FDB) est en cours d'exécution sur l'adresse IP ou le nom d'hôte spécifié. Utilisez l'option `-listen` pour le débogage USB. Démarrez tout d'abord l'application, puis démarrez FDB, qui lance une session de débogage pour l'application en cours d'exécution. Pour plus d'informations, voir « Connexion au débogueur Flash » à la page 110.

Création d'un package iOS à envoyer à l'App Store d'Apple

Pour créer un package iOS à envoyer à l'App Store d'Apple, utilisez la commande ADT package et définissez le type de cible sur *ios-app-store*. Avant d'exécuter cette commande, vous devez obtenir un profil de configuration et un certificat de signature du code de distribution auprès d'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
```

Saisissez la commande entière sur une ligne unique. Les sauts de ligne que contient l'exemple ci-dessus ont pour unique objet de faciliter la lecture. Par ailleurs, l'exemple part du principe que le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Pour plus d'informations, voir « Variables d'environnement path » à la page 321.)

Vous devez exécuter la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf, un répertoire d'icônes et le fichier Default.png.

Vous devez signer l'application à l'aide du certificat de distribution correct délivré par Apple. Il est impossible d'utiliser tout autre certificat développeur.

Important : *Apple requiert que vous utilisiez le programme Application Loader d'Apple pour télécharger une application sur l'App Store. Apple publie Application Loader pour Mac OS X uniquement. Par conséquent, bien que vous puissiez développer une application AIR pour l'iPhone sur un ordinateur Windows, vous devez disposer d'un ordinateur qui exécute OS X (version 10.5.3 ou ultérieure) pour envoyer l'application à l'App Store. Le programme Application Loader est disponible sur le centre de développement iOS d'Apple.*

Création d'un package iOS destiné à une distribution ad hoc

Pour créer un package iOS destiné à une distribution ad hoc, utilisez la commande ADT package et définissez le type de cible sur *ios-ad-hoc*. Avant d'exécuter cette commande, vous devez obtenir un profil de configuration et un certificat de signature du code de distribution ad hoc auprès d'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
```

Saisissez la commande entière sur une ligne unique. Les sauts de ligne que contient l'exemple ci-dessus ont pour unique objet de faciliter la lecture. Par ailleurs, l'exemple part du principe que le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Pour plus d'informations, voir « Variables d'environnement path » à la page 321.)

Vous devez exécuter la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf, un répertoire d'icônes et le fichier Default.png.

Vous devez signer l'application à l'aide du certificat de distribution correct délivré par Apple. Il est impossible d'utiliser tout autre certificat développeur.

Création d'un package iOS pour une application faisant appel à des extensions natives

Pour créer un package iOS pour une application faisant appel à des extensions natives, utilisez la commande du package ADT avec l'option `-extdir`. Utilisez la commande ADT correspondant à la cible (`ipa-app-store`, `ipa-debug`, `ipa-ad-hoc`, `ipa-test`). Exemple :

```
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
```

Saisissez la commande entière sur une ligne unique. Les sauts de ligne que contient l'exemple ci-dessus ont pour unique objet de faciliter la lecture.

En ce qui concerne les extensions natives, l'exemple ci-dessus suppose que le répertoire appelé `extensionsDir` est le répertoire dans lequel vous exécutez la commande. Le répertoire `extensionsDir` contient les fichiers ANE auxquels fait appel l'application.

Débogage d'une application AIR mobile

Vous disposez de plusieurs méthodes pour déboguer une application AIR mobile. La méthode la plus simple pour mettre en évidence les problèmes de logique d'une application consiste à procéder au débogage sur l'ordinateur de développement par le biais de l'application ADL ou le simulateur iOS. Vous pouvez également installer l'application sur un périphérique et la déboguer à distance à l'aide du débogueur Flash, qui s'exécute sur un ordinateur de bureau.

Simulation de périphérique à l'aide de l'application ADL

La méthode la plus rapide et la plus simple pour tester et déboguer la plupart des fonctionnalités d'une application mobile consiste à exécuter cette dernière sur l'ordinateur de développement à l'aide de l'application de débogage du lanceur AIR (ADL). L'application ADL utilise l'élément `supportedProfiles` du descripteur d'application pour déterminer le profil à utiliser. Si plusieurs profils sont recensés, l'application ADL utilise le premier de la liste. Vous disposez également du paramètre `-profile` de l'application ADL pour sélectionner l'un des autres profils de la liste `supportedProfiles`. (Si le descripteur d'application ne contient pas d'élément `supportedProfiles`, vous pouvez spécifier n'importe quel profil dans l'argument `-profile`.) Utilisez par exemple la commande suivante pour lancer une application en vue de simuler le profil de périphérique mobile :

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

Si vous adoptez cette méthode pour simuler le profil mobile sur le bureau, l'application s'exécute dans un environnement plus proche d'un périphérique mobile cible. Les API ActionScript qui ne font pas partie du profil mobile ne sont pas disponibles. L'application ADL n'établit toutefois pas de distinction entre les fonctionnalités de différents périphériques mobiles. Vous pouvez, par exemple, envoyer des simulations d'utilisation de touche programmable à l'application, même si le périphérique cible réel n'utilise pas de touches programmables.

L'application ADL prend en charge les simulations de changement d'orientation d'un périphérique et la saisie par touches programmables par le biais de commandes de menu. Si vous l'exécutez dans le profil de périphérique mobile, elle affiche un menu (dans la fenêtre de l'application ou la barre de menus du bureau) qui permet d'entrer les données de rotation du périphérique ou de touche programmable.

Touches programmables

L'application ADL simule les boutons de touche programmable Retour, Menu et Rechercher d'un périphérique mobile. Vous pouvez envoyer ces touches au périphérique simulé à l'aide du menu qui s'affiche au lancement de l'application ADL par le biais du profil mobile.

Rotation du périphérique

L'application ADL permet de simuler la rotation d'un périphérique à l'aide du menu qui s'affiche au lancement d'ADL par le biais du profil mobile. Vous pouvez faire pivoter le périphérique simulé vers la droite ou la gauche.

La simulation de la rotation n'affecte qu'une application qui gère l'orientation automatique. Pour activer cette fonctionnalité, définissez l'élément `autoOrients` sur `true` dans le descripteur d'application.

Taille de l'écran

Vous pouvez tester l'application sur des écrans de diverses tailles en définissant le paramètre ADL `-screensize`. Vous pouvez transmettre le code associé à l'un des types d'écran prédéfinis ou une chaîne qui contient les quatre valeurs représentant les dimensions en pixels des écrans de taille normale et agrandie.

Veillez à toujours spécifier les dimensions (en pixels) correspondant au format Portrait, c'est-à-dire de spécifier une valeur de largeur inférieure à la valeur de hauteur. La commande suivante permet, par exemple, d'ouvrir l'application ADL pour simuler l'écran du Droid Motorola :

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Pour consulter la liste des types d'écran prédéfinis, voir « Utilisation de l'application ADL » à la page 168.

Restrictions

L'application ADL ne peut pas simuler certaines API qui ne sont pas prises en charge par le profil de bureau, à savoir :

- Accelerometer

- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- Interactions tactiles multipoints et mouvements sur les systèmes d'exploitation de bureau qui ne prennent pas en charge ces fonctionnalités
- SystemIdleMode

Si l'application utilise ces classes, il est recommandé de tester ces fonctionnalités sur un émulateur ou un périphérique réel.

De même, certaines API fonctionnent si elles s'exécutent sous l'application ADL sur le bureau, mais sont réservées à des types de périphériques mobiles déterminés. Parmi ces API figurent :

- Codec audio AAC et Speex
- Prise en charge de l'accessibilité et des logiciels de lecture d'écran
- RTMPE
- Chargement de fichiers SWF contenant du pseudo-code ActionScript
- Shaders PixelBender

Veillez à tester les applications qui font appel à ces fonctionnalités sur les périphériques cibles, car l'application ADL ne réplique pas entièrement l'environnement d'exécution.

Simulation de périphérique avec le simulateur iOS

Le simulateur iOS (Mac uniquement) offre un moyen rapide d'exécuter et de déboguer des applications iOS. Lorsque vous testez l'application avec le simulateur iOS, il n'est pas nécessaire d'obtenir un certificat de développeur ou un profil d'attribution de privilèges d'accès. Vous devez tout de même créer un certificat p12, bien qu'il puisse être auto-signé.

Par défaut, ADT lance toujours le simulateur iPhone. Pour modifier le périphérique de simulation, procédez comme suit :

- Utilisez la commande ci-dessous pour afficher les simulateurs disponibles.

  ```
  xcrun simctl list devices
  ```

  La sortie ressemble à celle présentée ci-dessous.

```
== 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)
....
```

- Vous pouvez sélectionner un simulateur spécifique en définissant la variable d'environnement `AIR_IOS_SIMULATOR_DEVICE` comme suit :

```
export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
```

Redémarrez le processus après la définition de la variable d'environnement et exécutez l'application sur le périphérique de simulation de votre choix.

Remarque : *lorsque vous utilisez ADT avec le simulateur iOS, vous devez toujours inclure l'option `-platformsdk` en spécifiant le chemin vers le kit SDK du simulateur iOS.*

Pour exécuter une application dans le simulateur iOS :

1. Utilisez la commande adt -package soit avec `-target ipa-test-interpreter-simulator` soit avec `-target ipa-debug-interpreter-simulator`, comme dans l'exemple suivant :

```
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
```

   Remarque : *Les options de signature ne sont désormais plus requises dans le cas des simulateurs. Par conséquent, toute valeur peut être fournie dans l'indicateur `-keystore`, car elle n'est pas prise en charge par ADT.*

2. Utilisez la commande adt -installApp pour installer l'application dans le simulateur iOS, comme dans l'exemple suivant :

```
adt -installApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -package sample_ipa_name.ipa
```

3. Utilisez la commande adt -launchApp pour exécuter l'application dans le simulateur iOS, comme dans l'exemple suivant :

   Remarque : *Par défaut, la commande `adt -launchApp` exécute l'application dans le simulateur iPhone. Pour exécuter l'application dans le simulateur iPad, exportez la variable d'environnement, `AIR_IOS_SIMULATOR_DEVICE` = iPad, puis exécutez la commande `adt -launchApp`.*

```
adt -launchApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -appid sample_ipa_name
```

Pour tester une extension native dans le simulateur iOS, utilisez le nom de plate-forme `iPhone-x86` dans le fichier extension.xml, puis spécifiez `library.a` (bibliothèque statique) dans l'élément `nativeLibrary`, comme dans l'exemple extension.xml suivant :

```
<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>
```

Remarque : *lorsque vous testez une extension native dans le simulateur iOS, n'utilisez pas la bibliothèque statique (fichier* `.a`*) compilée pour le périphérique. Utilisez plutôt la bibliothèque statique compilée pour le simulateur.*

Instructions trace

Si vous exécutez l'application mobile sur le bureau, la sortie trace est imprimée sur le débogueur ou la fenêtre du terminal utilisée pour lancer l'application ADL. Si vous exécutez l'application sur un périphérique ou un émulateur, vous pouvez configurer une session de débogage à distance pour afficher la sortie trace. Si cette fonctionnalité est prise en charge, vous pouvez également afficher la sortie trace à l'aide des outils de développement logiciel intégrés au périphérique ou au système d'exploitation.

Dans tous les cas, vous devez compiler les fichiers SWF de l'application en activant le débogage pour que le moteur d'exécution puisse générer les instructions trace.

Instructions trace distantes sous Android

Si vous exécutez une application sur un émulateur ou périphérique Android, vous pouvez afficher la sortie d'instructions trace dans le journal système à l'aide de l'utilitaire Debug Bridge (ADB) intégré au kit SDK d'Android. Pour afficher la sortie de l'application, exécutez la commande suivante à partir d'une invite de commande ou d'une fenêtre de terminal sur l'ordinateur de développement :

```
tools/adb logcat air.MyApp:I *:S
```

où *MyApp* correspond à l'ID de l'application AIR. L'argument `*:S` supprime la sortie de tout autre processus. Pour afficher les informations système relatives à l'application parallèlement à la sortie trace, vous pouvez inclure ActivityManager dans la spécification du filtre logcat :

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

Ces exemples de commande considèrent comme acquis que vous exécutez l'utilitaire ADB à partir du dossier du kit SDK d'Android ou que vous avez ajouté le dossier `SDK` à la variable d'environnement path.

Remarque : *dans AIR 2.6+, l'utilitaire ADB est intégré au kit SDK d'AIR et réside dans le dossier lib/android/bin.*

Instructions trace distantes sous iOS

Pour afficher la sortie des instructions trace issues d'une application qui s'exécute sur un périphérique iOS, vous devez lancer une session de débogage à distance à l'aide de l'utilitaire Flash Debugger (FDB).

Voir aussi

Android Debug Bridge: Enable logcat Logging (disponible en anglais uniquement)

« Variables d'environnement path » à la page 321

Connexion au débogueur Flash

Pour déboguer une application qui s'exécute sur un périphérique mobile, vous pouvez exécuter le débogueur Flash sur l'ordinateur de développement et vous y connecter via le réseau. Pour activer le débogage à distance, procédez comme suit :

- Sous Android, spécifiez l'autorisation android:permission.INTERNET dans le descripteur d'application.
- Compilez les fichiers SWF de l'application en activant le débogage.
- Mettez en package l'application avec `-target apk-debug (pour Android)` ou `-target ipa-debug` (pour iOS), et soit l'indicateur `-connect` (débogage wi-fi) soit l'indicateur `-listen` (débogage USB).

Pour un débogage à distance via wi-fi, le périphérique doit être en mesure d'accéder au port TCP 7935 de l'ordinateur qui exécute le débogueur Flash par le biais de l'adresse IP ou du nom de domaine complet. Pour un débogage via USB, le périphérique doit être en mesure d'accéder au port TCP 7936 ou au port spécifié dans l'indicateur `-listen`.

Pour iOS, vous pouvez également spécifier `-target ipa-debug-interpreter` ou `-target ipa-debug-interpreter-simulator`.

Débogage à distance avec Flash Professional

Lorsque vous êtes prêt à déboguer l'application et que les autorisations sont définies dans le descripteur d'application, procédez comme suit :

1. Ouvrez la boîte de dialogue Paramètres AIR for Android.
2. Sous l'onglet Déploiement :
   - Sélectionnez le type de déploiement « Déboguer ».
   - Sélectionnez l'option « Installer l'application sur le périphérique Android raccordé » dans la zone de groupe Après la publication.
   - Désélectionnez l'option « Lancer l'application sur le périphérique Android raccordé » dans la zone de groupe Après la publication.
   - Le cas échéant, définissez le chemin d'accès au kit SDK d'Android.
3. Cliquez sur Publier.

   L'application est installée et lancée sur le périphérique.
4. Fermez la boîte de dialogue Paramètres AIR for Android.
5. Sélectionnez Déboguer > Commencer la session de débogage à distance > ActionScript 3 dans le menu Flash Professional.

   Flash Professional affiche le message « En attente de la connexion du lecteur » dans le panneau Sortie.
6. Lancez l'application sur le périphérique.
7. Entrez l'adresse IP ou le nom d'hôte de l'ordinateur qui exécute le débogueur Flash dans la boîte de dialogue de connexion Adobe AIR, puis cliquez sur OK.

Débogage à distance avec le programme FDB via une connexion réseau

Pour déboguer une application qui s'exécute sur un périphérique à l'aide du programme de ligne de commande Flash Debugger (FDB), commencez par exécuter ce dernier sur l'ordinateur de développement, puis démarrez l'application sur le périphérique. Les procédures ci-dessous font appel aux outils AMXMLC, FDB et ADT pour compiler, mettre en package et déboguer une application sur le périphérique. Les exemples partent du principe que vous utilisez un kit SDK Flex et AIR combiné et que le répertoire bin figure dans la variable d'environnement path. (Cette hypothèse a pour unique objet de simplifier les exemples de commande.)

1 Ouvrez une fenêtre d'invite de commande ou de terminal et accédez au répertoire qui contient le code source de l'application.

2 Compilez l'application avec amxmlc en activant le débogage :

```
amxmlc -debug DebugExample.as
```

3 Mettez en package l'application en définissant la cible `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
```

Si vous utilisez toujours le même nom d'hôte ou la même adresse IP pour le débogage, vous pouvez saisir cette valeur après l'indicateur `-connect`. L'application tente alors automatiquement de se connecter à ce nom d'hôte ou cette adresse IP. Si tel n'est pas le cas, entrez les informations sur le périphérique à chaque session de débogage.

4 Installez l'application.

Sous Android, vous disposez de la commande ADT `-installApp` :

```
adt -installApp -platform android -package DebugExample.apk
```

Sous iOS, vous pouvez installer l'application à l'aide de la commande `-installApp` de l'outil ADT ou via iTunes.

5 Dans une seconde fenêtre de commande ou de terminal, exécutez le programme FDB :

```
fdb
```

6 Dans la fenêtre du programme FDB, saisissez la commande `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 Lancez l'application sur le périphérique.

8 Lorsque l'application est lancée sur le périphérique ou l'émulateur, la boîte de dialogue de connexion d'Adobe AIR s'ouvre. (Si vous avez spécifié un nom d'hôte ou une adresse IP avec l'option -connect lors de la mise en package de l'application, elle tente automatiquement de se connecter à cette adresse.) Entrez l'adresse appropriée et touchez OK.

Pour établir une connexion au débogueur dans ce mode, le périphérique doit pouvoir résoudre l'adresse ou le nom d'hôte et se connecter au port TCP 7935. Il est nécessaire de disposer d'une connexion au réseau.

9 Lorsque le moteur d'exécution distant se connecte au débogueur, vous pouvez définir les points d'arrêt avec la commande FDB `break`, puis démarrer l'exécution avec la commande `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
```

Débogage à distance avec le programme FDB via USB

AIR 2.6 (Android) AIR 3.3 (iOS)

Pour déboguer une application mobile via une connexion USB, mettez en package l'application en utilisant l'option `-listen` au lieu de l'option `-connect`. Si vous spécifiez l'option `-listen`, le moteur d'exécution écoute une connexion provenant du débogueur Flash (FDB) sur le port TCP 7936 au démarrage de l'application. Exécutez alors FDB avec l'option `-p` ; FDB établit la connexion.

Procédure de débogage USB pour Android

Pour que le débogueur Flash qui s'exécute sur l'ordinateur de bureau se connecte au moteur d'exécution d'AIR qui s'exécute sur le périphérique ou l'émulateur, vous devez utiliser l'utilitaire Android Debug Bridge (ADB, intégré au kit SDK d'Android) ou l'utilitaire iOS Debug Bridge (IDB, intégré au kit SDK d'AIR) pour transmettre le port du périphérique au port de l'ordinateur de bureau.

1. Ouvrez une fenêtre d'invite de commande ou de terminal et accédez au répertoire qui contient le code source de l'application.
2. Compilez l'application avec amxmlc en activant le débogage :

   ```
   amxmlc -debug DebugExample.as
   ```

3. Mettez en package l'application à l'aide de la cible de débogage appropriée (telle que `apk-debug`), puis spécifiez l'option `-listen` :

   ```
   adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12
   DebugExample.apk DebugExample-app.xml DebugExample.swf
   ```

4. Connectez le périphérique à l'ordinateur de débogage via un câble USB. (Cette procédure permet également de déboguer une application qui s'exécute sur un émulateur, auquel cas la connexion USB est superflue — et impossible.)
5. Installez l'application.

   Vous pouvez utiliser la commande ADT `-installApp` :

   ```
   adt -installApp -platform android -package DebugExample.apk
   ```

6. Transmettez le port TCP 7936 du périphérique ou de l'émulateur à l'ordinateur de bureau par le biais de l'utilitaire Android ADB :

   ```
   adb forward tcp:7936 tcp:7936
   ```

7. Lancez l'application sur le périphérique.
8. Dans une fenêtre de terminal ou de commande, exécutez FDB à l'aide de l'option -p :

   ```
   fdb -p 7936
   ```

9 Dans la fenêtre du programme FDB, saisissez la commande `run` :

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 L'utilitaire FDB tente de se connecter à l'application.

11 Une fois la connexion distante établie, vous pouvez définir les points d'arrêt avec la commande FDB `break`, puis démarrer l'exécution avec la commande `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
```

Remarque : *le moteur d'exécution d'AIR et FDB utilisent par défaut le port 7936 pour le débogage. Vous pouvez spécifier l'utilisation d'autres ports avec le paramètre de port ADT -listen et le paramètre de port FDB -p. Si tel est le cas, vous devez transférer le numéro de port spécifié dans l'outil ADT au port spécifié dans FDB par le biais de l'utilitaire Android Debug Bridge :* `adb forward tcp:adt_listen_port# tcp:fdb_port#`

Procédure de débogage USB pour iOS

Pour que le débogueur Flash qui s'exécute sur l'ordinateur de bureau se connecte au moteur d'exécution d'AIR qui s'exécute sur le périphérique ou l'émulateur, vous devez utiliser l'utilitaire iOS Debug Bridge (IDB, intégré au kit SDK d'AIR) pour transmettre le port du périphérique au port de l'ordinateur de bureau.

1 Ouvrez une fenêtre d'invite de commande ou de terminal et accédez au répertoire qui contient le code source de l'application.

2 Compilez l'application avec amxmlc en activant le débogage :

```
amxmlc -debug DebugExample.as
```

3 Mettez en package l'application à l'aide de la cible de débogage appropriée (telle que `ipa-debug` ou `ipa-debug-interpreter`), puis spécifiez l'option `-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 Connectez le périphérique à l'ordinateur de débogage via un câble USB. (Cette procédure permet également de déboguer une application qui s'exécute sur un émulateur, auquel cas la connexion USB est superflue — et impossible.)

5 Installez et lancez l'application sur le périphérique iOS. Dans AIR 3.4 et les versions ultérieures, vous pouvez utiliser `adt -installApp` pour installer l'application via USB.

6 Déterminez le handle de périphérique à l'aide de la commande `idb -devices` (IDB est situé dans le dossier *`air_sdk_root`*`/lib/aot/bin/iOSBin/idb`) :

```
./idb -devices
                                List of attached devices
                                Handle    UUID
                                    1     91770d8381d12644df91fbcee1c5bbdacb735500
```

Remarque : *(AIR 3.4 et versions ultérieures) vous pouvez utiliser* `adt -devices` *au lieu de* `idb -devices` *pour déterminer le handle de périphérique.*

7 Transmettez un port sur votre bureau au port spécifié dans le paramètre `adt -listen` (dans ce cas, 16000 ; le port par défaut est 7936) à l'aide de l'utilitaire IDB et de l'ID de périphérique indiqué à l'étape précédente :

```
idb -forward 7936 16000 1
```

Dans cet exemple, 7936 est le port de bureau, 16000 est le port sur lequel écoute le périphérique connecté et 1 est l'ID de périphérique du périphérique connecté.

8 Dans une fenêtre de terminal ou de commande, exécutez FDB à l'aide de l'option -p :

```
fdb -p 7936
```

9 Dans la fenêtre du programme FDB, saisissez la commande `run` :

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                (fdb) run
```

10 L'utilitaire FDB tente de se connecter à l'application.

11 Une fois la connexion distante établie, vous pouvez définir les points d'arrêt avec la commande FDB `break`, puis démarrer l'exécution avec la commande `continue` :

Remarque : *le moteur d'exécution d'AIR et FDB utilisent par défaut le port 7936 pour le débogage. Vous pouvez spécifier l'utilisation d'autres ports avec le paramètre de port -listen d'IDB et le paramètre de port -p de FDB.*

Installation d'AIR et d'applications AIR sur un périphérique mobile

Les utilisateurs finaux de l'application peuvent installer le moteur d'exécution et les applications AIR par le biais du mécanisme d'installation et de distribution d'application standard du périphérique.

Sous Android, par exemple, les utilisateurs peuvent installer une application à partir d'Android Market. S'ils ont autorisé l'installation d'applications à partir de sources inconnues dans les paramètres Applications, ils peuvent installer une application en cliquant sur un lien dans une page Web ou en copiant le package de l'application sur le périphérique (il suffit alors d'ouvrir le package de l'application). Si un utilisateur tente d'installer une application Android, mais que le moteur d'exécution d'AIR n'est pas encore installé, il accède automatiquement à Android Market, d'où il peut installer le moteur d'exécution.

Sous iOS, les utilisateurs finaux disposent de deux mécanismes de distribution d'application. La voie de distribution principale correspond à l'App Store d'Apple. Une distribution ad hoc permet d'autoriser un nombre limité d'utilisateurs à installer l'application sans transiter par l'App Store.

Installation du moteur d'exécution et des applications AIR à des fins de développement

Etant donné qu'une application AIR est installée sur un périphérique mobile en tant que package natif, vous disposez des fonctionnalités standard d'installation d'applications à tester de la plate-forme. Si elles sont prises en charge, vous pouvez utiliser les commandes ADT pour installer le moteur d'exécution et les applications AIR. Cette approche est actuellement prise en charge sous Android.

Sous iOS, vous pouvez installer une application à tester via iTunes. Les applications de test doivent être signées par un certificat développeur Apple délivré spécifiquement à des fins de développement et mises en package avec un profil de configuration pour le développement d'applications. Sous iOS, une application AIR est un package autonome. Elle ne requiert pas de moteur d'exécution distinct.

Installation d'une application AIR à l'aide de l'outil ADT

Si vous développez des applications AIR, vous disposez de l'outil ADT pour installer et désinstaller tant le moteur d'exécution que les applications. (L'IDE intègre parfois ces commandes, vous évitant ainsi d'exécuter vous-même l'outil ADT.)

L'outil ADT d'AIR permet d'installer le moteur d'exécution d'AIR sur un périphérique ou un émulateur. Veillez à installer au préalable le kit SDK associé au périphérique. Utilisez la commande `-installRuntime` :

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

Si vous ne spécifiez pas le paramètre `-package`, le package du moteur d'exécution adapté au périphérique ou à l'émulateur est sélectionné parmi les options disponibles dans le SDK d'AIR installé.

Pour installer une application AIR sur Android ou iOS (AIR 3.4 et les versions ultérieures), utilisez la commande `-installApp` similaire :

```
adt -installApp -platform android -device deviceID -package path-to-app
```

La valeur définie de l'argument `-platform` doit correspondre au périphérique sur lequel vous installez l'application.

Remarque : *vous devez supprimer toute version existante du moteur d'exécution ou de l'application AIR avant de procéder à une nouvelle installation.*

Installation d'une application AIR sur un périphérique iOS via iTunes

Pour installer une application AIR sur un périphérique iOS à des fins de test :

1. Ouvrez l'application iTunes.
2. Le cas échéant, ajoutez à iTunes le profil de configuration associé à l'application. Dans iTunes, sélectionnez Fichier > Ajouter le fichier à la bibliothèque. Sélectionnez ensuite le fichier du profil de configuration (dont le type de fichier correspond à mobileprovision).
3. Certaines versions d'iTunes ne remplacent pas l'application si une version identique de l'application est déjà installée. Dans ce cas de figure, supprimez l'application du périphérique et de la liste d'applications dans iTunes.
4. Double-cliquez sur le fichier IPA associé à l'application. Elle devrait apparaître dans la liste d'applications d'iTunes.
5. Connectez le périphérique au port USB de l'ordinateur.
6. Dans iTunes, vérifiez sur l'onglet Application associé au périphérique que l'application est sélectionnée dans la liste d'applications à installer.
7. Sélectionnez le périphérique dans la liste de gauche d'applications. Cliquez ensuite sur le bouton Synchroniser. Une fois la synchronisation terminée, l'application Hello World apparaît sur l'iPhone.

Si la nouvelle version n'est pas installée, supprimez-la du périphérique et de la liste d'applications dans iTunes, puis répétez la procédure. Ce cas de figure se produit parfois si la version actuellement installée utilise le même ID et le même numéro.

Voir aussi

« Commande ADT installRuntime » à la page 186

« Commande ADT installApp » à la page 184

Exécution d'une application AIR sur un périphérique

Vous pouvez lancer une application AIR installée via l'interface utilisateur du périphérique. Si cette fonctionnalité est prise en charge, vous pouvez également lancer une application à distance à l'aide de l'outil ADT AIR :

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

La valeur de l'argument `-appid` doit correspondre à l'ID de l'application AIR à lancer. Utilisez la valeur spécifiée dans le descripteur de l'application AIR (sans le préfixe *air.* ajouté lors de la mise en package).

Si un seul périphérique ou émulateur est connecté et en cours d'exécution, vous pouvez omettre l'indicateur `-device`. La valeur définie de l'argument `-platform` doit correspondre au périphérique sur lequel vous installez l'application. L'unique valeur prise en charge à l'heure actuelle est *android*.

Suppression du moteur d'exécution et d'une application AIR

Vous disposez des techniques standard de suppression d'applications proposées par le système d'exploitation du périphérique. S'il est pris en charge, vous pouvez également faire appel à l'outil ADT pour supprimer le moteur d'exécution et les applications AIR. Pour supprimer le moteur d'exécution, utilisez la commande `-uninstallRuntime` :

```
adt -uninstallRuntime -platform android -device deviceID
```

Pour désinstaller une application, utilisez la commande `-uninstallApp` :

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Si un seul périphérique ou émulateur est connecté et en cours d'exécution, vous pouvez omettre l'indicateur `-device`. La valeur définie de l'argument `-platform` doit correspondre au périphérique sur lequel vous installez l'application. L'unique valeur prise en charge à l'heure actuelle est *android*.

Configuration d'un émulateur

Pour exécuter l'application AIR sur un émulateur de périphérique, vous devez généralement faire appel au kit SDK associé au périphérique pour créer et exécuter une occurrence d'émulateur sur l'ordinateur de développement. Vous pouvez ensuite installer la version pour émulateur du moteur d'exécution d'AIR et l'application AIR sur l'émulateur. Notez que les applications s'exécutent généralement considérablement plus lentement sur un émulateur que sur un périphérique réel.

Création d'un émulateur Android

1 Lancez le kit SDK d'Android et l'application AVD Manager :

- Sous Windows, exécutez le fichier Setup.exe du kit SDK dans la racine du répertoire du kit SDK d'Android.
- Sous Mac OS, exécutez l'application Android dans le sous-répertoire tools du répertoire du kit SDK d'Android.

2 Sélectionnez l'option Settings, puis l'option « Force https:// ».

3 Sélectionnez l'option Available Packages. La liste des kits SDK d'Android disponibles devrait s'afficher.

4 Sélectionnez un kit SDK d'Android compatible (Android 2.3 ou ultérieur), puis cliquez sur le bouton Install Selected.

5 Sélectionnez l'option Virtual Devices et cliquez sur le bouton New.

6 Définissez les paramètres suivants :

- Nom du périphérique virtuel
- API cible, telle qu'Android 2.3, API niveau 8
- Taille de la carte SD (1 024, par exemple)
- Enveloppe (Default HVGA, par exemple)

7 Cliquez sur le bouton Create AVD.

Notez que, selon la configuration du système, la création du périphérique virtuel risque de prendre un certain temps.

Vous pouvez à présent lancer le nouveau périphérique virtuel.

1 Sélectionnez Virtual Device dans l'application AVD Manager. Le périphérique virtuel que vous venez de créer devrait figurer dans la liste.

2 Sélectionnez le périphérique virtuel, puis cliquez sur le bouton Start.

3 Cliquez sur le bouton Launch sur l'écran suivant.

Une fenêtre d'émulateur devrait s'ouvrir sur le bureau. Ce processus risque de prendre quelques secondes. L'initialisation du système d'exploitation Android risque également de prendre un moment. Vous pouvez installer sur un émulateur les applications mises en package à l'aide des options *apk-debug* et *apk-emulator*. Les applications mises en package avec la cible *apk* ne fonctionnent pas sur un émulateur.

Voir aussi

http://developer.android.com/guide/developing/tools/othertools.html#android

http://developer.android.com/guide/developing/tools/emulator.html

Mise à jour des applications AIR mobiles

Les applications AIR mobiles sont distribuées sous forme de packages natifs et utilisent par conséquent les mécanismes de mise à jour standard des autres applications de la plate-forme. Ce processus implique généralement l'envoi de la mise à jour au site Market Place ou App Store de distribution de l'application d'origine.

Les applications AIR mobiles ne peuvent pas utiliser la structure ou la classe Updater d'AIR.

Mise à jour des applications AIR for Android

Vous pouvez mettre à jour une application distribuée via Android Market en plaçant une nouvelle version sur ce dernier, sous réserve de satisfaire aux exigences suivantes (imposées par Android Market, plutôt qu'AIR) :

- Le package APK est signé par le même certificat.
- L'ID AIR est identique.
- La valeur `versionNumber` du descripteur d'application est supérieure. (Le cas échéant, vous devez également incrémenter la valeur `versionLabel`.)

Les utilisateurs qui ont téléchargé l'application à partir d'Android Market sont avertis par le logiciel du périphérique qu'une mise à jour est disponible.

Voir aussi

Android Developers : Publishing Updates on Android Market (disponible en anglais uniquement)

Mise à jour des applications AIR sous iOS

Vous pouvez mettre à jour les applications AIR distribuées via iTunes en envoyant la mise à jour à ce dernier, sous réserve de satisfaire aux exigences suivantes (ces règles sont imposées par l'App Store d'Apple, et non par AIR) :

- Le certificat développeur et les profils de configuration sont délivrés pour le même ID Apple.
- Le package IPA utilise le même identifiant d'application (suffixe) Apple.
- La mise à jour ne réduit pas le parc de périphériques pris en charge (en d'autres termes, si l'application d'origine prend en charge les périphériques qui exécutent iOS 3, il est impossible de créer une mise à jour qui ne prend pas en charge iOS 3).

Important : *étant donné que le kit SDK d'AIR 2.6 et les versions ultérieures ne prennent pas en charge iOS 3, contrairement à AIR 2, il est impossible de mettre à jour les applications iOS publiées ayant été déployées avec AIR 2 par le biais d'une mise à jour développée avec AIR 2.6+.*

Utilisation des notifications Push

Grâce aux notifications Push, les fournisseurs de notifications à distance peuvent envoyer des notifications aux applications qui s'exécutent sur des périphériques mobiles. AIR 3.4 prend en charge les notifications pour les périphériques iOS qui font appel au service Apple Push Notification (APN).

Remarque : *pour activer les notifications Push dans une application AIR for Android, utilisez une extension native, telle que as3c2dm, développée par l'expert Adobe Piotr Walczyszyn.*

Le reste de cette section décrit comment activer les notifications Push dans les applications AIR for iOS.

Remarque : *nous supposons que vous disposez d'un ID de développement Apple, que vous connaissez le processus de développement iOS et que vous avez déployé au moins une application sur un périphérique iOS.*

Présentation des notifications Push

Le service Apple Push Notification (APN) permet aux fournisseurs de notifications à distance d'envoyer des notifications aux applications qui s'exécutent sur des périphériques iOS. Le service APN prend en charge les types de notification suivants :

- Alertes
- Badges
- Sons

Pour obtenir des informations complètes sur le service APN, consulter le site developer.apple.com.

L'utilisation des notifications Push dans votre application implique plusieurs aspects :

- **Application cliente** : souscrit aux notifications Push, communique avec les fournisseurs de notifications à distance et reçoit des notifications Push.

- **iOS** : gère l'interaction entre l'application cliente et le service APN.
- **Service APN** : fournit un ID de jeton lors de la souscription du client et transmet les notifications des fournisseurs de notifications à distance au périphérique iOS.
- **Fournisseur de notifications à distance** : stocke les informations de l'application cliente (fournie par l'ID de jeton) et envoie des notifications Push au service APN.

Procédure de souscription

La procédure de souscription aux notifications Push avec un service côté serveur est la suivante :

1. L'application cliente demande au périphérique iOS d'activer les notifications Push.
2. iOS transfère la demande au service APN.
3. Le serveur du service APN renvoie un ID de jeton au périphérique iOS.
4. iOS renvoie cet ID de jeton à l'application cliente.
5. A l'aide d'un mécanisme qui lui est propre, l'application cliente transmet l'ID de jeton au fournisseur de notifications à distance, qui enregistre ce dernier pour procéder à l'envoi de notifications Push.

Procédure de notification

La procédure de notification est la suivante :

1. Le fournisseur de notifications à distance génère une notification et transmet la charge utile de la notification au service APN, accompagnée de l'ID de jeton.
2. Le service APN transfère la notification au système iOS sur le périphérique.
3. iOS envoie la charge utile de la notification à l'application.

API pour les notifications Push

AIR 3.4 intègre plusieurs API qui prennent en charge les notifications Push sur iOS. Ces API se trouvent dans le package `flash.notifications` et incluent les classes suivantes :

- `NotificationStyle` : définit les constantes pour les types de notification `ALERT`, `BADGE` et `SOUND`.C
- `RemoteNotifier` : permet de souscrire et d'annuler la souscription aux notifications Push.
- `RemoteNotifierSubscribeOptions` : permet de sélectionner les types de notifications à recevoir. Utilisez la propriété `notificationStyles` pour définir un vecteur de chaînes souscrivant à divers types de notifications.

AIR 3.4 inclut également l'événement `flash.events.RemoteNotificationEvent`, distribué par `RemoteNotifier` dans les cas suivants :

- Lorsque la souscription d'une application aboutit et après la réception d'un nouvel ID de jeton envoyé par le service APN ;
- A la réception d'une nouvelle notification à distance.

Par ailleurs, `RemoteNotifier` distribue l'événement `flash.events.StatusEvent` s'il rencontre une erreur lors du processus de souscription.

Gestion des notifications Push dans une application

Pour souscrire aux notifications Push, vous devez :

- créer un code qui souscrit aux notifications Push dans votre application ;
- activer les notifications Push dans le fichier XML de l'application ;

- créer un profil et un certificat d'attribution de privilèges d'accès pour activer les services Push sur iOS.

L'exemple de code annoté suivant souscrit aux notifications Push et gère les événements de notification 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());
                        }
                        }
                        }
```

Activation des notifications Push dans le fichier XML de l'application

Pour utiliser les notifications Push dans votre application, fournissez les informations suivantes dans la balise `Entitlements` (sous la balise `iphone`) :

```
<iphone>
                                ...
                                   <Entitlements>
                                      <![CDATA[
                                         <key>aps-environment</key>
                                         <string>development</string>
                                      ]]>
                                   </Entitlements>
                                </iphone>
```

Lorsque vous êtes prêt à soumettre l'application à l'App Store, un élément `<string>` pour le développement à la production :

```
<string>production</string>
```

Si votre application prend en charge les chaînes localisées, spécifiez les langues dans la balise `supportedLanguages`, sous la balise `intialWindow`, comme dans l'exemple suivant :

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

Création d'un profil et d'un certificat d'attribution de privilèges d'accès pour activer les services Push sur iOS

Pour permettre la communication entre le service APN et l'application, vous devez mettre en package l'application avec un profil et un certificat d'attribution de privilèges d'accès permettant d'activer les services Push sur iOS. Pour cela, procédez comme suit :

1. Connectez-vous à votre compte de développement Apple.
2. Accédez au portail de configuration.
3. Cliquez sur l'onglet App IDs.
4. Cliquez sur le bouton New App ID.
5. Entrez une description et un identificateur d'offres groupées (vous ne devez pas utiliser le signe * dans cet identificateur).
6. Cliquez sur Submit. Le portail de configuration génère votre ID d'application et vous êtes redirigé vers la page App IDs.
7. Cliquez sur Configure (en regard de votre ID d'application). La page Configure App ID s'affiche.
8. Cochez la case Enable for Apple Push Notification service. Notez qu'il existe deux types de certificats SSL Push : un pour le développement et le test, un autre pour la production.
9. Cliquez sur le bouton Configure situé en regard de Development Push SSL Certificate. La page Generate Certificate Signing Request (CSR) s'affiche.
10. Générez une demande de signature de certificat (CSR) à l'aide de l'utilitaire Keychain Access, comme indiqué sur cette page.
11. Générez le certificat SSL.
12. Téléchargez et installez le certificat SSL.
13. (Facultatif) Répétez les étapes 9 à 12 pour le certificat SSL Push de production.
14. Cliquez sur Done. La page Configure App ID s'affiche.
15. Cliquez sur Done. La page App IDs s'affiche. Notez le cercle vert en regard de la notification Push correspondant à votre ID d'application.
16. Veillez à enregistrer vos certificats SSL, car ils seront utilisés ultérieurement pour communiquer avec l'application et le fournisseur.

17 Cliquez sur l'onglet Provisioning pour afficher la page Provisioning Profiles.

18 Créez un profil d'attribution de privilèges d'accès pour votre nouvel ID d'application et téléchargez-le.

19 Cliquez sur l'onglet Certificates et téléchargez un nouveau certificat pour le nouveau profil d'attribution de privilèges d'accès.

Utilisation de sons pour les notifications Push

Pour activer les notifications sonores dans votre application, regroupez les fichiers son comme vous le feriez pour tout autre actif, mais dans le même répertoire que les fichiers SWF et app-xml. Exemple :

```
Build/adt -package -target ipa-app-store -provisioning-profile _-_.mobileprovision -storetype
pkcs12 -keystore _-_.p12 test.ipa test-app.xml test.swf sound.caf sound1.caf
```

Apple prend en charge les formats de données audio suivants (dans les fichiers aiff, wav et caf) :

- PCM linéaire
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Utilisation de notifications d'alerte localisées

Pour utiliser des notifications d'alerte localisées dans votre application, regroupez les chaînes localisées dans des dossiers lproj. Par exemple, si vous souhaitez créer des alertes en espagnol, procédez comme suit :

1 Créez un dossier es.lproj dans le projet, au même niveau que le fichier app-xml.

2 Au sein du dossier es.lproj, créez un fichier texte appelé Localizable.Strings.

3 Ouvrez le fichier Localizable.Strings dans un éditeur de texte et ajoutez les clés du message et les chaînes localisées correspondantes. Exemple :

```
"PokeMessageFormat" = "La notificación de alertas en español."
```

4 Enregistrez le fichier.

5 Lorsque l'application reçoit une notification d'alerte avec cette valeur de clé et que la langue du périphérique est l'espagnol, le texte d'alerte traduit s'affiche.

Configuration d'un fournisseur de notifications à distance

Vous devez disposer d'un fournisseur de notifications à distance pour envoyer des notifications Push à votre application. Cette application serveur agit comme un fournisseur ; elle accepte votre demande de réception de notifications Push et transmet les notifications et les données de notification au service APN qui, à son tour, envoie les notifications Push à l'application cliente.

Pour obtenir des informations détaillées sur l'envoi de notifications à partir d'un fournisseur de notifications à distance, voir l'article Provider Communication with Apple Push Notification Service dans Apple Developer Library (disponible en anglais uniquement).

Options du fournisseur de notifications à distance

Les options d'un fournisseur de notifications à distance sont les suivantes :

- Créer votre propre fournisseur en fonction du serveur Open Source APNS-php. Vous pouvez configurer un serveur PHP en suivant les instructions de la page http://code.google.com/p/apns-php/. Ce projet de code Google permet de créer une interface adaptée à vos besoins spécifiques.

- Utiliser un fournisseur de services. Par exemple, http://urbanairship.com/ offre un fournisseur de service APN prêt à l'emploi. Après avoir souscrit ce service, vous devez commencer par fournir votre jeton de périphérique à l'aide d'un code similaire au suivant :

```
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);
                                    }
```

  Vous pouvez envoyer des notifications de test à l'aide des outils Urban Airship.

Certificats pour le fournisseur de notifications à distance

Vous devez copier le certificat SSL et la clé privée (générée antérieurement) à l'emplacement approprié sur le serveur du fournisseur de notifications à distance. En règle générale, vous devez combiner ces deux fichiers dans un fichier `.pem` unique. Pour cela, procédez comme suit :

1. Ouvrez une fenêtre de terminal.
2. Créez un fichier `.pem` à partir du certificat SSL en saisissant la commande suivante :

   ```
   openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
   ```

3. Créez un fichier .pem à partir du fichier de clé privée (`.p12`) en saisissant la commande suivante :

   ```
   openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
   ```

4. Combinez ces deux fichiers .pem au sein d'un fichier unique en saisissant la commande suivante :

   ```
   cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
   ```

5. Fournissez le fichier `.pem` combiné au serveur du fournisseur lorsque vous créez votre application d'envoi côté serveur.

Pour plus d'informations, voir la rubrique Installing the SSL Certificate and Key on the Server du document Apple Local and Push Notification Programming Guide (disponible en anglais uniquement).

Gestion des notifications Push dans une application

La gestion des notifications Push dans une application implique les aspects suivants :

- Configuration et acceptation des notifications Push de la part de l'utilisateur global
- Acceptation des notifications Push individuelles de la part de l'utilisateur
- Gestion des notifications Push et des données de charge utile des notifications

Configuration et acceptation des notifications Push

La première fois qu'un utilisateur lance une application prenant en charge les notifications Push, iOS affiche la boîte de dialogue ***appname*** **Would Like to Send You Push Notifications** accompagnée des boutons Don't Allow et OK. Si l'utilisateur sélectionne OK, l'application peut recevoir tous les styles de notifications auxquels elle a souscrit. Si l'utilisateur sélectionne Don't Allow, l'application ne reçoit aucune notification.

Remarque : *les utilisateurs peuvent accéder à Réglages > Notifications pour contrôler les types de notifications spécifiques qu'ils peuvent recevoir pour chaque application prenant en charge les notifications Push.*

Selon les recommandations d'Apple, chaque fois qu'une application est activée elle doit souscrire aux notifications Push. Lorsqu'une application appelle `RemoteNotifier.subscribe()`, elle reçoit un événement `RemoteNotificationEvent` de type `token`, qui contient un ID de jeton numérique unique de 32 octets permettant d'identifier de façon unique l'application sur ce périphérique.

Lorsque le périphérique reçoit une notification Push, il affiche une fenêtre contextuelle contenant les boutons Fermer et Lancer. Si l'utilisateur appuie sur Fermer, rien ne se passe ; si l'utilisateur appuie sur Lancer, iOS invoque l'application, qui reçoit un événement `flash.events.RemoteNotificationEvent` de type `notification`, tel que décrit ci-dessous.

Gestion des notifications Push et des données de charge utile

Lorsque le fournisseur de notifications à distance envoie une notification à un périphérique (à l'aide de l'ID de jeton), votre application reçoit un événement `flash.events.RemoteNotificationEvent` de type `notification`, que l'application soit en cours d'exécution ou pas. A ce stade, votre application effectue un traitement des notifications propre à l'application. Si votre application gère les données de notification, vous pouvez y accéder via la propriété `RemoteNotificationEvent.data` au format JSON.

Chapitre 8 : Développement d'applications AIR pour périphériques TV

Fonctionnalités AIR propres aux téléviseurs

Vous pouvez créer des applications Adobe® AIR® pour périphériques TV tels que les téléviseurs, les enregistreurs vidéo numériques et les lecteurs de disque Blu-ray, sous réserve que le périphérique comporte Adobe AIR pour TV. AIR pour TV est optimisé pour les périphériques TV, notamment grâce aux accélérateurs matériels d'un périphérique, en vue d'assurer des performances vidéo et graphiques élevées.

Les applications AIR pour périphériques TV sont de type SWF plutôt que HTML. Votre application AIR pour TV peut bénéficier de l'accélération matérielle, ainsi que d'autres fonctionnalités AIR parfaitement adaptées aux salons.

Profils de périphérique

AIR fait appel à des profils pour définir un ensemble de périphériques cible dotés de fonctionnalités similaires. Utilisez les profils suivants pour les applications AIR pour TV :

- Profil `tv` : utilisez ce profil dans les applications AIR qui ciblent un périphérique TV.
- Profil `extendedTV` : Utilisez ce profil si l'application AIR pour TV fait appel à des extensions natives.

Les fonctionnalités ActionScript définies pour ces profils sont passées en revue à la section « Profils de périphérique » à la page 259. Diverses différences ActionScript liées aux applications AIR pour TV sont indiquées dans le manuel Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash.

Pour plus d'informations sur les profils AIR pour TV, voir « Profils pris en charge » à la page 149.

Accélération matérielle

Les périphériques TV intègrent des accélérateurs matériels qui augmentent considérablement les performances graphiques et vidéo dans l'application AIR. Pour exploiter ces accélérateurs matériels, voir « Considérations à prendre en compte lors de la création d'une application AIR pour TV » à la page 129.

Protection du contenu

AIR pour TV permet d'assurer au consommateur de contenu vidéo premium une expérience de qualité, qu'il s'agisse d'une superproduction hollywoodienne, d'un film d'art et d'essai ou d'un épisode de série TV. Les fournisseurs de contenu peuvent créer des applications interactives à l'aide d'outils Adobe. Ils peuvent intégrer des produits serveur Adobe à leur infrastructure de distribution de contenu ou collaborer avec l'un des membres de la communauté Adobe.

La protection du contenu constitue un élément clé de la distribution de vidéo premium. AIR pour TV prend en charge Adobe® Flash® Access™, une solution de monétisation et de protection de contenu qui satisfait aux exigences sécuritaires rigoureuses des propriétaires de contenus, y compris les principaux studios.

Flash Access prend en charge les éléments suivants :

- Téléchargement et diffusion en continu de vidéo
- Divers modèles de vente, tels que le financement par la publicité, l'abonnement, la location et la vente dématérialisée (ou EST, « Electronic Sell-Through »)

- Diverses technologies de diffusion de contenu, notamment la diffusion en continu dynamique HTTP, la diffusion en continu via RTMP (Real Time Media Protocol) par le biais de Flash® Media Server et le téléchargement progressif avec HTTP

AIR pour TV intègre également la prise en charge de RTMPE, la version chiffrée de RTMP, pour les solutions de diffusion en continu existantes aux exigences de sécurité moins rigoureuses. Les technologies de vérification RTMPE et SWF annexes sont prises en charge par Flash Media Server.

Pour plus d'informations, voir Adobe Flash Access.

Son multicanal

A partir d'AIR 3, AIR pour TV prend en charge le son multicanal pour les vidéos téléchargées progressivement à partir d'un serveur HTTP. Cette prise en charge inclut les codecs suivants :

- AC-3 (Dolby Digital)
- E-AC-3 (Enhanced Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

Remarque : *Le son multicanal des vidéos diffusées à partir d'Adobe Flash Media Server n'est pas encore pris en charge.*

Entrée de jeu

A partir d'AIR 3, AIR pour TV prend en charge les API ActionScript permettant aux applications de communiquer avec les périphériques d'entrée de jeu raccordés tels que manettes de jeu et boîtiers de commande. Bien que ces périphériques soient appelés « périphériques d'entrée de jeu », toutes les applications AIR pour TV peuvent utiliser ces périphériques, pas seulement les jeux.

Une gamme complète de périphériques d'entrée de jeu proposant diverses fonctions sont disponibles. Les périphériques sont donc généralisés dans l'API afin qu'une application puisse fonctionner correctement avec des types de périphériques d'entrée de jeu différents (et probablement inconnus).

La classe GameInput est le point d'entrée dans les API ActionScript d'entrée de jeu. Pour plus d'informations, voir GameInput.

Rendu des graphiques par accélération matérielle via l'API Stage3D

A partir d'AIR 3, AIR pour TV prend en charge le rendu des graphiques par accélération matérielle via l'API Stage3D. Les API ActionScript Stage3D sont un ensemble d'API à accélération matérielle par GPU permettant d'utiliser des fonctionnalités 2D et 3D avancées. Ces API de bas niveau permettent aux développeurs de tirer profit de l'accélération matérielle par GPU pour augmenter les performances de façon significative. Il est également possible d'utiliser des moteurs de jeu prenant en charge les API ActionScript Stage3D.

Pour plus d'informations, voir Moteurs de jeu, 3D et Stage 3D.

Extensions natives

Si l'application cible le profil `extendedTV`, elle peut faire appel à des packages ANE (AIR Native Extension).

En règle générale, les constructeurs de périphériques fournissent des packages ANE en vue d'accéder aux fonctionnalités du périphérique qui ne sont pas prises en charge par AIR. Une extension native pourrait, par exemple, permettre de changer de chaîne sur un téléviseur ou de mettre en pause la lecture sur un lecteur vidéo.

Lorsque vous mettez en package une application AIR pour TV qui fait appel aux packages ANE, vous obtenez un fichier AIRN et non un fichier AIR.

Les extensions natives pour les périphériques AIR pour TV sont toujours des extensions natives *intégrées dans le périphérique*. Cela signifie que les bibliothèques d'extensions sont installées sur le périphérique AIR pour TV. Le package ANE que vous incluez dans le package de votre application n'inclut *jamais* les bibliothèques natives de l'extension. Il contient parfois une version ActionScript uniquement de l'extension native. Cette version en ActionScript uniquement est une version temporaire ou un simulateur de l'extension. Le fabricant du périphérique installe l'extension réelle, notamment les bibliothèques natives, sur le périphérique.

Si vous développez des extensions natives, tenez compte des points suivants :

- Rapprochez-vous toujours du fabricant lorsque vous créez une extension native AIR pour TV pour ses périphériques.
- Sur certains périphériques AIR pour TV, seul le fabricant du périphérique crée des extensions natives.
- Sur tous les périphériques AIR pour TV, le fabricant du périphérique décide des extensions natives à installer.
- Les outils de développement d'extensions natives AIR pour TV varient selon le fabricant.

Pour plus d'informations sur l'utilisation d'extensions natives dans votre application AIR, voir « Utilisation d'extensions natives pour Adobe AIR » à la page 157.

Pour plus d'informations sur la création d'extensions natives, voir Développement d'extensions natives pour Adobe AIR.

Considérations à prendre en compte lors de la création d'une application AIR pour TV

Considérations liées à la vidéo

Directives d'encodage vidéo

Si vous diffusez en continu la vidéo sur un périphérique TV, Adobe recommande de respecter les directives d'encodage suivantes :

Codec vidéo :	encodage progressif, H.264 ou faisant appel au profil Principal ou Haut
Résolution :	720i, 720p, 1 080i ou 1 080p
Cadence :	24 images par seconde ou 30 images par seconde
Codec audio :	AAC-LC ou AC-3, 44,1 kHz, stéréo, ou les codecs audio multicanal suivants : E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio ou DTS-HD Master Audio

Vitesse de transmission combinée :	jusqu'à 8 Mbits/s selon la bande passante disponible
Vitesse de transmission audio :	jusqu'à 192 Kbits/s
Format des pixels :	1 × 1

Adobe recommande d'adopter le codec H.264 pour la vidéo diffusée sur un périphérique AIR pour TV.

Remarque : *AIR pour TV prend également en charge la vidéo codée par le biais des codecs Sorenson Spark ou On2 VP6. Le matériel n'assure toutefois pas le décodage ou la présentation de ces codecs. Au lieu de cela, le moteur d'exécution décode et présente ces codecs via un processus logiciel. La cadence de lecture de la vidéo est par conséquent beaucoup moins élevée. Dans la mesure du possible, il est par conséquent conseillé d'adopter le codec H.264.*

Classe StageVideo

AIR pour TV prend en charge le décodage et la présentation à accélération matérielle des vidéos codées par le biais du codec H.264. La classe StageVideo permet d'activer cette fonctionnalité.

Voir Présentation à accélération matérielle par le biais de la classe StageVideo dans le *Guide du développeur d'ActionScript 3.0* pour plus d'informations sur les points suivants :

- API de la classe StageVideo et des classes apparentées
- Restrictions d'utilisation de la classe StageVideo

Pour optimiser la prise en charge des applications AIR existantes qui adoptent l'objet Video pour les vidéos codées au format H.264, AIR pour TV utilise *en interne* un objet StageVideo. Cette caractéristique signifie que la lecture de la vidéo exploite le rendu et la présentation à accélération matérielle. L'objet Video est toutefois soumis aux mêmes restrictions qu'un objet StageVideo. Par exemple, si l'application tente de faire pivoter la vidéo, il ne se produit pas de rotation, puisque c'est le matériel, et non le moteur d'exécution, qui assure la présentation de la vidéo.

Lorsque vous programmez une nouvelle application, adoptez toutefois l'objet StageVideo pour la vidéo H.264.

Pour plus d'informations sur l'utilisation de la classe StageVideo, voir Delivering video and content for the Flash Platform on TV (disponible en anglais uniquement).

Directives de diffusion de la vidéo

Sur un périphérique AIR pour TV, la bande passante disponible du réseau risque de varier au cours de la lecture de la vidéo. Ces variations se produisent parfois lorsqu'un autre utilisateur commence à utiliser la même connexion à Internet, par exemple.

Adobe recommande par conséquent que le système de diffusion de la vidéo intègre des fonctionnalités de vitesse de transmission adaptative. Côté serveur, Flash Media Server prend par exemple en charge des fonctionnalités de vitesse de transmission adaptative. Côté client, vous disposez de la structure OSMF (Open Source Media Framework).

Les protocoles suivants permettent de diffuser un contenu vidéo via le réseau à l'intention d'une application AIR pour TV :

- Diffusion en continu dynamique HTTP et HTTPS (format F4F)
- Diffusion en continu RTMP, RTMPE, RTMFP, RTMPT et RTMPTE
- Téléchargement progressif HTTP et HTTPS

Pour plus d'informations, consulter les références suivantes :

- Guide du développeur d'Adobe Flash Media Server

- Open Source Media Framework

Considérations liées à l'audio

Le code ActionScript de lecture du son ne présente aucune différence entre les applications AIR pour TV et les autres applications AIR. Pour plus d'informations, voir Utilisation du son dans le manuel *Guide du développeur d'ActionScript 3.0*.

En ce qui concerne la prise en charge du son multicanal dans AIR pour TV, tenez compte des points suivants :

- AIR pour TV prend en charge le son multicanal des vidéos téléchargées progressivement à partir d'un serveur HTTP. Le son multicanal des vidéos diffusées à partir d'Adobe Flash Media Server n'est pas encore pris en charge.
- Bien qu'AIR pour TV prenne en charge de nombreux codecs audio, tous les *périphériques* AIR pour TV ne prennent pas en charge tous les codecs. Utilisez la méthode hasMultiChannelAudio() de `flash.system.Capabilities` pour vérifier si un périphérique AIR pour TV prend en charge un codec audio multicanaux particulier, tel que AC-3.

  Prenons par exemple le cas d'une application qui télécharge progressivement un fichier vidéo d'un serveur. Le serveur dispose de plusieurs fichiers vidéo H.264 qui prennent en charge divers codecs audio multicanaux. L'application peut utiliser la méthode `hasMultiChannelAudio()` pour déterminer le fichier vidéo à solliciter auprès du serveur. L'application peut par ailleurs envoyer au serveur la chaîne contenue dans `Capabilities.serverString`. Cette chaîne indique les codecs audio multicanaux disponibles, ce qui permet au serveur de sélectionner le fichier vidéo approprié.
- lorsque vous utilisez l'un des codecs audio DTS, il existe des cas dans lesquels `hasMultiChannelAudio()` renvoie `true`, mais où le son DTS n'est pas lu.

  Prenons par exemple un lecteur Blu-ray muni d'une sortie S/PDIF, raccordé à un ancien amplificateur. L'ancien d'amplificateur ne prend pas en charge DTS, mais S/PDIF ne dispose d'aucun protocole pour en informer le lecteur Blu-ray. Si le lecteur Blu-ray envoie le flux DTS à l'ancien amplificateur, l'utilisateur n'entend rien. Par conséquent, lors de l'utilisation de DTS, il est recommandé de fournir une interface utilisateur afin que l'utilisateur puisse indiquer si aucun son n'est en cours de lecture. Votre application peut ensuite utiliser un autre codec.

Le tableau suivant indique quand utiliser les différents codecs audio dans une application AIR pour TV : Ce tableau indique également quand les périphériques AIR pour TV font appel aux accélérateurs matériels pour décoder un codec audio. Le décodage matériel améliore les performances et décharge l'UC.

Codec audio	Disponibilité sur le périphérique AIR pour TV	Décodage matériel	Utilisation du codec audio	Informations complémentaires
AAC	Toujours	Toujours	Dans les vidéos H.264 Pour la diffusion en continu de l'audio, tel qu'un service de diffusion en continu de musique sur Internet	Lors de l'utilisation d'un flux AAC audio uniquement, encapsulez le flux dans un conteneur MP4.
MP3	Toujours	Non	Pour le son dans les fichiers SWF de l'application Dans les vidéos codées avec Sorenson Spark ou On2 VP6.	Une vidéo H.264 dont le son est au format mp3 n'est pas reproduite sur les périphériques AIR pour TV.
AC-3 (Dolby Digital) E-AC-3 (Enhanced Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Vérifier	Oui	Dans les vidéos H.264	En général, AIR pour TV transmet un flux audio multicanaux à un récepteur audio/vidéo externe qui décode et lit le son.
Speex	Toujours	Non	Lors de la réception d'un flux vocal en direct.	Une vidéo H.264 qui utilise le codec audio Speex n'est pas reproduite sur les périphériques AIR pour TV. Utilisez Speex uniquement avec les vidéos codées à l'aide de Sorenson Spark ou d'On2 VP6.
NellyMoser	Toujours	Non	Lors de la réception d'un flux vocal en direct.	Une vidéo H.264 qui utilise le codec audio NellyMoser n'est pas reproduite sur les périphériques AIR pour TV. Utilisez NellyMoser uniquement avec les vidéos codées à l'aide de Sorenson Spark ou d'On2 VP6.

Remarque : *certains fichiers vidéo contiennent deux flux audio. Par exemple, un fichier vidéo peut contenir un flux AAC et un flux AC3. AIR pour TV ne prend pas en charge ces fichiers vidéo ; l'utilisation de tels fichiers peut empêcher la diffusion du son de la vidéo.*

Accélération graphique matérielle

Utilisation de l'accélération graphique matérielle

Un périphérique AIR pour TV prend en charge l'accélération matérielle des opérations graphiques 2D. Les accélérateurs graphiques matériels du périphérique confient à l'unité centrale l'exécution des opérations suivantes :

- Rendu des bitmaps
- Mise à l'échelle des bitmaps
- Fusion des bitmaps
- Remplissage rectangulaire solide

Cette accélération graphique matérielle permet à de nombreuses opérations graphiques d'une application AIR pour TV d'être particulièrement performantes. Parmi ces opérations figurent :

- Transitions par glissement
- Transitions par mise à l'échelle
- Ouverture et fermeture en fondu
- Composition de plusieurs images avec couche alpha

Pour que ces types d'opérations exploitent les avantages de l'accélération graphique matérielle, utilisez l'une des techniques suivantes :

- Définissez la propriété `cacheAsBitmap` sur `true` pour les objets MovieClip et autres objets d'affichage dont le contenu ne change généralement pas. Exécutez ensuite des transitions par glissement, des transitions en fondu et la fusion alpha sur ces objets.
- Utilisez la propriété `cacheAsBitmapMatrix` sur les objets d'affichage qui requièrent une mise à l'échelle ou une translation (appliquez le repositionnement x et y).

  Les accélérateurs matériels du périphérique exécutent les tâches requises en recourant aux opérations de la classe Matrix pour la mise à l'échelle et la translation. Vous pouvez également envisager un scénario dans lequel vous modifiez les dimensions d'un objet d'affichage dont la propriété `cacheAsBitmap` est définie sur `true`. En cas de modification des dimensions, le logiciel du moteur d'exécution actualise le bitmap. Les performances d'une actualisation effectuée par le biais du logiciel sont inférieures à une mise à l'échelle réalisée via une accélération matérielle par le biais d'une opération Matrix.

  Considérez par exemple une application qui affiche une image dont la taille augmente lorsqu'un utilisateur la sélectionne. Utilisez plusieurs fois l'opération de mise à l'échelle de la classe Matrix pour donner une impression d'agrandissement de l'image. Toutefois, selon la taille de l'image d'origine et celle de l'image finale, la qualité de cette dernière risque d'être inacceptable. Réinitialisez par conséquent les dimensions de l'objet d'affichage une fois les opérations d'agrandissement terminées. Etant donné que la propriété `cacheAsBitmap` est définie sur `true`, le logiciel du moteur d'exécution actualise une fois seulement l'objet d'affichage et produit une image de qualité élevée.

  Remarque : *en règle générale, les périphériques AIR pour TV ne prennent pas en charge la rotation et l'inclinaison à accélération matérielle. Par conséquent, si vous spécifiez une opération de rotation et d'inclinaison dans la classe Matrix, AIR pour TV exécute toutes les opérations Matrix par le biais du logiciel. Ces opérations logicielles ont parfois des répercussions négatives sur les performances.*
- Utilisez la classe BitmapData pour créer un comportement personnalisé de mise en cache des bitmaps.

Pour plus d'informations sur la mise en cache sous forme de bitmap, consulter les références suivantes :

- Mise en cache des objets d'affichage
- Mise en cache sous forme de bitmap
- Mise en cache manuelle sous forme de bitmap

Gestion de la mémoire graphique

Pour effectuer des opérations graphiques accélérées, les accélérateurs graphiques ont recours à une mémoire graphique spéciale. Si l'application utilise la totalité de la mémoire graphique, elle s'exécute plus lentement, car AIR pour TV confie à nouveau le traitement des opérations graphiques au logiciel.

Pour gérer l'utilisation de la mémoire graphique par l'application :

- Une fois l'utilisation d'une image ou autres données de bitmap terminée, libérez la mémoire graphique correspondante. Pour ce faire, appelez la méthode `dispose()` de la propriété `bitmapData` de l'objet Bitmap. Exemple :

  ```
  myBitmap.bitmapData.dispose();
  ```

  Remarque : *libérer la référence à l'objet BitmapData ne libère pas immédiatement la mémoire graphique. L'utilitaire de récupération de place du moteur d'exécution libère la mémoire graphique, mais appeler la méthode* `dispose()` *permet à l'application de mieux contrôler le processus.*

- Faites appel à PerfMaster Deluxe, une application AIR fournie par Adobe, pour mieux comprendre l'accélération graphique matérielle sur le périphérique cible. Cette application illustre le nombre d'images par seconde requis pour exécuter diverses opérations. PerfMaster Deluxe permet de comparer diverses implémentations d'une même opération. Vous pouvez, par exemple, comparer le déplacement d'une image bitmap au déplacement d'une image vectorielle. PerfMaster Deluxe est disponible à l'adresse suivante : Flash Platform for TV.

Gestion de la liste d'affichage

Pour rendre un objet d'affichage invisible, définissez sa propriété `visible` sur `false`. L'objet se trouve toujours dans la liste d'affichage, mais AIR pour TV n'effectue pas son rendu et ne l'affiche pas. Cette technique est utile pour les objets qui apparaissent et disparaissent fréquemment, car elle n'implique qu'une faible charge de traitement. Définir la propriété `visible` sur `false` ne libère toutefois aucune des ressources de l'objet. Par conséquent, lorsque l'affichage d'un objet est terminé ou si un objet ne va plus s'afficher pendant un certain temps, supprimez l'objet de la liste d'affichage. Définissez également toutes les références à l'objet sur `null`. Ces actions permettent au nettoyeur de mémoire de libérer les ressources de l'objet.

Utilisation des images PNG et JPEG

PNG et JPEG sont des formats d'image standard dans les applications. En ce qui concerne ces formats d'image dans les applications AIR pour TV, tenez compte des considérations suivantes :

- AIR pour TV a généralement recours à l'accélération matérielle pour décoder les fichiers JPEG.
- AIR pour TV décode généralement les fichiers PNG par le biais du logiciel. Le décodage logiciel des fichiers PNG est rapide.
- Le format PNG est le seul format bitmap multiplate-forme qui prend en charge la transparence (un canal alpha).

Par conséquent, utilisez comme suit ces formats d'image dans une application :

- Adoptez le format JPEG pour les photographies en vue d'exploiter le décodage à accélération matérielle.
- Adoptez le format d'image PNG pour les éléments de l'interface utilisateur. Les performances du décodage logiciel sont suffisamment élevées pour ces éléments, qui gèrent un paramètre alpha.

Scène dans les applications AIR pour TV

Si vous créez une application AIR pour TV, tenez compte des considérations suivantes lors de l'utilisation de la classe Stage :

- Résolution d'écran
- Zone de visualisation efficace
- Mode de mise à l'échelle de la scène
- Alignement de la scène

- État d'affichage de la scène
- Prise en compte de tailles d'écran diverses
- Paramètre de qualité de la scène

Résolution d'écran

A l'heure actuelle, la résolution des écrans de périphériques TV correspond à 540p, 720p et 1 080p. Ces résolutions d'écran donnent lieu aux valeurs suivantes dans la classe Capabilities d'ActionScript :

Résolution d'écran	`Capabilities.screenResolutionX`	`Capabilities.screenResolutionY`
540p	960	540
720p	1 280	720
1 080p	1 920	1 080

Pour créer une application AIR pour TV en plein écran destinée à un périphérique donné, codez en dur les propriétés `Stage.stageWidth` et `Stage.stageHeight` sur la résolution de l'écran du périphérique. Toutefois, pour créer une application en plein écran qui s'exécute sur plusieurs périphériques, définissez les dimensions de l'objet Stage à l'aide des propriétés `Capabilities.screenResolutionX` et `Capabilities.screenResolutionY`.

Exemple :

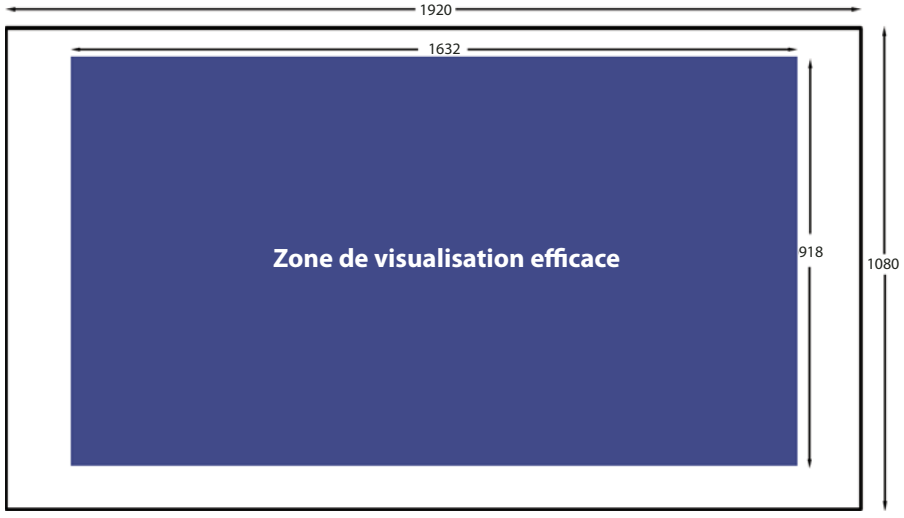
```
stage.stageWidth = Capabilities.screenResolutionX;
stage.stageHeight = Capabilities.screenResolutionY;
```

Zone de visualisation efficace

La *zone de visualisation efficace* d'un téléviseur correspond à la partie de l'écran qui est encastrée par rapport aux bords de l'écran. L'écart est suffisamment important pour que l'utilisateur final puisse visualiser la zone dans son intégralité, sans que la bordure du téléviseur ne la masque partiellement. Etant donné que les dimensions de la bordure physique qui entoure l'écran varient selon le constructeur, l'écart requis est défini en conséquence. La zone de visualisation efficace tente de garantir la visibilité d'une partie de l'écran. Elle porte également le nom de *zone de titre sécurisée*.

Le *surbalayage* identifie la partie de l'écran qui n'est pas visible, car elle est placée derrière la bordure.

Adobe recommande un écart de 7,5 % sur chaque bord de l'écran. Exemple :



Zone de visualisation efficace pour une résolution d'écran de 1 920 x 1 080

Veillez à tenir compte de la zone de visualisation efficace lorsque vous créez une application AIR pour TV en plein écran :

- Utilisez l'écran entier pour les arrière-plans (images ou couleurs d'arrière-plan).
- Limitez-vous à la zone de visualisation efficace pour les éléments déterminants de l'application tels que le texte, les graphiques, la vidéo et les éléments de l'interface utilisateur (les boutons, par exemple).

Le tableau ci-dessous indique les dimensions de la zone de visualisation efficace pour chaque résolution d'écran standard avec un écart de 7,5 %.

Résolution d'écran	Largeur et hauteur de la zone de visualisation efficace	Largeur de l'écart gauche et droit	Hauteur de l'écart supérieur et inférieur
960 x 540	816 x 460	72	40
1 280 x 720	1 088x 612	96	54
1 920 x 1 080	1 632 x 918	144	81

Il est toutefois recommandé de calculer la zone de visualisation efficace de façon dynamique. Exemple :

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

Mode de mise à l'échelle de la scène

Définissez la propriété `Stage.scaleMode` sur `StageScaleMode.NO_SCALE` et écoutez les événements de redimensionnement de la scène.

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

Ce paramètre définit les coordonnées de la scène sur les coordonnées de pixel. A l'instar de l'état d'affichage `FULL_SCREEN_INTERACTIVE` et de l'alignement de la scène `TOP_LEFT`, ce paramètre permet d'exploiter pleinement la zone de visualisation efficace.

Dans les applications en plein écran, ce mode de mise à l'échelle signifie que les propriétés `stageWidth` et `stageHeight` de la classe Stage correspondent aux propriétés `screenResolutionX` et `screenResolutionY` de la classe Capabilities.

Par ailleurs, lorsque la fenêtre de l'application change de taille, le contenu de la scène conserve sa taille définie. Le moteur d'exécution n'exécute ni mise en forme, ni mise à l'échelle automatique. Il distribue également l'événement `resize` de la classe Stage en cas de changement de taille de la fenêtre. Vous maîtrisez par conséquent pleinement le mode de réglage du contenu de l'application au démarrage de cette dernière et lors du redimensionnement de sa fenêtre.

Remarque : *le comportement `NO_SCALE` est identique à celui de toute application AIR. Dans une application AIR pour TV, l'utilisation de ce paramètre joue toutefois un rôle déterminant dans l'utilisation de la zone de visualisation efficace.*

Alignement de la scène

Définissez la propriété `Stage.align` sur `StageAlign.TOP_LEFT` :

```
stage.align = StageAlign.TOP_LEFT;
```

Cet alignement place la coordonnée `0,0` dans l'angle supérieur gauche de l'écran, ce qui est adapté au positionnement du contenu à l'aide d'ActionScript.

A l'instar du mode de mise à l'échelle `NO_SCALE` et de l'état d'affichage `FULL_SCREEN_INTERACTIVE`, ce paramètre permet d'exploiter pleinement la zone de visualisation efficace.

État d'affichage de la scène

Dans une application AIR pour TV en plein écran, définissez la propriété `Stage.displayState` sur `StageDisplayState.FULL_SCREEN_INTERACTIVE` :

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Cette valeur définit l'application AIR de sorte que la scène occupe la totalité de l'écran et autorise la saisie utilisateur.

Adobe recommande l'utilisation du paramètre `FULL_SCREEN_INTERACTIVE`. A l'instar du mode de mise à l'échelle `NO_SCALE` et de l'alignement de la scène `TOP_LEFT`, ce paramètre permet d'exploiter pleinement la zone de visualisation efficace.

Par conséquent, pour une application en plein écran, procédez comme suit dans un gestionnaire associé à l'événement `ADDED_TO_STAGE` dans la classe principale du document :

```
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;
}
```

Puis, dans le gestionnaire de l'événement `RESIZE` :

- Comparez les tailles de résolution d'écran à la largeur et la hauteur de la scène. Si elles sont identiques, l'événement `RESIZE` s'est produit, parce que l'état d'affichage de la scène est passé à `FULL_SCREEN_INTERACTIVE`.
- Calculez et enregistrez les dimensions de la zone de visualisation efficace et les écarts correspondants.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {

        // Calculate and save safe viewing area dimensions.
    }
}
```

Lorsque les dimensions de la scène équivalent à `Capabilities.screenResolutionX` et `screenResolutionY`, AIR pour TV oblige le périphérique à fournir la meilleur fidélité possible pour vos vidéos et images.

Remarque : *la fidélité à laquelle les images et les vidéos s'affichent sur l'écran d'un téléviseur peut différer des valeurs de `Capabilities.screenResolutionX` et de `screenResolutionY`, qui dépendent du périphérique exécutant AIR pour TV. Par exemple, la résolution d'écran du décodeur exécutant AIR pour TV peut être de 1280 x 720 et celle du téléviseur raccordé de 1920 x 1080. AIR pour TV oblige toutefois le périphérique à fournir la meilleure fidélité possible. Par conséquent, dans cet exemple, le périphérique affiche une vidéo de 1080p avec une résolution d'écran de 1920 x 1080.*

Prise en compte de tailles d'écran diverses

Vous pouvez développer la même application AIR pour TV en plein écran de sorte à assurer son exécution sur divers périphériques AIR pour TV sans problème d'affichage. Procédez comme suit :

1. Définissez les propriétés de la scène `scaleMode`, `align` et `displayState` sur les valeurs recommandées : `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` et `StageDisplayState.FULL_SCREEN_INTERACTIVE` (respectivement).
2. Configurez la zone de visualisation efficace en fonction des paramètres `Capabilities.screenResolutionX` et `Capabilities.screenResolutionY`.
3. Réglez la taille et la mise en forme du contenu en fonction de la largeur et de la hauteur de la zone de visualisation efficace.

   En dépit de la taille élevée des objets du contenu, en particulier si vous les comparez aux applications pour périphériques mobiles, les concepts tels que la mise en forme dynamique, le positionnement relatif et le contenu adaptatif demeurent inchangés. Pour plus d'informations sur la prise en charge de ces concepts par le biais d'ActionScript, voir Authoring mobile Flash content for multiple screen sizes (disponible en anglais uniquement).

Qualité de la scène

La propriété `Stage.quality` d'une application AIR pour TV est toujours `StageQuality.High`. Il est impossible de la modifier.

Cette propriété spécifie la qualité de rendu de tous les objets Stage.

Gestion de la saisie par télécommande

Les utilisateurs communiquent généralement avec l'application AIR pour TV à l'aide d'une télécommande. Gérez toutefois la saisie par touche de télécommande comme la saisie par touche de clavier dans une application de bureau. Gérez en particulier l'événement `KeyboardEvent.KEY_DOWN`. Pour plus d'informations, voir Capture de la saisie au clavier dans le manuel *Guide du développeur d'ActionScript 3.0*.

Les touches de la télécommande sont mappées sur des constantes ActionScript. Par exemple, le mappage des touches du pavé directionnel d'une télécommande est le suivant :

Touche du pavé directionnel d'une télécommande	Constante ActionScript 3.0
Haut	`Keyboard.UP`
Bas	`Keyboard.DOWN`
Gauche	`Keyboard.LEFT`
Droite	`Keyboard.RIGHT`
OK ou Sélectionner	`Keyboard.ENTER`

AIR 2.5 intègre de nombreuses autres constantes Keyboard pour prendre en charge la saisie par télécommande. Pour en consulter la liste complète, voir Classe Keyboard dans le manuel *Guide de référence ActionScript 3.0 pour la plate-forme Adobe Flash*.

Pour assurer le fonctionnement de l'application sur un nombre aussi élevé que possible de périphériques, Adobe recommande de respecter les directives suivantes :

- Dans la mesure du possible, utilisez uniquement les touches du pavé directionnel.

  Les ensembles de touches varient selon les télécommandes. La grande majorité des télécommandes intègrent toutefois des touches de pavé directionnel.

  Ainsi, une télécommande de périphérique Blu-ray ne comporte généralement pas de touche permettant de passer à la chaîne suivante ou à la chaîne précédente. Même les touches de lecture, pause et arrêt ne figurent pas sur toutes les télécommandes.

- Utilisez les touches Menu et Informations si les touches du pavé directionnel ne suffisent pas à l'application.

  Après les touches du pavé directionnel, les touches Menu et Informations sont les plus courantes sur les télécommandes.

- Tenez compte de l'utilisation fréquente des télécommandes universelles.

  Même si vous créez une application destinée à un périphérique déterminé, gardez à l'esprit le fait que de nombreux utilisateurs préfèrent éviter la télécommande fournie avec le périphérique. Ils ont au contraire recours à une télécommande universelle. Par ailleurs, les utilisateurs ne programment pas toujours la télécommande universelle de sorte à mapper toutes les touches de la télécommande du périphérique. Il est par conséquent recommandé de n'utiliser que les touches les plus courantes.

- Veillez à ce que les utilisateurs puissent désactiver toute opération par le biais de l'une des touches du pavé directionnel.

  L'application a parfois recours à bon escient à une touche qui n'est pas fréquemment utilisée sur une télécommande. Fournir une voie de sortie à l'aide de l'une des touches du pavé directionnel assure le comportement approprié de l'application sur tous les périphériques.

- Ne sollicitez pas d'entrée de pointeur à moins que vous ne sachiez que le périphérique AIR pour TV cible dispose d'une fonctionnalité d'entrée de pointeur.

  Bien que de nombreuses applications de bureau attendent une entrée de souris, la plupart des téléviseurs ne prennent pas en charge l'entrée de pointeur. Par conséquent, si vous convertissez une application de bureau en application pour téléviseur, veillez à la modifier de façon à ce qu'elle n'attende pas d'entrée de souris. Ces modifications affectent la gestion des événements et les instructions destinées à l'utilisateur. Lorsque l'écran de démarrage de l'application apparaît, n'affichez pas le texte « Cliquez pour démarrer », par exemple.

Gestion de la cible d'action

Lorsqu'un élément de l'interface utilisateur correspond à la cible d'action dans une application de bureau, il est la cible d'événements de saisie utilisateur tels que les événements de clavier et de souris. Par ailleurs, une application affiche en surbrillance l'élément de l'interface utilisateur qui est la cible d'action. Une application AIR pour TV et une application de bureau gèrent différemment la cible d'action, car :

- Les applications de bureau ont souvent recours à la touche de tabulation pour faire de l'élément suivant de l'interface utilisateur la cible d'action. L'utilisation de la touche de tabulation ne s'applique pas aux applications AIR pour TV. Les télécommandes ne comportent généralement pas de touche de tabulation. Par conséquent, à l'encontre du bureau, la gestion de la cible d'action à l'aide de la propriété `tabEnabled` d'un objet DisplayObject n'est pas prise en charge.
- Les applications de bureau attendent souvent de l'utilisateur qu'il sélectionne un élément de l'interface utilisateur à l'aide de la souris.

Procédez de ce fait comme suit dans l'application :

- Ajoutez un écouteur d'événements à l'objet Stage qui écoute des événements Keyboard tels que `KeyboardEvent.KEY_DOWN`.
- Définissez la logique de l'application de sorte à déterminer l'élément de l'interface à mettre en surbrillance à l'intention de l'utilisateur. Veillez à mettre en surbrillance un élément de l'interface utilisateur au démarrage de l'application.
- En vous basant sur la logique de l'application, distribuez l'événement Keyboard reçu par l'objet Stage à l'objet approprié de l'élément de l'interface utilisateur.

  Vous disposez également de la propriété `Stage.focus` ou `Stage.assignFocus()` pour faire d'un élément de l'interface utilisateur la cible d'action. Vous pouvez ensuite ajouter un écouteur d'événements à cet objet DisplayObject pour qu'il reçoive les événements de clavier.

Création d'une interface utilisateur

Pour assurer le bon fonctionnement de l'interface utilisateur d'une application AIR pour TV sur un téléviseur, tenez compte des recommandations relatives :

- au temps de réaction de l'application ;
- à la convivialité de l'application ;
- à la personnalité et aux attentes de l'utilisateur.

Temps de réaction

Tenez compte des conseils suivants pour réduire autant que possible le temps de réaction d'une application AIR pour TV.

- Réduisez autant que possible la taille du fichier SWF initial de l'application.

  Dans le fichier SWF initial, chargez uniquement les ressources requises pour démarrer l'application. Limitez-vous par exemple à l'image de l'écran de démarrage de l'application.

  Bien que cette recommandation s'applique aux applications AIR de bureau, elle est d'une importance particulière pour les périphériques AIR pour TV. Les périphériques AIR pour TV ne disposent par exemple pas de la puissance de traitement d'un ordinateur de bureau. Ils stockent par ailleurs l'application en mémoire flash, dont l'accès n'est pas aussi rapide qu'un disque dur sur un ordinateur de bureau.
- Veillez à ce que l'application s'exécute à une cadence de 20 images/seconde au moins.

Créez les graphiques de sorte à atteindre cet objectif. La complexité des opérations graphiques risque d'affecter la cadence. Pour plus d'informations sur l'amélioration des performances de rendu, voir Optimisation des performances pour la plate-forme Adobe Flash.

Remarque : *le matériel graphique d'un périphérique AIR pour TV applique généralement un taux de rafraîchissement de l'écran de 60 ou 120 Hz (en d'autres termes, l'écran est mis à jour 60 ou 120 fois par seconde). Le matériel analyse la scène pour identifier les mises à jour à une cadence de 30 ou 60 images par seconde, par exemple, sur un écran de 60 ou 120 Hz. La complexité des opérations graphiques de l'application détermine toutefois si ces cadences élevées sont appliquées ou non.*

- Mettez à jour l'écran dans les 100 - 200 millisecondes qui suivent une saisie utilisateur.

  Si le délai de mise à jour est plus long, l'utilisateur s'impatiente et appuie souvent plusieurs fois sur les touches.

Convivialité

Les utilisateurs d'une application AIR pour TV résident dans un environnement de type « salon ». Ils sont assis à 3 mètres au moins du téléviseur. La pièce est souvent sombre. Ils ont généralement recours à une télécommande à des fins de saisie. Il est possible que plusieurs personnes utilisent l'application, parfois conjointement, parfois en série.

Tenez par conséquent compte des considérations suivantes pour assurer la convivialité de l'interface utilisateur sur un téléviseur :

- Assignez une taille élevée aux éléments de l'interface utilisateur.

  Lors de la conception du texte, des boutons ou de tout autre élément de l'interface utilisateur, tenez compte du fait que l'utilisateur est assis à l'autre bout de la pièce. Assurez-vous que tout élément est clairement visible et lisible à une distance de 3 mètres, par exemple. Evitez d'encombrer l'écran pour l'unique raison que l'écran est de taille élevée.

- Définissez un contraste approprié afin que l'utilisateur puisse visualiser et lire aisément le contenu de l'autre bout de la pièce.
- Identifiez clairement la cible d'action en assurant la luminosité de l'élément de l'interface utilisateur concerné.
- N'utilisez le mouvement qu'en cas de besoin. Glisser d'un écran à l'autre à des fins de continuité, par exemple, peut donner des résultats satisfaisants. Toutefois, le mouvement empêche parfois l'utilisateur de se concentrer s'il ne l'aide pas à naviguer ou s'il n'est pas intrinsèque à l'application.
- Proposez toujours à l'utilisateur un moyen évident de retourner à son point de départ par le biais de l'interface utilisateur.

Pour plus d'informations sur l'utilisation de la télécommande, voir « Gestion de la saisie par télécommande » à la page 138.

Personnalité et attentes de l'utilisateur

Gardez à l'esprit le fait que les utilisateurs d'applications AIR pour TV recherchent généralement une expérience télévisuelle divertissante de qualité, dans un environnement dénué de stress. Ils ne maîtrisent pas nécessairement l'informatique.

Créez de ce fait des applications AIR pour TV aux caractéristiques suivantes :

- Evitez le jargon technique.
- Evitez les boîtes de dialogue modales.
- Utilisez des instructions conviviales et informelles adaptées à un environnement de type salon, plutôt qu'à la sphère professionnelle ou technique.
- Intégrez les graphiques de qualité élevée auxquels s'attendent les téléspectateurs.

- Créez une interface utilisateur qui fonctionne de façon transparente avec une télécommande. N'utilisez pas une interface utilisateur ou des éléments de conception adaptés à une application de bureau ou à une application mobile. Par exemple, dans les interfaces utilisateur des périphériques de bureau et des périphériques mobiles, il est souvent nécessaire de pointer, cliquer ou appuyer sur des boutons avec une souris ou un doigt.

Polices et texte

Dans l'application AIR pour TV, utilisez les polices de périphérique ou les polices incorporées.

Les polices de périphérique sont installées sur le périphérique. Tous les périphériques AIR pour TV intègrent les polices suivantes :

Nom de la police	Description
`_sans`	La police de périphérique `_sans` contient des caractères sans serif. La police de périphérique `_sans` installée sur tous les périphériques AIR pour TV est Myriad Pro. En règle générale, une police sans-sérif est mieux reproduite sur un téléviseur que les polices sérif en raison de la distance de visionnage.
`_serif`	La police de périphérique `_serif` contient des caractères serif. La police de périphérique `_serif` installée sur tous les périphériques AIR pour TV est Minion Pro.
`_typewriter`	La police de périphérique `_typewriter` est une police à espacement fixe. La police de périphérique `_typewriter` installée sur tous les périphériques AIR pour TV est Courier Std.

Tous les périphériques AIR pour TV intègrent également les polices de périphérique asiatiques suivantes :

Nom de la police	Langue	Catégorie de caractère	Code des paramètres régionaux
RyoGothicPlusN-Regular	japonais	sans	ja
RyoTextPlusN-Regular	japonais	serif	ja
AdobeGothicStd-Light	coréen	sans	ko
AdobeHeitiStd-Regular	chinois simplifié	sans	zh_CN
AdobeSongStd-Light	chinois simplifié	serif	zh_CN
AdobeMingStd-Light	chinois traditionnel	serif	zh_TW et zh_HK

Ces polices de périphérique AIR pour TV :

- sont extraites de la typothèque d'Adobe® ;
- donnent un résultat satisfaisant sur un téléviseur ;
- sont adaptées au titrage vidéo ;
- sont des polices vectorielles, plutôt que les polices bitmap.

Remarque : *les constructeurs de périphériques intègrent souvent d'autres polices à ces derniers. Ces polices sont installées parallèlement aux polices de périphérique AIR pour TV.*

Adobe fournit l'application FontMaster Deluxe, qui affiche toutes les polices du périphérique. Vous trouverez cette application à l'adresse Flash Platform for TV.

Vous pouvez également incorporer des polices à l'application AIR pour TV. Pour plus d'informations sur les polices incorporées, voir Fonctions avancées d'affichage de texte dans le manuel *Guide du développeur d'ActionScript 3.0*.

Adobe recommande d'appliquer les directives d'utilisation des champs de texte TLF suivantes :

- Faites appel aux champs de texte TLF pour le texte en langue asiatique afin d'exploiter les paramètres régionaux en vigueur dans l'application. Définissez la propriété `locale` de l'objet TextLayoutFormat associé à l'objet TLFTextField. Pour déterminer les paramètres régionaux actifs, voir Choix d'un jeu de paramètres régionaux dans le manuel *Guide du développeur d'ActionScript 3.0*.
- Spécifiez le nom de la police dans la propriété `fontFamily` de l'objet TextLayoutFormat si la police ne fait pas partie des polices de périphérique AIR pour TV. AIR pour TV utilise la police si elle est disponible sur le périphérique. Si la police requise ne figure pas sur le périphérique, AIR pour TV lui substitue la police de périphérique AIR pour TV appropriée en se référant au paramètre `locale`.
- Spécifiez le paramètre `_sans`, `_serif` ou `_typewriter` pour la propriété `fontFamily` et définissez la propriété `locale` pour qu'AIR pour TV sélectionne la police de périphérique AIR pour TV appropriée. Selon les paramètres régionaux en vigueur, AIR pour TV effectue une sélection dans le jeu de polices de périphérique asiatiques ou non asiatiques. Ces paramètres permettent d'utiliser automatiquement la police appropriée pour les quatre principaux jeux de paramètres régionaux et l'anglais.

Remarque : *si vous utilisez des champs de texte classiques pour le texte en langue asiatique, spécifiez le nom d'une police de périphérique AIR pour TV pour assurer un rendu approprié. Si vous savez qu'une autre police est installée sur le périphérique cible, vous pouvez également l'indiquer.*

En ce qui concerne les performances de l'application, tenez compte des considérations suivantes :

- Les champs de texte classiques assurent de meilleures performances que les champs de texte TLF.
- Un champ de texte classique qui a recours à des polices bitmap assure des performances optimales.

  A l'encontre des polices vectorielles, qui ne proposent que des données vectorielles associées à chaque caractère, les polices bitmap proposent un bitmap par caractère. Les polices de périphérique comme les polices incorporées peuvent correspondre à des polices bitmap.
- Si vous spécifiez une police de périphérique, veillez à l'installer sur le périphérique cible. Si tel n'est pas le cas, AIR pour TV recherche et utilise une autre police installée sur le périphérique. Ce comportement ralentit toutefois les performances de l'application.
- A l'instar de tout objet d'affichage, si un objet TextField n'est que rarement modifié, définissez la propriété `cacheAsBitmap` correspondante sur `true`. Ce paramètre améliore les performances des transitions de type fondu, glissement et fusion alpha. Utilisez la propriété `cacheAsBitmapMatrix` pour la mise à l'échelle et la translation. Pour plus d'informations, voir « Accélération graphique matérielle » à la page 132.

Sécurité du système de fichiers

Etant donné que les applications AIR pour TV sont des applications AIR, elles peuvent accéder au système de fichiers du périphérique. Toutefois, si l'application est installée sur un « périphérique de salon », il est déterminant qu'elle ne puisse pas accéder au système de fichiers de ce dernier ou aux fichiers d'autres applications. Les utilisateurs de téléviseurs ou périphériques associés ne s'attendent pas à des pannes et ne les tolèrent pas. Après tout, ils regardent la télévision.

Une application AIR pour TV dispose par conséquent d'une vue limitée du système de fichiers du périphérique. Par le biais d'ActionScript 3.0, l'application peut accéder à des répertoires (et sous-répertoires associés) déterminés uniquement. Par ailleurs, les noms de répertoires en vigueur dans ActionScript ne correspondent pas aux noms de répertoires réels sur le périphérique. Cette couche supplémentaire permet de protéger les applications AIR pour TV contre tout accès malveillant ou accidentel à des fichiers locaux qui ne leur appartiennent pas.

Pour plus d'informations, voir Vue arborescente associée aux applications AIR pour TV.

Sandbox de l'application AIR

Les applications AIR pour TV s'exécutent dans le sandbox d'application AIR, décrit dans Sandbox de l'application AIR.

L'unique différence pour les applications AIR pour TV consiste en un accès limité au système de fichiers, comme indiqué à la section « Sécurité du système de fichiers » à la page 143.

Cycle de vie d'une application

A l'encontre d'un environnement de bureau, l'utilisateur final ne peut pas fermer la fenêtre d'exécution de l'application AIR pour TV. Intégrez de ce fait à l'interface utilisateur un mécanisme de sortie de l'application.

En règle générale, un périphérique autorise l'utilisateur final à quitter sans condition une application à l'aide de la touche de sortie de la télécommande. AIR pour TV ne distribue toutefois pas d'événement `flash.events.Event.EXITING` à l'application. Enregistrez de ce fait régulièrement l'état de l'application, afin qu'elle puisse restaurer un état raisonnable lors du démarrage suivant.

Cookies HTTP

AIR pour TV prend en charge les cookies persistants HTTP et les cookies de session. AIR pour TV enregistre les cookies de chaque application AIR dans un répertoire propre à l'application :

`/app-storage/<app id>/Local Store`

Le fichier de cookies est appelé `cookies`.

Remarque : *sur les autres périphériques, notamment les périphériques de bureau, AIR n'enregistre pas les cookies séparément pour chaque application. L'enregistrement des cookies propres à une application prend en charge le modèle de sécurité de l'application et du système d'AIR pour TV.*

Faites appel à la propriété ActionScript `URLRequest.manageCookies` comme suit :

- Définissez `manageCookies` sur `true`. Cette valeur est le paramètre par défaut. Elle signifie qu'AIR pour TV ajoute automatiquement des cookies aux requêtes HTTP et les mémorise dans la réponse HTTP.

  Remarque : *même si la propriété `manageCookies` est définie sur `true`, l'application peut ajouter manuellement un cookie à une requête HTTP à l'aide d'`URLRequest.requestHeaders`. Si ce cookie porte le même nom qu'un cookie géré par AIR pour TV, la requête contient deux cookies au nom identique. La valeur des deux cookies peut être différente.*

- Définissez `manageCookies` sur `false`. Cette valeur signifie qu'il incombe à l'application d'envoyer des cookies dans les requêtes HTTP et de les mémoriser dans la réponse HTTP.

Pour plus d'informations, voir URLRequest.

Flux de travail de développement d'une application AIR pour TV

Vous pouvez développer une application AIR pour TV à l'aide des outils de développement suivants de la plate-forme Adobe Flash :

- Adobe Flash Professional

  Adobe Flash Professional CS5.5 prend en charge AIR 2.5 pour TV, la première version d'AIR compatible avec les applications AIR pour TV.

- Adobe Flash® Builder®

  Flash Builder 4.5 prend en charge AIR 2.5 pour TV.

- Kit de développement d'AIR

  A partir d'AIR 2.5, vous pouvez développer des applications à l'aide des outils de ligne de commande fournis avec le kit SDK d'AIR. Pour télécharger le kit SDK d'AIR, voir http://www.adobe.com/fr/products/air/sdk/.

Utilisation de Flash Professional

Développer, tester et publier des applications AIR pour TV à l'aide de Flash Professional s'apparente aux procédures mises en œuvre pour les applications AIR de bureau.

Toutefois, pour programmer le code ActionScript 3.0, utilisez uniquement les classes et méthodes prises en charge par les profils AIR `tv` et `extendedTV`. Pour plus d'informations, voir « Profils de périphérique » à la page 259.

Paramètres d'un projet

Procédez comme suit pour configurer le projet associé à une application AIR pour TV :

- Dans l'onglet Flash de la boîte de dialogue Paramètres de publication, définissez la valeur du lecteur sur au moins AIR 2.5.
- Dans l'onglet Général de la boîte de dialogue Paramètres d'Adobe AIR (Paramètres de l'application Adobe AIR et du programme d'installation), définissez le profil sur `TV` ou TV `étendu`.

Débogage

Vous pouvez exécuter l'application à l'aide de l'application de débogage du lanceur AIR (ADL) intégrée à Flash Professional. Procédez comme suit :

- Pour exécuter l'application en mode de débogage, sélectionnez :

  Déboguer > Déboguer l'animation > dans l'application de débogage du lanceur AIR (bureau)

  Il suffit ensuite de sélectionner les options suivantes lors des exécutions de débogage ultérieures :

  Déboguer > Déboguer l'animation > Déboguer

- Pour exécuter l'application sans fonctionnalité de débogage, sélectionnez :

  Contrôle > Tester l'animation > dans l'application de débogage du lanceur AIR (bureau)

  Il suffit ensuite de sélectionner les options Contrôle > Tester l'animation > Tester lors des exécutions ultérieures.

Etant donné que vous avez défini le profil AIR sur TV ou TV étendu, l'application de débogage du lanceur AIR (ADL) affiche le menu Touches de commande à distance. Ce menu permet de simuler la sélection de touches sur une télécommande.

Pour plus d'informations, voir « Débogage à distance avec Flash Professional » à la page 155.

Utilisation des extensions natives

Si votre application fait appel à une extension native, suivez les instructions de la section « Liste de tâches pour l'utilisation d'une extension native » à la page 159.

Toutefois, si une application utilise des extensions natives :

- Il est impossible de publier l'application à l'aide de Flash Professional. Pour publier l'application, faites appel à l'outil AIR Developer (ADT). Voir « Mise en package avec l'outil ADT » à la page 151.
- Il est impossible d'exécuter ou de déboguer l'application avec Flash Professional. Pour déboguer l'application sur la machine de développement, faites appel à l'application ADL. Voir « Simulation de périphérique à l'aide de l'application ADL » à la page 152

Utilisation de Flash Builder

A partir de la version 4.5, Flash Builder prend en charge le développement d'applications AIR pour TV. Développer, tester et publier des applications AIR pour TV à l'aide de Flash Builder s'apparente aux procédures mises en œuvre pour les applications AIR de bureau.

Configuration de l'application

Assurez-vous que l'application :

- Utilise l'élément `Application` en tant que classe de conteneur dans le fichier MXML (le cas échéant) :

```
<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>.
```

  Important : *Les applications AIR pour TV ne prennent pas en charge l'élément* `WindowedApplication`.

  Remarque : *il n'est pas impératif d'utiliser un fichier MXML. Vous pouvez opter pour la création d'un projet ActionScript 3.0.*

- Utilise uniquement les classes et méthodes ActionScript 3.0 prises en charge par les profils AIR `tv` et `extendedTV`. Pour plus d'informations, voir « Profils de périphérique » à la page 259.

Par ailleurs, dans le fichier XML de l'application, assurez-vous que :

- L'attribut `xmlns` de l'élément `application` est défini sur AIR 2.5 :

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

- L'élément `supportedProfiles` comprend `tv` ou `extendedTV` :

```
<supportedProfiles>tv</supportedProfiles>
```

Débogage de l'application

Vous pouvez exécuter l'application à l'aide de l'application de débogage du lanceur AIR (ADL) intégrée à Flash Builder. Procédez comme suit :

1 Sélectionnez Exécuter > Déboguer les configurations.

2 Assurez-vous que le champ Profil est défini sur Bureau.

3 Sélectionnez Exécuter > Déboguer pour activer le mode de débogage ou Exécuter > Exécuter pour exécuter l'application sans fonctionnalité de débogage.

Etant donné que vous avez défini l'élément `supportedProfiles` sur tv ou extendedTV, l'application de débogage du lanceur AIR (ADL) affiche le menu Touches de commande à distance. Ce menu permet de simuler la sélection de touches sur une télécommande.

Pour plus d'informations, voir « Débogage à distance avec Flash Builder » à la page 155.

Utilisation des extensions natives

Si votre application fait appel à une extension native, suivez les instructions de la section « Liste de tâches pour l'utilisation d'une extension native » à la page 159.

Toutefois, si une application utilise des extensions natives :

- Il est impossible de publier l'application à l'aide de Flash Builder. Pour publier l'application, faites appel à l'outil AIR Developer (ADT). Voir « Mise en package avec l'outil ADT » à la page 151.
- Il est impossible d'exécuter ou de déboguer l'application avec Flash Builder. Pour déboguer l'application sur la machine de développement, faites appel à l'application ADL. Voir « Simulation de périphérique à l'aide de l'application ADL » à la page 152

Propriétés du descripteur de l'application AIR pour TV

A l'instar des autres applications AIR, vous définissez les propriétés de base d'une application dans le fichier descripteur correspondant. Les applications associées au profil TV ignorent certaines propriétés propres au bureau, telles que la transparence et la taille des fenêtres. Les applications qui ciblent les périphériques associés au profil `extendedTV` peuvent faire appel aux extensions natives. Ces applications identifient les extensions natives utilisées dans un élément `extensions`.

Paramètres standard

Divers paramètres de descripteur d'application jouent un rôle important dans toutes les applications associées au profil TV.

Version du moteur d'exécution d'AIR requise

Indiquez la version du moteur d'exécution d'AIR requise par l'application à l'aide de l'espace de noms du fichier descripteur de l'application.

Affecté dans l'élément `application`, l'espace de noms détermine globalement les fonctionnalités dont dispose l'application. Prenons l'exemple d'une application qui utilise l'espace de noms AIR 2.5, mais l'utilisateur dispose d'une version ultérieure. Dans ce cas de figure, l'application continue à assimiler le comportement d'AIR à la version 2.5, même s'il a été modifié dans la version ultérieure d'AIR. Pour que l'application accède au nouveau comportement et aux nouvelles fonctionnalités, vous devez au préalable modifier l'espace de noms et publier une mise à jour. Les correctifs de sécurité constituent une exception notable à la règle.

Spécifiez l'espace de noms à l'aide de l'attribut `xmlns` de l'élément racine `application` :

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5 est la première version d'AIR à prendre en charge les applications TV.

Identité d'une application

Plusieurs paramètres devraient être propres à chaque application publiée, à savoir Ces paramètres incluent les éléments `id`, `name` et `filename`.

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

Version d'une application

Spécifiez la version de l'application dans l'élément `versionNumber`. Si vous définissez une valeur dans `versionNumber`, vous pouvez utiliser une séquence de trois nombres au plus, séparés par un point (« 0.1.2 », par exemple). Chaque segment du numéro de la version se compose de trois chiffres au plus. (En d'autres termes, la version la plus longue autorisée correspond à « 999.999.999 ».) Vous ne devez pas obligatoirement inclure trois segments dans le nombre. Les numéros de version « 1 » et « 1.0 » sont également valides.

Vous pouvez aussi définir le libellé de la version à l'aide de l'élément `versionLabel`. Si vous ajoutez un libellé de version, il remplace le numéro de version à l'écran.

```
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

Fichier SWF principal d'une application

Spécifiez le fichier SWF principal de l'application dans l'enfant `versionLabel` de l'élément `initialWindow`. Si vous ciblez des périphériques associés au profil TV, vous devez utiliser un fichier SWF (les applications de type HTML n'étant pas prises en charge).

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

Vous devez inclure ce fichier dans le package AIR (à l'aide de l'outil ADT ou de l'IDE). Se contenter de faire référence au nom dans le descripteur d'application n'entraîne pas l'inclusion automatique du fichier dans le package.

Propriétés de l'écran principal

Plusieurs éléments enfants de l'élément `initialWindow` contrôlent le comportement et l'aspect initiaux de l'écran principal de l'application. Bien que la plupart de ces propriétés soient ignorées dans les périphériques associés aux profils TV, vous disposez de l'élément `fullScreen` :

- **`fullScreen`** : indique si l'application doit occuper l'écran du périphérique dans son intégralité ou le partager avec le chrome standard du système d'exploitation.

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

Élément visible

L'élément `visible` est un élément enfant de l'élément `initialWindow`. AIR pour TV ignore l'élément `visible`, car le contenu de votre application est toujours visible sur les périphériques AIR pour TV.

Définissez toutefois l'élément `visible` sur `true` si votre application cible également les périphériques de bureau.

Sur les périphériques de bureau, la valeur de cet élément est par défaut définie sur `false`. Par conséquent, si vous n'incluez pas l'élément `visible`, le contenu de l'application n'est pas visible sur les périphériques de bureau. Bien que vous puissiez utiliser la classe ActionScript NativeWindow pour rendre le contenu visible sur les périphériques de bureau, les profils de périphériques TV ne prennent pas en charge la classe NativeWindow. Si vous tentez d'utiliser la classe NativeWindow dans une application s'exécutant sur un périphérique AIR pour TV, le chargement de l'application échoue. L'appel d'une méthode de la classe NativeWindow n'a aucun effet ; une application qui fait appel à cette classe ne se charge pas sur un périphérique AIR pour TV.

Profils pris en charge

Si l'application n'a de sens que sur un téléviseur, vous pouvez interdire son installation sur d'autres types de périphériques informatiques. Excluez les autres profils de la liste des profils pris en charge de l'élément `supportedProfiles` :

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Si une application fait appel à une extension native, incluez uniquement le profil `extendedTV` à la liste des profils pris en charge :

```
<supportedProfiles>extendedTV</supportedProfiles>
```

Si vous omettez l'élément `supportedProfiles`, il est considéré comme acquis que l'application prend en charge tous les profils.

N'incluez pas *uniquement* le profil `tv` dans la liste `supportedProfiles`. Certains périphériques TV exécutent toujours AIR pour TV dans un mode qui correspond au profil `extendedTV`. Ce comportement permet à AIR pour TV d'exécuter des applications faisant appel à des extensions natives. Si votre élément `supportedProfiles` spécifie uniquement `tv`, cela indique que votre contenu est incompatible avec le mode AIR pour TV pour `extendedTV`. Par conséquent, certains périphériques TV ne chargent pas une application qui spécifie uniquement le profil `tv`.

Pour consulter la liste des classes ActionScript prises en charge par les profils `tv` et `extendedTV`, voir « Fonctionnalités des différents profils » à la page 260.

Extensions natives requises

Les applications qui prennent en charge le profil `extendedTV` peuvent recourir aux extensions natives.

Déclarez toutes les extensions natives auxquelles fait appel l'application AIR dans le descripteur d'application à l'aide des éléments `extensions` et `extensionID`. L'exemple suivant illustre la syntaxe permettant de spécifier deux extensions natives requises :

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

Si une extension n'est pas répertoriée, l'application ne peut pas l'utiliser.

La valeur de l'élément `extensionID` est identique à celle de l'élément `id` dans le fichier descripteur de l'extension. Le fichier descripteur de l'extension est un fichier XML appelé extension.xml. Il est mis en package dans le fichier ANE fourni par le constructeur du périphérique.

Si vous répertoriez une extension dans l'élément `extensions` alors que le périphérique AIR pour TV ne dispose pas de cette extension, l'exécution de l'application échoue. Il existe toutefois une exception à cette règle : lorsque le fichier ANE que vous mettez en package avec votre application AIR pour TV dispose d'une version temporaire de l'extension. Dans ce cas, l'application s'exécute et utilise la version temporaire de l'extension. La version temporaire dispose de code ActionScript, mais pas de code natif.

Icônes d'une application

Chaque périphérique TV a des exigences qui lui sont propres en matière d'icônes d'application. Le constructeur de périphérique stipule par exemple les éléments suivants :

- Icônes obligatoires et taille de ces dernières
- Types de fichiers obligatoires et conventions de dénomination
- Accès de l'application aux icônes (mise en package des icônes avec l'application, par exemple)
- Spécification ou non des icônes dans un élément `icon` du fichier descripteur de l'application
- Comportement si l'application ne fournit pas d'icônes

Pour plus d'informations, consultez le constructeur du périphérique.

Paramètres ignorés

Les applications pour périphériques TV ignorent les paramètres relatifs aux fonctionnalités d'applications mobiles, de fenêtre native ou d'un système d'exploitation de bureau. Les paramètres ignorés sont les suivants :

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

Mise en package d'une application AIR pour TV

Mise en package avec l'outil ADT

L'outil de ligne de commande ADT AIR permet de mettre en package une application AIR pour TV. A partir de la version 2.5 du kit SDK d'AIR, ADT prend en charge la mise en package des périphériques AIR pour TV. Avant la mise en package, compilez tout le code ActionScript et MXML. Vous devez également disposer d'un certificat de signature du code. Pour créer un certificat, faites appel à la commande ADT -certificate.

Pour consulter des informations de référence détaillées sur les commandes et options de l'outil ADT, voir « Outil AIR Developer (ADT) » à la page 174.

Création d'un package AIR

La commande ADT package permet de créer un package AIR :

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

L'exemple part du principe que :

- Le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Voir « Variables d'environnement path » à la page 321.)
- Le certificat codesign.p12 réside dans le répertoire parent à partir duquel vous exécutez la commande ADT.

Exécutez la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf et un répertoire d'icônes.

Si vous exécutez la commande comme indiqué, l'outil ADT vous invite à entrer le mot de passe associé au keystore. Certaines interfaces de commande n'affichent pas les caractères du mot de passe saisis. Appuyez simplement sur Entrée une fois la saisie terminée. Vous disposez également du paramètre `storepass` pour inclure le mot de passe dans la commande ADT.

Création d'un package AIRN

Si l'application AIR pour TV fait appel à une extension native, créez un package AIRN plutôt qu'un package AIR. Pour créer un package AIRN, utilisez la commande ADT package en définissant le type cible sur `airn`.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml
myApp.swf icons -extdir C:\extensions
```

L'exemple part du principe que :

- Le chemin d'accès à l'outil ADT figure dans la définition path de l'interface de commande de ligne de commande. (Voir « Variables d'environnement path » à la page 321.)
- Le certificat codesign.p12 réside dans le répertoire parent à partir duquel vous exécutez la commande ADT.
- Le paramètre `-extdir` identifie le répertoire qui contient les fichiers ANE utilisés par l'application.

  Ces fichiers ANE contiennent une version de simulation ou une implémentation uniquement temporaire de l'extension ActionScript. La version de l'extension qui contient le code natif est installée sur le périphérique AIR pour TV.

Exécutez la commande à partir du répertoire qui contient les fichiers de l'application. Dans l'exemple illustré, les fichiers de l'application sont myApp-app.xml (fichier descripteur de l'application), myApp.swf et un répertoire d'icônes.

Si vous exécutez la commande comme indiqué, l'outil ADT vous invite à entrer le mot de passe associé au keystore. Certaines interfaces de commande n'affichent pas les caractères du mot de passe saisis. Appuyez simplement sur Entrée une fois la saisie terminée. Vous disposez par ailleurs du paramètre `storepass` pour inclure le mot de passe dans la commande.

Vous pouvez également créer un fichier AIRI pour une application AIR pour TV qui fait appel à des extensions natives. Mise à part l'absence de signature, le fichier AIRI est identique au fichier AIRN. Exemple :

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Lorsque vous êtes prêt à signer l'application, vous pouvez ensuite créer un fichier AIRN à partir du fichier AIRI :

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Pour plus d'informations, voir Développement d'applications natives pour Adobe AIR.

Mise en package dans Flash Builder ou Flash Professional

Flash Professional et Flash Builder permettent de publier ou d'exporter des packages AIR sans avoir à exécuter vous-même l'outil ADT. La documentation de ces programmes décrit la procédure de création d'un package AIR d'application AIR.

Néanmoins, seul ADT peut actuellement créer des packages AIRN, les packages d'application pour les applications AIR pour TV qui font appel à des extensions natives.

Pour plus d'informations, consulter les références suivantes :

- Création de packages d'application AIR
- Publication pour Adobe Air

Débogage d'applications AIR pour TV

Simulation de périphérique à l'aide de l'application ADL

La méthode la plus rapide et la plus simple pour tester et déboguer la plupart des fonctionnalités d'une application consiste à exécuter cette dernière sur l'ordinateur de développement à l'aide de l'application de débogage du lanceur AIR (ADL).

L'application ADL utilise l'élément `supportedProfiles` du descripteur d'application pour sélectionner le profil à utiliser. Plus précisément :

- Si plusieurs profils sont recensés, l'application ADL utilise le premier de la liste.
- Vous disposez du paramètre `-profile` de l'application ADL pour sélectionner l'un des autres profils de la liste `supportedProfiles`.
- Si le descripteur d'application ne contient pas d'élément `supportedProfiles`, vous pouvez spécifier n'importe quel profil dans l'argument `-profile`.

Utilisez par exemple la commande suivante pour lancer une application de simulation du profil `tv` :

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

Si vous simulez le profil `tv` ou `extendedTV` sur le bureau à l'aide de l'application ADL, l'application s'exécute dans un environnement plus proche d'un périphérique cible. Exemple :

- Les API ActionScript qui ne font pas partie du profil dans l'argument `-profile` ne sont pas disponibles.

- L'application ADL autorise la saisie de contrôles d'entrée de périphériques, tels que les télécommandes, par le biais de commandes de menu.
- En spécifiant `tv` ou `extendedTV` dans l'argument `-profile`, l'application ADL est en mesure de simuler la classe StageVideo sur le bureau.
- Spécifier `extendedTV` dans l'argument `-profile` autorise l'application à utiliser des versions de simulation ou des implémentations d'extensions natives temporaires avec le fichier AIRN de l'application.

Toutefois, étant donné que l'application ADL exécute l'application sur le bureau, tester une application AIR pour TV à l'aide d'ADL est soumis à certaines restrictions :

- Cette méthode ne reflète pas les performances de l'application sur le périphérique. Exécutez des tests de performance sur le périphérique cible.
- Elle ne simule pas les restrictions de l'objet StageVideo. Vous utilisez généralement la classe StageVideo au lieu de la classe Video pour lire une vidéo si vous ciblez les périphériques AIR pour TV. La classe StageVideo exploite les performances optimisées du matériel du périphérique, mais souffre de restrictions au niveau de l'affichage. L'application ADL n'est pas affectée par ces restrictions lorsqu'elle lit la vidéo sur le bureau. Testez par conséquent la lecture de la vidéo sur le périphérique cible.
- Il est impossible de simuler le code natif d'une extension native. Vous pouvez toutefois spécifier le profil `extendedTV`, qui prend en charge les extensions natives, dans l'argument `-profile` de l'application ADL. L'application ADL permet de procéder au test avec la version de simulation ou l'implémentation uniquement temporaire de l'extension ActionScript intégrée au package ANE. En règle générale, l'extension correspondante installée sur le périphérique comprend toutefois également du code natif. Pour procéder au test en utilisant l'extension avec le code natif, exécutez l'application sur le périphérique cible.

Pour plus d'informations, voir « Application de débogage du lanceur AIR (ADL) » à la page 168.

Utilisation des extensions natives

Si l'application fait appel à des extensions natives, la commande ADL s'apparente à l'exemple suivant :

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

L'exemple part du principe que :

- Le chemin d'accès à l'application ADL figure dans la définition path de l'interface de commande de ligne de commande. (Voir « Variables d'environnement path » à la page 321.)
- Le répertoire actif contient les fichiers de l'application. Ces fichiers incluent les fichiers SWF et le fichier descripteur de l'application (soit myApp-app.xml dans l'exemple présent).
- Le paramètre `-extdir` identifie un répertoire qui contient un sous-répertoire par extension native utilisée par l'application. Chacun de ces sous-répertoires contient le fichier ANE *extrait du package* d'une extension native. Exemple :

```
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
```

Ces fichiers ANE extraits du package contiennent une version de simulation ou une implémentation ActionScript uniquement temporaire de l'extension native. La version de l'extension qui contient le code natif est installée sur le périphérique AIR pour TV.

Pour plus d'informations, voir Développement d'applications natives pour Adobe AIR.

Saisie de contrôle

L'application ADL simule les boutons d'une télécommande sur un périphérique TV. Vous pouvez envoyer ces saisies de bouton au périphérique simulé à l'aide du menu qui s'affiche au lancement de l'application ADL par le biais de l'un des profils TV.

Taille de l'écran

Vous pouvez tester l'application sur des écrans de diverses tailles en définissant le paramètre ADL `-screensize`. Vous pouvez spécifier une chaîne contenant les quatre valeurs qui représentent la largeur et la hauteur des écrans standard et agrandis.

Veillez à toujours spécifier les dimensions (en pixels) correspondant au format Portrait, c'est-à-dire de spécifier une valeur de largeur inférieure à la valeur de hauteur. Exemple :

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

Instructions trace

Si vous exécutez l'application TV sur le bureau, la sortie trace est imprimée sur le débogueur ou la fenêtre du terminal utilisé pour le lancement de l'application ADL.

Débogage à distance avec Flash Professional

Vous pouvez déboguer à distance l'application AIR pour TV à l'aide de Flash Professional lorsqu'elle s'exécute sur le périphérique cible. La procédure de configuration du débogage à distance varie toutefois selon le périphérique. Le kit de développement pour matériel MAX 2010 d'Adobe® AIR® pour TV contient par exemple des instructions détaillées relatives au périphérique.

Quel que soit le périphérique cible utilisé, procédez comme suit pour préparer le débogage à distance :

1. Sur l'onglet Flash de la boîte de dialogue Paramètres de publication, sélectionnez Autoriser le débogage.

   Cette option permet à Flash Professional d'inclure des informations de débogage dans tous les fichiers SWF créés à partir du fichier FLA.

2. Dans l'onglet Signature de la boîte de dialogue Paramètres d'Adobe AIR (Paramètres de l'application Adobe AIR et du programme d'installation), sélectionnez l'option de préparation d'un fichier AIR Intermediate (AIRI).

   Si votre application est toujours en cours de développement, l'utilisation d'un fichier AIRI, qui ne nécessite aucune signature numérique, est suffisante.

3. Publiez l'application en créant le fichier AIRI.

Les dernières étapes de la procédure consistent à installer et exécuter l'application sur le périphérique cible. Elles varient toutefois selon le périphérique.

Débogage à distance avec Flash Builder

Vous pouvez déboguer à distance l'application AIR pour TV à l'aide de Flash Builder lorsqu'elle s'exécute sur le périphérique cible. La procédure de configuration du débogage à distance varie toutefois selon le périphérique.

Quel que soit le périphérique cible utilisé, procédez comme suit pour préparer le débogage à distance :

1. Sélectionnez Projet > Exporter vers une version validée. Sélectionnez l'option de préparation d'un fichier AIR Intermediate (AIRI).

   Si votre application est toujours en cours de développement, l'utilisation d'un fichier AIRI, qui ne nécessite aucune signature numérique, est suffisante.

2. Publiez l'application en créant le fichier AIRI.

3. Modifiez le package AIRI de l'application de sorte qu'il contienne les fichiers SWF qui stockent les informations de débogage.

   Les fichiers SWF qui contiennent les informations de débogage résident dans le sous-répertoire bin-debug du répertoire de projet Flash Builder de l'application. Remplacez les fichiers SWF du package AIRI par les fichiers SWF du sous-répertoire bin-debug.

Sur une machine de développement Windows, procédez comme suit pour effectuer ce remplacement :

1. Remplacez l'extension .airi du fichier du package AIRI par l'extension.zip.

2. Extrayez le contenu du fichier ZIP.

3. Remplacez les fichiers SWF de la structure de répertoire par les fichiers du dossier bin-debug.

4. Recompressez les fichiers dans le répertoire extrait.

5. Affectez à nouveau l'extension .airi au fichier compressé.

Sur une machine de développement Mac, la procédure de remplacement varie selon le périphérique. En règle générale, elle comprend toutefois les étapes suivantes :

1. Installation du package AIRI sur le périphérique cible

2 Remplacement des fichiers SWF résidant dans le répertoire d'installation de l'application sur le périphérique cible par les fichiers SWF issus du sous-répertoire bin-debug

   Prenons l'exemple du périphérique inclus dans le kit de développement pour matériel MAX 2010 d'Adobe AIR pour TV. Installez le package AIRI comme indiqué dans la documentation du kit de développement. Accédez ensuite au périphérique cible à l'aide de telnet sur la ligne de commande de la machine de développement Mac. Remplacez les fichiers SWF du répertoire d'installation de l'application (/opt/adobe/stagecraft/apps/<*nom application*>/) par les fichiers SWF issus du sous-répertoire bin-debug.

La procédure suivante permet de déboguer à distance une application à l'aide de Flash Builder et du périphérique associé au kit de développement pour matériel MAX 2010 d'Adobe AIR pour TV.

1 Sur l'ordinateur qui exécute Flash Builder (l'ordinateur de développement), exécutez le connecteur de périphérique d'AIR pour TV fourni avec le kit de développement pour matériel MAX 2010. Elle indique l'adresse IP de l'ordinateur de développement.

2 Sur le périphérique associé au kit de développement pour matériel, lancez l'application DevMaster, également intégrée à ce dernier.

3 Dans l'application DevMaster, entrez l'adresse IP de l'ordinateur de développement indiquée dans le connecteur de périphérique d'AIR pour TV.

4 Dans l'application DevMaster, veillez à sélectionner l'option d'activation du débogage à distance.

5 Quittez l'application DevMaster.

6 Sur l'ordinateur de développement, sélectionnez Démarrer dans le connecteur de périphérique d'AIR pour TV.

7 Sur le périphérique associé au kit de développement, démarrez une autre application. Vérifiez que des informations de trace sont affichées dans le connecteur de périphérique d'AIR pour TV.

   Si aucune information de trace n'est affichée, l'ordinateur de développement et le périphérique associé au kit de développement ne sont pas connectés. Assurez-vous que le port de l'ordinateur de développement utilisé pour les informations de trace est disponible. Vous pouvez sélectionner un autre port dans le connecteur de périphérique d'AIR pour TV. Veillez également à ce que le pare-feu autorise l'accès au port sélectionné.

Démarrez ensuite le débogueur dans Flash Builder. Procédez comme suit :

1 Dans Flash Builder, sélectionnez Exécuter > Déboguer les configurations.

2 Dans la configuration de débogage existante, qui est adaptée à un débogage local, copiez le nom du projet.

3 Dans la boîte de dialogue Déboguer les configurations, sélectionnez Application Web. Sélectionnez ensuite l'icône Créer une configuration de lancement.

4 Collez le nom du projet dans le champ Projet.

5 Dans la section URL ou chemin à lancer, désactivez l'option Utiliser la valeur par défaut. Saisissez également `about:blank` dans le champ de texte.

6 Sélectionnez Appliquer pour enregistrer les modifications.

7 Sélectionnez Déboguer pour démarrer le débogueur Flash Builder.

8 Démarrez l'application sur le périphérique associé au kit de développement.

A l'aide du débogueur Flash Builder, vous pouvez à présent définir des points d'arrêt ou examiner des variables, par exemple.

Chapitre 9 : Utilisation d'extensions natives pour Adobe AIR

Les extensions natives pour Adobe AIR fournissent des API ActionScript permettant d'accéder à des fonctionnalités propres aux périphériques programmées en code natif. Bien qu'ils puissent être indépendants, les développeurs d'extensions natives collaborent parfois avec les fabricants des périphériques.

Si vous développez une extension native, consultez le document Développement d'extensions natives pour Adobe AIR.

Une extension native est une combinaison de :

- Classes ActionScript.
- Code natif.

Néanmoins, en tant que développeur d'applications AIR faisant appel à une extension native, vous utilisez exclusivement les classes ActionScript.

Les extensions natives sont utiles dans les cas suivants :

- L'implémentation du code natif permet d'accéder à des fonctions propres à la plate-forme. Ces fonctions propres à la plate-forme ne sont pas disponibles dans les classes ActionScript intégrées. Il est en outre impossible de les implémenter dans des classes ActionScript propres à l'application. L'implémentation du code natif peut fournir cette fonctionnalité, car elle a accès au matériel et aux logiciels propres au périphérique.
- Une implémentation de code natif peut parfois s'avérer plus rapide qu'une implémentation qui utilise uniquement ActionScript.
- Grâce à l'implémentation du code natif, ActionScript peut accéder au code natif hérité.

Vous trouverez des exemples d'extensions natives sur le Pôle de développement Adobe. Par exemple, une extension native permet aux applications AIR d'accéder à la fonction de vibration d'Android. Voir Extensions natives pour Adobe AIR.

Fichiers AIR Native Extension (ANE)

Les développeurs d'extensions natives mettent en package une extension native dans un fichier ANE. Un fichier ANE est un fichier archive qui contient les bibliothèques et les ressources nécessaires à l'extension native.

Notez que pour certains périphériques, le fichier ANE contient la bibliothèque de code natif qu'utilise l'extension native. Pour d'autres, la bibliothèque de code natif est installée sur le périphérique. Dans certains cas, l'extension native ne possède pas de code natif pour un périphérique donné, elle est implémentée avec ActionScript uniquement.

En tant que développeur d'applications AIR, vous devez utiliser le fichier ANE de la manière suivante :

- Ajoutez le fichier ANE au chemin d'accès à la bibliothèque de l'application de la même façon que vous ajoutez un fichier SWC au chemin d'accès à la bibliothèque. Cette action permet à l'application de référencer les classes ActionScript de l'extension.

  Remarque : *lorsque vous compilez votre application, veillez à utiliser la liaison dynamique pour le fichier ANE. Si vous utilisez Flash Builder, spécifiez Externe dans le panneau des propriétés du chemin de génération ActionScript ; si vous utilisez la ligne de commande, spécifiez -external-library-path.*

- Mettez en package le fichier ANE avec l'application AIR.

Extensions natives ou classe NativeProcess ActionScript ?

ActionScript 3.0 fournit une classe NativeProcess. Cette classe permet à une application AIR d'exécuter des processus natifs sur le système d'exploitation hôte. Cette fonctionnalité est similaire aux extensions natives, qui permettent d'accéder aux fonctions et aux bibliothèques propres à la plate-forme. Si vous hésitez entre l'utilisation de la classe NativeProcess et l'utilisation d'une extension native, tenez compte des points suivants :

- Seul le profil AIR `extendedDesktop` prend en charge la classe NativeProcess. Ainsi, pour les applications dotées des profils AIR `mobileDevice` et `extendedMobileDevice`, les extensions natives constituent la seule option.
- Les développeurs d'extensions natives proposent souvent des implémentations natives pour diverses plates-formes, mais l'API ActionScript qu'ils fournissent est généralement la même sur toutes les plates-formes. Lors de l'utilisation de la classe NativeProcess, le code ActionScript permettant de lancer le processus natif peut varier selon les plates-formes.
- La classe NativeProcess lance un processus distinct, tandis que l'extension native exécute le même processus que l'application AIR. Par conséquent, si le blocage de code vous préoccupe, utilisez la classe NativeProcess pour plus de sécurité. Un processus distinct signifie néanmoins que vous devrez probablement gérer la communication entre les processus.

Extensions natives ou bibliothèques de classes ActionScript (fichiers SWC) ?

Un fichier SWC est une bibliothèque de classes ActionScript dans un format d'archive. Le fichier SWC contient un fichier SWF et d'autres fichiers de ressources. Le fichier SWC permet de partager des classes ActionScript au lieu de partager du code ActionScript et des fichiers de ressources individuels.

Un package d'extensions natives est un fichier ANE. À l'instar d'un fichier SWC, un fichier ANE est une bibliothèque de classes ActionScript contenant un fichier SWF et d'autres fichiers de ressources dans un format d'archive. Toutefois, la principale différence entre un fichier ANE et un fichier SWC est que seul le fichier ANE peut contenir une bibliothèque de code natif.

Remarque : *lorsque vous compilez votre application, veillez à utiliser la liaison dynamique pour le fichier ANE. Si vous utilisez Flash Builder, spécifiez Externe dans le panneau des propriétés du chemin de génération ActionScript ; si vous utilisez la ligne de commande, spécifiez -external-library-path.*

Voir aussi

A propos des fichiers SWC

Périphériques pris en charge

A partir d'AIR 3, vous pouvez utiliser des extensions natives dans les applications pour les périphériques suivants :

- Périphériques Android, à partir d'Android 2.2

- Périphériques iOS, à partir d'iOS 4.0
- Simulateur iOS, à partir d'AIR 3.3
- Blackberry Playbook
- Périphériques de bureau Windows prenant en charge AIR 3.0
- Périphériques de bureau Mac OS X prenant en charge AIR 3.0

Il n'est pas rare que la même extension native cible plusieurs plates-formes. Le fichier ANE de l'extension contient les bibliothèques ActionScript et les bibliothèques natives pour chaque plate-forme prise en charge. En règle générale, les bibliothèques ActionScript disposent des mêmes interfaces publiques pour toutes les plates-formes. Les bibliothèques natives sont nécessairement différentes.

Une extension native prend parfois en charge une plate-forme par défaut. L'implémentation de la plate-forme par défaut possède uniquement du code ActionScript. Si vous mettez en package une application pour une plate-forme que l'extension ne prend pas spécifiquement en charge, l'application fait appel à l'implémentation par défaut lors de son exécution. Par exemple, prenez le cas d'une extension disposant d'une fonction qui s'applique uniquement aux périphériques mobiles. L'extension peut également fournir une implémentation par défaut à laquelle peut faire appel une application de bureau pour simuler cette fonction.

Profils de périphérique pris en charge

Les profils AIR suivants prennent en charge les extensions natives :

- `extendedDesktop`, à partir d'AIR 3.0
- `mobileDevice`, à partir d'AIR 3.0
- `extendedMobileDevice`, à partir d'AIR 3.0

Voir aussi

Prise en charge du profil AIR

Liste de tâches pour l'utilisation d'une extension native

Pour utiliser une extension native dans votre application, procédez de la façon suivante :

1. Déclarez l'extension dans le fichier descripteur de votre application.
2. Ajoutez le fichier ANE au chemin d'accès à la bibliothèque de l'application.
3. Mettez en package l'application.

Déclaration de l'extension dans le fichier descripteur de l'application

Toutes les applications AIR possèdent un fichier descripteur de l'application. Lorsqu'une application utilise une extension native, le fichier descripteur de l'application inclut un élément `<extensions>`. Exemple :

```
<extensions>
    <extensionID>com.example.Extension1</extensionID>
    <extensionID>com.example.Extension2</extensionID>
</extensions>
```

La valeur de l'élément `extensionID` est identique à celle de l'élément `id` dans le fichier descripteur de l'extension. Le fichier descripteur de l'extension est un fichier XML appelé extension.xml. Il est mis en package dans le fichier ANE. Vous pouvez utiliser un outil d'extraction pour accéder au fichier extension.xml.

Ajout du fichier ANE au chemin d'accès à la bibliothèque de l'application

Pour compiler une application qui utilise une extension native, ajoutez le fichier ANE au chemin d'accès à la bibliothèque.

Utilisation du fichier ANE avec Flash Builder

Si votre application a recours à une extension native, ajoutez le fichier ANE associé au chemin d'accès à la bibliothèque. Vous pouvez ensuite compiler le code ActionScript à l'aide de Flash Builder.

Suivez les étapes ci-dessous, qui font appel à Flash Builder 4.5.1 :

1. Remplacez l'extension du fichier ANE, .ane, par .swc. Cette étape est nécessaire afin que Flash Builder puisse trouver le fichier.
2. Sélectionnez Projet > Propriétés pour le projet Flash Builder.
3. Sélectionnez Chemin de création Flex dans la boîte de dialogue Propriétés.
4. Dans l'onglet Chemin d'accès à la bibliothèque, sélectionnez Ajouter un fichier SWC...
5. Accédez au fichier SWC et sélectionnez Ouvrir.
6. Cliquez sur OK dans la boîte de dialogue Ajouter un fichier SWC...

   Le fichier ANE apparaît à présent dans l'onglet du chemin d'accès à la bibliothèque de la boîte de dialogue Propriétés.
7. Développez l'entrée relative au fichier SWC. Double-cliquez sur Type de lien pour ouvrir la boîte de dialogue Options de l'élément de chemin d'accès à la bibliothèque.
8. Dans la boîte de dialogue Options de l'élément de chemin d'accès à la bibliothèque, sélectionnez le type de lien Externe.

Vous pouvez à présent compiler l'application à l'aide de Projet > Générer le projet, par exemple.

Utilisation du fichier ANE avec Flash Professional

Si votre application a recours à une extension native, ajoutez le fichier ANE associé au chemin d'accès à la bibliothèque. Vous pouvez ensuite compiler le code ActionScript à l'aide de Flash Professional CS5.5. Procédez comme suit :

1. Remplacez l'extension du fichier ANE, .ane, par .swc. Cette étape est nécessaire afin que Flash Professional puisse trouver le fichier.
2. Sélectionnez Fichier > Paramètres d'ActionScript pour le fichier FLA.
3. Dans la boîte de dialogue Paramètres avancés d'ActionScript 3.0, sélectionnez l'onglet Chemin bibliothèque.

4 Cliquez sur le bouton Localiser le fichier SWC.

5 Accédez au fichier SWC et sélectionnez Ouvrir.

Le fichier SWC figure à présent dans l'onglet Chemin bibliothèque de la boîte de dialogue Paramètres avancés d'ActionScript 3.0.

6 Après avoir sélectionné le fichier SWC, sélectionnez le bouton Définir les options de liaison d'une bibliothèque.

7 Dans la boîte de dialogue Options de l'élément de chemin d'accès à la bibliothèque, sélectionnez le type de lien Externe.

Mise en package d'une application faisant appel à des extensions natives

Utilisez l'outil AIR Developer (ADT) pour mettre en package une application faisant appel à des extensions natives. Vous ne pouvez pas mettre en package l'application avec Flash Professional CS5.5 ou Flash Builder 4.5.1.

Pour plus d'informations sur l'utilisation d'ADT, voir Outil AIR Developer (ADT).

Par exemple, la commande d'ADT suivante crée un fichier DMG (fichier de programme d'installation natif pour Mac OS X) pour une application faisant appel à des extensions natives :

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

La commande suivante créé un package APK pour un périphérique Android :

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

La commande suivante crée un package iOS pour une application 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
```

Remarques importantes :

- Utilisez un package de type programme d'installation natif.
- Spécifiez le répertoire de l'extension.

- Assurez-vous que le fichier ANE prend en charge le périphérique cible de l'application.

Utilisation d'un package de type programme d'installation natif

Le package d'application doit être un programme d'installation natif. Vous ne pouvez pas créer un package AIR multiplateformes (un package .air) pour une application faisant appel à une extension native, car les extensions natives contiennent généralement du code natif. En revanche, une extension native prend généralement en charge plusieurs plates-formes natives avec les mêmes API ActionScript. Dans ces cas, vous pouvez recourir au même fichier ANE dans plusieurs packages de programmes d'installation natifs.

Le tableau suivant récapitule la valeur à utiliser pour l'option `-target` de la commande d'ADT :

Plate-forme cible de l'application	**-target**
Périphériques de bureau Mac OS X ou Windows	-target native -target bundle
Android	-target apk ou autres cibles de package Android.
iOS	-target ipa-ad-hoc ou autres cibles de package iOS
Simulateur iOS	-target ipa-test-interpreter-simulator -target ipa-debug-interpreter-simulator

Spécification du répertoire de l'extension

Utilisez l'option d'ADT `-extdir` pour indiquer à l'outil ADT le répertoire contenant les extensions natives (fichiers ANE).

Pour plus d'informations sur cette option, voir « Options associées aux fichiers et chemins » à la page 191.

Vérification de la prise en charge du périphérique cible de l'application par le fichier ANE

Lorsqu'il fournit un fichier ANE, le développeur de l'extension native vous indique les plates-formes prises en charge par l'extension. Vous pouvez également utiliser un outil d'extraction pour accéder au contenu du fichier ANE. Les fichiers extraits comprennent un répertoire pour chaque plate-forme prise en charge.

Il est important de connaître les plates-formes prises en charge par l'extension lors de la mise en package de l'application qui utilise le fichier ANE. Tenez compte des règles suivantes :

- Pour créer un package d'application Android, le fichier ANE doit inclure la plate-forme `Android-ARM`. Par ailleurs, le fichier ANE doit inclure la plate-forme par défaut et au moins une autre plate-forme.
- Pour créer un package d'application iOS, le fichier ANE doit inclure la plate-forme `iPhone-ARM`. Par ailleurs, le fichier ANE doit inclure la plate-forme par défaut et au moins une autre plate-forme.
- Pour créer un package de l'application Simulateur iOS, le fichier ANE doit inclure la plate-forme `iPhone-x86`.
- Pour créer un package d'application Mac OS X, le fichier ANE doit inclure la plate-forme `MacOS-x86` . Par ailleurs, le fichier ANE doit inclure la plate-forme par défaut et au moins une autre plate-forme.
- Pour créer un package d'application Windows, le fichier ANE doit inclure la plate-forme `Windows-x86` . Par ailleurs, le fichier ANE doit inclure la plate-forme par défaut et au moins une autre plate-forme.

Chapitre 10 : Compilateurs ActionScript

Pour pouvoir inclure du code ActionScript et MXML dans une application AIR, vous devez le compiler. Si vous utilisez un IDE (Integrated Development Environment), tel qu'Adobe Flash Builder ou Adobe Flash Professional, il gère la compilation en arrière-plan. Si vous ne faites pas appel à un IDE ou que vous utilisez un script de création, vous pouvez toutefois appeler les compilateurs ActionScript à partir de la ligne de commande pour créer des fichiers SWF.

Présentation des outils de ligne de commande d'AIR intégrés au kit SDK Flex

Chaque outil de ligne de commande que vous utilisez pour créer une application AIR appelle l'outil de création d'applications correspondant :

- amxmlc appelle mxmlc pour compiler les classes d'application.
- acompc appelle compc pour compiler les classes de bibliothèque et de composant.
- aasdoc appelle asdoc pour générer des fichiers de documentation à partir des commentaires du code source.

La seule différence entre les versions AIR et Flex des outils, c'est que la version AIR charge les options de configuration à partir du fichier air-config.xml et non du fichier flex-config.xml

Les outils du kit SDK Flex et leurs options de ligne de commande sont décrits en détail dans la Documentation Flex. Les outils du kit SDK de Flex sont décrits dans les grandes lignes, afin de vous permettre de vous familiariser avec eux et de mettre en évidence les différences entre la création d'applications Flex et la création d'applications AIR.

Voir aussi

« Création d'une première application de bureau AIR à l'aide du kit SDK de Flex » à la page 38

Configuration des compilateurs

En règle générale, vous spécifiez les options de compilation sur la ligne de commande et par le biais d'un ou de plusieurs fichiers de configuration. Le fichier de configuration global du kit SDK Flex contient les valeurs par défaut utilisées à chaque exécution des compilateurs. Vous pouvez le modifier en fonction de votre environnement de développement. Le répertoire frameworks de l'installation du kit SDK Flex contient deux fichiers de configuration Flex globaux. Utilisez le fichier air-config.xml lorsque vous exécutez le compilateur amxmlc. Il configure le compilateur pour AIR en incluant les bibliothèques AIR. Utilisez le fichier flex-config.xml lorsque vous exécutez mxmlc.

Les valeurs de configuration par défaut sont adaptées pour s'initier au fonctionnement de Flex et AIR. Lorsque vous vous lancez dans un projet de grande envergure, examinez plus en détail toutes les options disponibles. Vous pouvez définir des valeurs propres à un projet pour les options de compilation dans un fichier de configuration local, qui est prioritaire sur les valeurs globales dans tout projet donné.

Remarque : *aucune option de compilation ne s'utilise spécifiquement pour les applications AIR. Vous devez cependant référencer les bibliothèques AIR lors de la compilation d'une application AIR. En règle générale, ces bibliothèques sont référencées dans un fichier de configuration au niveau du projet, dans un fichier réservé à un outil de création tel qu'Ant ou directement sur la ligne de commande.*

Compilation de fichiers sources MXML et ActionScript pour AIR

Vous avez la possibilité de compiler les actifs Adobe® ActionScript® 3.0 et MXML de l'application AIR à l'aide du compilateur MXML de ligne de commande (amxmlc). (Il est inutile de compiler les applications HTML. Pour compiler un fichier SWF dans Flash Professional, il suffit de publier le clip dans un fichier SWF.)

Pour appeler amxmlc, utilisez la syntaxe de ligne de commande de base suivante :

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

où *[compiler options]* indique les options de ligne de commande utilisées pour compiler l'application AIR.

La commande amxmlc appelle le compilateur mxmlc Flex standard avec un paramètre supplémentaire, `+configname=air`. Ce paramètre indique au compilateur d'utiliser le fichier air-config.xml à la place du fichier flex-config.xml. Mise à part cette différence, l'emploi d'amxmlc est en tout point identique à celui de mxmlc.

Le compilateur charge le fichier de configuration air-config.xml indiquant les bibliothèques AIR et Flex généralement requises pour compiler une application AIR. Une autre solution consiste à utiliser un fichier de configuration local, de niveau projet, destiné à remplacer des options disponibles dans la configuration globale ou à lui en ajouter d'autres. En général, le moyen le plus simple de créer un fichier de configuration local consiste à modifier une copie de la version globale. Vous pouvez charger le fichier local à l'aide de l'option `-load-config` :

-load-config=project-config.xml Remplace les options globales.

-load-config+=project-config.xml Ajoute des valeurs supplémentaires aux options globales admettant plus d'une valeur, telles que l'option `-library-path`. Les options globales admettant une seule valeur sont remplacées.

Si vous appliquez une convention d'appellation spéciale aux fichiers de configuration, le compilateur amxmlc charge automatiquement le fichier local. Si, par exemple, le fichier MXML principal s'intitule `RunningMan.mxml`, nommez le fichier de configuration local `RunningMan-config.xml`. A présent, pour compiler l'application, vous devez uniquement saisir :

```
amxmlc RunningMan.mxml
```

`RunningMan-config.xml` est chargé automatiquement, car son nom de fichier correspond à celui du fichier MXML compilé.

Exemples d'utilisation du compilateur amxmlc

Les exemples suivants illustrent l'utilisation du compilateur amxmlc. (Seules les ressources ActionScript et MXML de l'application doivent être compilées.)

Compilation d'un fichier MXML AIR :

```
amxmlc myApp.mxml
```

Compilation et définition d'un nom de sortie :

```
amxmlc –output anApp.swf -- myApp.mxml
```

Compilation d'un fichier ActionScript AIR :

```
amxmlc myApp.as
```

Spécification d'un fichier de configuration pour le compilateur :

```
amxmlc –load-config config.xml -- myApp.mxml
```

Ajout d'options supplémentaires provenant d'un autre fichier de configuration :

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

Ajout de bibliothèques sur la ligne de commande (en plus des bibliothèques déjà présentes dans le fichier de configuration) :

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

Compilation d'un fichier MXML AIR sans recourir à un fichier de configuration (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
```

Compilation d'un fichier MXML AIR sans recourir à un fichier de configuration (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
```

Compilation d'un fichier MXML AIR afin d'utiliser une bibliothèque partagée à l'exécution :

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --
myApp.mxml
```

Compilez un fichier MXML AIR pour utiliser un fichier ANE (veillez à utiliser `-external-library-path` pour le fichier ANE) :

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

Compilation à partir du code Java (dont le chemin de classe est défini de manière à inclure `mxmlc.jar`) :

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional
compiler options] -- myApp.mxml
```

L'option flexlib identifie l'emplacement du répertoire des structures SDK Flex, ce qui permet au compilateur de localiser le fichier flex_config.xml.

Compilation à partir du code Java (sans chemin de classe défini) :

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air
[additional compiler options] -- myApp.mxml
```

Pour appeler le compilateur via Apache Ant (une tâche Java est utilisée dans cet exemple pour exécuter 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>
```

Compilation d'un composant ou d'une bibliothèque de code AIR (Flex)

Servez-vous du compilateur de composants (acompc) afin de compiler des bibliothèques et des composants indépendants AIR. Le compilateur de composants acompc se comporte comme le compilateur amxmlc, à l'exception des points suivants :

- Vous devez spécifier les classes au sein de la base de code à inclure dans la bibliothèque ou le composant.
- acompc ne recherche pas automatiquement un fichier de configuration local. Pour utiliser un fichier de configuration de projet, vous devez spécifier l'option –load-config.

La commande acompc appelle le compilateur de composants compc Flex standard, mais elle charge ses options de configuration à partir du fichier `air-config.xml` au lieu du fichier `flex-config.xml`.

Fichier de configuration du compilateur de composants

Faites appel à un fichier de configuration local afin d'éviter de saisir (et peut-être de mal orthographier) le chemin source et les noms de classes sur la ligne de commande. Insérez l'option -load-config à la ligne de commande acompc afin de charger le fichier de configuration local.

L'exemple suivant illustre une configuration de développement d'une bibliothèque dotée de deux classes, ParticleManager et Particle, toutes deux contenues dans le package `com.adobe.samples.particles`. Les fichiers de classe se trouvent dans le dossier `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>
```

Pour compiler la bibliothèque à l'aide du fichier de configuration intitulé `ParticleLib-config.xml`, saisissez :

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Pour exécuter la même commande uniquement sur la ligne de commande, saisissez :

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Tapez la commande entière sur une ligne ou utilisez le caractère de suite de ligne sur le shell de commande.)

Exemples utilisant acompc

Dans ces exemples, il est supposé que vous utilisez un fichier de configuration intitulé `myLib-config.xml`.

Compilation d'un composant ou d'une bibliothèque AIR :

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Compilation d'une bibliothèque partagée à l'exécution :

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Notez qu'avant d'exécuter la commande, vous devez vérifier que le dossier lib existe et qu'il est vide.)

Chapitre 11 : Application de débogage du lanceur AIR (ADL)

L'application de débogage du lanceur AIR (ADL) permet d'exécuter à la fois des applications SWF et HTML lors de la phase de développement. Grâce à ADL, vous pouvez exécuter une application sans la mettre en package et l'installer au préalable. Par défaut, ADL utilise un moteur d'exécution fourni avec le kit SDK. Autrement dit, il est inutile d'installer le moteur d'exécution séparément pour se servir d'ADL.

ADL imprime les instructions trace et les erreurs d'exécution au format de sortie standard, mais ne prend pas en charge les points d'arrêt ou d'autres fonctions de débogage. Vous pouvez utiliser l'outil Flash Debugger (ou un environnement de développement intégré tel que Flash Builder) pour résoudre les problèmes plus complexes.

Remarque : *si vos instructions* `trace()` *ne s'affichent pas sur la console, assurez-vous de ne pas avoir spécifié* `ErrorReportingEnable` *ou* `TraceOutputFileEnable` *dans le fichier mm.cfg. Pour plus d'informations sur l'emplacement propre à la plate-forme de ce fichier, voir Modification du fichier mm.cfg.*

AIR prend directement en charge le débogage. Vous n'avez donc pas besoin d'une version de débogage du moteur d'exécution (contrairement à Adobe® Flash® Player). Pour effectuer le débogage sur la ligne de commande, vous utilisez le débogueur de Flash et l'application de débogage du lanceur AIR (ADL).

Le débogueur de Flash est distribué dans le répertoire du kit SDK Flex. Les versions natives, telles que fdb.exe sous Windows, résident dans le sous-répertoire bin. La version Java se trouve dans le sous-répertoire lib. L'application de débogage du lanceur AIR (ADL), adl.exe, figure dans le répertoire bin de l'installation du kit SDK Flex (il n'existe pas de version propre à Java).

Remarque : *il est impossible de démarrer une application AIR directement à l'aide du débogueur de Flash (fdb), car celui-ci tente de la lancer à l'aide de Flash Player. Laissez plutôt l'application AIR se connecter à une session fdb en cours.*

Utilisation de l'application ADL

Pour exécuter une application à l'aide d'ADL, utilisez la syntaxe suivante :

```
adl application.xml
```

où *application.xml* est le fichier descripteur de l'application.

La syntaxe complète d'ADL est la suivante :

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(Les éléments entre crochets, [], sont facultatifs.)

-runtime *RépertoireMoteurExécution* Indique le répertoire contenant le moteur d'exécution à utiliser. Si vous ne le précisez pas, le répertoire du moteur d'exécution situé dans le même kit SDK que le programme ADL est utilisé. Si vous déplacez ADL hors de son dossier SDK, spécifiez le répertoire du moteur d'exécution. Sous Windows et Linux, indiquez le répertoire contenant le dossier `Adobe AIR`. Sous Mac OS X, spécifiez le répertoire contenant `Adobe AIR.framework`.

-pubid *IdentifiantEditeur* Affecte la valeur indiquée comme identifiant d'éditeur de l'application AIR pour cette exécution. L'utilisation d'un ID d'éditeur temporaire vous permet de tester les fonctions d'une application AIR (telles que la communication via une connexion locale) nécessitant ce type d'identifiant afin d'identifier une application de manière unique. Depuis la version 1.5.3 d'AIR, vous pouvez également stipuler l'identifiant d'éditeur dans le fichier descripteur de l'application (n'utilisez donc pas ce paramètre).

***Remarque :** depuis la version 1.5.3 d'AIR, un identifiant d'éditeur n'est plus automatiquement calculé et affecté à une application AIR. Vous pouvez stipuler un identifiant d'éditeur lorsque vous créez une mise à jour d'application AIR existante, mais les nouvelles applications ne nécessitent pas - et ne devraient pas comporter - d'identifiant d'éditeur.*

-nodebug Désactive la prise en charge du débogage. Si cette option est utilisée, le processus de l'application ne peut pas se connecter au programme Flash Debugger et les boîtes de dialogue relatives aux exceptions non gérées sont masquées. (Toutefois, les instructions trace sont toujours imprimées dans la fenêtre de la console.) La désactivation de la fonction de débogage permet d'accélérer l'exécution de votre application et d'émuler plus étroitement le mode d'exécution d'une application installée.

-atlogin Simule le lancement de l'application lors de la connexion. Cet indicateur permet de tester la logique applicative qui s'exécute uniquement lorsqu'une application est configurée de sorte à démarrer lorsque l'utilisateur se connecte. Lors de l'utilisation de `-atlogin`, la propriété `reason` de l'objet InvokeEvent envoyée à l'application correspond à `login`, et non à `standard` (à moins que l'application soit déjà en cours d'exécution).

-profile *NomProfil* L'application ADL débogue l'application avec le profil indiqué. La valeur *`NomProfil`* gère l'une des valeurs suivantes :

- desktop
- extendedDesktop
- mobileDevice

Si le descripteur de l'application comprend un élément `supportedProfiles`, le profil spécifié avec l'indicateur `-profile` doit figurer dans la liste prise en charge. Si vous n'utilisez pas l'indicateur `-profile`, le premier profil du descripteur de l'application fait office de profil actif. Si le descripteur de l'application ne comprend pas d'élément `supportedProfiles` et que vous n'utilisez pas l'indicateur `-profile`, le profil *desktop* est utilisé.

Pour plus d'informations, voir « supportedProfiles » à la page 252 et « Profils de périphérique » à la page 259.

-screensize *valeur* Taille de l'écran simulée à utiliser lors de l'exécution d'une application associée au profil mobileDevice sur le bureau. Indiquez la taille de l'écran en sélectionnant le type d'écran prédéfini ou saisissez les dimensions (en pixels) de la largeur et de la hauteur standard correspondant au mode Portrait, plus la largeur et la hauteur en plein écran. Pour spécifier la valeur par type, utilisez l'un des types d'écran prédéfinis suivants :

Type d'écran	Largeur x hauteur standard	Largeur x hauteur standard en plein écran
480	720 x 480	720 x 480
720	1 280 x 720	1 280 x 720
1 080	1 920 x 1 080	1 920 x 1 080
Droid	480 x 816	480 x 854

Type d'écran	Largeur x hauteur standard	Largeur x hauteur standard en plein écran
FWQVGA	240 x 432	240 x 432
FWVGA	480 x 854	480 x 854
HVGA	320 x 480	320 x 480
iPad	768 x 1 004	768 x 1 024
iPadRetina	1 536 x 2 008	1 536 x 2 048
iPhone	320 x 460	320 x 480
iPhoneRetina	640 x 920	640 x 960
iPhone5Retina	640 x 1 096	640 x 1 136
iPhone6	750 x 1 294	750 x 1 334
iPhone6Plus	1 242 x 2 148	1 242 x 2 208
iPod	320 x 460	320 x 480
iPodRetina	640 x 920	640 x 960
iPod5Retina	640 x 1 096	640 x 1 136
NexusOne	480 x 762	480 x 800
QVGA	240 x 320	240 x 320
SamsungGalaxyS	480 x 762	480 x 800
SamsungGalaxyTab	600 x 986	600 x 1 024
WQVGA	240 x 400	240 x 400
WVGA	480 x 800	480 x 800

Pour spécifier directement les dimensions de l'écran en pixels, utilisez le format suivant :

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Veillez à toujours spécifier les dimensions (en pixels) correspondant au format Portrait, c'est-à-dire de spécifier une valeur de largeur inférieure à la valeur de hauteur. Spécifiez par exemple comme suit l'écran NexusOne :

```
-screensize 480x762:480x800
```

-extdir *extension-directory* Répertoire dans lequel le moteur d'exécution doit rechercher les extensions natives. Ce répertoire contient un sous-répertoire pour chaque extension native qu'utilise l'application. Chacun de ces sous-répertoires contient le fichier ANE *extrait du package* d'une extension. Exemple :

```
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
```

Lors de l'utilisation du paramètre -extdir, tenez compte des points suivants :

- La commande ADL requiert que chacun des répertoires spécifiés possède l'extension de nom de fichier .ane. La partie du nom de fichier qui précède le suffixe « .ane » peut néanmoins correspondre à n'importe quel nom de fichier valide. Il n'est toutefois *pas* impératif qu'elle corresponde à la valeur de l'élément `extensionID` du fichier descripteur de l'application.
- Vous pouvez spécifier plusieurs fois le paramètre `-extdir`.
- L'outil ADT et l'outil ADL traitent différemment l'utilisation du paramètre `-extdir`. Dans ADT, ce paramètre spécifie un répertoire contenant les fichiers ANE.
- Vous pouvez également utiliser la variable d'environnement `AIR_EXTENSION_PATH` pour spécifier les répertoires de l'extension. Voir « Variables d'environnement ADT » à la page 198.

application.xml Fichier descripteur d'application. Voir « Fichiers descripteurs d'applications AIR » à la page 217. Le descripteur d'application est l'unique paramètre requis par ADL et, dans la plupart des cas, l'unique paramètre nécessaire.

RépertoireRacine Indique le répertoire racine de l'application à exécuter. S'il n'est pas spécifié, c'est le répertoire contenant le fichier descripteur d'application qui est utilisé.

-- ***arguments*** Toutes les chaînes de caractères figurant après « -- » sont transmises à l'application sous forme d'arguments de ligne de commande.

Remarque : *lorsque vous lancez une application AIR déjà en cours d'exécution, aucune nouvelle occurrence de l'application n'est ouverte. Au lieu de cela, un événement* `invoke` *est distribué à l'occurrence en cours d'exécution.*

Exemples ADL

Exécutez une application dans le répertoire actif :

```
adl myApp-app.xml
```

Exécutez une application dans un sous-répertoire du répertoire actif :

```
adl source/myApp-app.xml release
```

Exécutez une application et transmettez deux arguments de ligne de commande, « tick » et « tock » :

```
adl myApp-app.xml -- tick tock
```

Exécutez une application à l'aide d'un moteur d'exécution spécifique :

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Exécutez une application sans prise en charge du débogage :

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

Exécutez une application du profil de périphérique mobile et simulez la taille de l'écran Nexus One :

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Exécutez une application par le biais d'Apache Ant (les chemins illustrés dans l'exemple se réfèrent à 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>
```

Codes d'erreur et de sortie d'ADL

Le tableau suivant décrit les codes de sortie imprimés par ADL :

Code de sortie	Description
0	Lancement réussi. ADL se ferme après la fermeture de l'application AIR.
1	Appel réussi d'une application AIR déjà exécutée. Fermeture immédiate d'ADL.
2	Erreur d'utilisation. Les arguments transmis à ADL sont incorrects.
3	Moteur d'exécution introuvable.
4	Impossible de démarrer le moteur d'exécution. En général, cela se produit car la version spécifiée dans l'application ne correspond pas à celle du moteur d'exécution.
5	Une erreur d'origine inconnue s'est produite.
6	Fichier descripteur d'application introuvable.
7	Le contenu du descripteur de l'application est incorrect. Cette erreur indique généralement que le fichier XML n'est pas bien constitué.

Code de sortie	Description
8	Le fichier de contenu principal de l'application (spécifié dans l'élément <content> du fichier descripteur d'application) est introuvable.
9	Le fichier du contenu principal de l'application n'est pas un fichier SWF ou HTML valide.
10	L'application ne prend pas en charge le profil spécifié par le biais de l'option -profile.
11	L'argument -screensize n'est pas pris en charge dans le profil actuel.

Chapitre 12 : Outil AIR Developer (ADT)

ADT est un outil de ligne de commande multifonctions destiné au développement d'applications AIR. Il permet d'exécuter les tâches suivantes :

- Mise en package d'une application AIR en tant que fichier d'installation .air
- Mise en package d'une application AIR en tant que fichier d'installation natif (fichier d'installation .exe sous Windows, .ipa sous iOS ou .apk sous Android, par exemple)
- Mise en package d'une extension native en tant que fichier ANE (AIR Native Extension)
- Signature d'une application AIR avec un certificat numérique
- Modification (migration) de la signature numérique utilisée pour mettre à jour une application
- Identification des périphériques raccordés à un ordinateur
- Création d'un certificat développeur numérique auto-signé
- Installation, lancement et désinstallation à distance d'une application sur un périphérique mobile
- Installation et désinstallation à distance du moteur d'exécution d'AIR sur un périphérique mobile

L'outil ADT est un programme Java intégré au kit de développement d'AIR. Vous devez disposer de Java 1.5 ou ultérieur pour l'utiliser. Le kit SDK comprend un fichier de script destiné à appeler l'outil ADT. Pour utiliser ce script, l'emplacement du programme Java doit être indiqué dans la variable d'environnement path. Si le répertoire `bin` du kit SDK d'AIR est également indiqué dans la variable d'environnement path, vous pouvez saisir `adt` sur la ligne de commande, accompagné des arguments appropriés, pour appeler l'outil ADT. (Si vous ne savez pas comment configurer la variable d'environnement path, consultez la documentation du système d'exploitation. Pour plus d'informations, les procédures de configuration de path sur la plupart des systèmes informatiques sont décrites dans la section « Variables d'environnement path » à la page 321.)

L'ordinateur doit disposer d'au moins 2 Go de mémoire pour exécuter l'outil ADT. Si la mémoire disponible est inférieure à 2 Go, l'outil ADT risque de ne pas disposer de suffisamment de mémoire, en particulier lors de la mise en package d'applications pour iOS.

Sous réserve que Java et le répertoire bin du kit SDK d'AIR figurent tous deux dans la variable path, vous pouvez exécuter l'outil ADT en appliquant la syntaxe de base suivante :

```
adt -command options
```

Remarque : *la plupart des environnements de développement intégrés (notamment Adobe Flash Builder et Adobe Flash Professional) peuvent mettre en package et signer des applications AIR à votre intention. En règle générale, si vous utilisez déjà un tel environnement de développement, il est inutile d'effectuer ces tâches courantes par le biais de l'outil ADT. En revanche, vous devrez peut-être vous servir de l'outil ADT en tant qu'outil de ligne de commande pour exécuter des fonctions qui ne sont pas prises en charge par l'environnement de développement intégré. Vous pouvez par ailleurs utiliser ADT en tant qu'outil de ligne de commande dans le cadre d'un processus de création automatisé.*

Commandes de l'outil ADT

Le premier argument transmis à l'outil ADT spécifie l'une des commandes suivantes.

- -package : met en package une application AIR ou une extension ANE (AIR Native Extension).

- -prepare : met en package une application AIR sous forme de fichier intermédiaire (AIRI), mais ne la signe pas. Il est impossible d'installer un fichier AIRI.
- -sign : signe un package AIRI créé à l'aide de la commande -prepare. Les commandes -prepare et -sign permettent d'exécuter la mise en package et la signature en plusieurs étapes. Vous disposez également de la commande -sign pour signer ou signer à nouveau un package ANE.
- -migrate : applique une signature de migration à un package AIR signé, vous permettant ainsi d'utiliser un nouveau certificat développeur ou un certificat renouvelé.
- -certificate : crée un certificat développeur numérique auto-signé.
- -checkstore : vérifie qu'il est possible d'accéder à un certificat numérique dans un keystore.
- -installApp : installe une application AIR sur un périphérique ou un émulateur de périphérique.
- -launchApp : lance une application AIR sur un périphérique ou un émulateur de périphérique.
- -appVersion : indique la version d'une application AIR actuellement installée sur un périphérique ou un émulateur de périphérique.
- -uninstallApp : désinstalle une application AIR à partir d'un périphérique ou d'un émulateur de périphérique.
- -installRuntime : installe le moteur d'exécution d'AIR sur un périphérique ou un émulateur de périphérique.
- -runtimeVersion : indique la version du moteur d'exécution d'AIR actuellement installée sur un périphérique ou un émulateur de périphérique.
- -uninstallRuntime : désinstalle le moteur d'exécution d'AIR actuellement installé sur un périphérique ou un émulateur de périphérique.
- -version : indique le numéro de la version de l'outil ADT.
- -devices : affiche des informations sur les périphériques mobiles ou les émulateurs raccordés.
- -help : affiche la liste de commandes et d'options.

De nombreuses commandes ADT partagent des ensembles associés de paramètres et d'indicateurs d'option. Ces ensembles d'options font l'objet d'une description détaillée distincte :

- « Options de signature du code de l'outil ADT » à la page 189
- « Options associées aux fichiers et chemins » à la page 191
- « Options de connexion au débogueur » à la page 192
- « Options d'extension native » à la page 193

Commande ADT package

La commande `-package` doit être exécutée à partir du répertoire principal de l'application. Elle gère les syntaxes suivantes :

Création d'un package AIR à partir des fichiers d'application du composant :

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Création d'un package natif à partir des fichiers d'application du composant :

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Création d'un package natif qui inclut une extension native à partir des fichiers d'application du composant :

```
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
```

Création d'un package natif à partir d'un fichier AIR ou AIRI :

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Création d'un package d'extensions natives à partir des fichiers d'extension native du composant :

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Remarque : *il n'est pas nécessaire de signer un fichier ANE ; les paramètres* `AIR_SIGNING_OPTIONS` *sont donc facultatifs dans cet exemple.*

AIR_SIGNING_OPTIONS Les options de signature AIR identifient le certificat utilisé pour signer un fichier d'installation AIR. Les options de signature font l'objet d'une description détaillée à la section « Options de signature du code de l'outil ADT » à la page 189.

-migrate Cet indicateur spécifie que l'application est signée avec un certificat de migration en plus du certificat spécifié par les paramètres `AIR_SIGNING_OPTIONS`. Cet indicateur n'est valide que si vous mettez en package une application de bureau sous la forme d'un programme d'installation natif et que l'application utilise une extension native. Dans les autres cas, une erreur survient. Les options de signature du certificat de migration sont spécifiées dans les paramètres **MIGRATION_SIGNING_OPTIONS**. Elles font l'objet d'une description détaillée à la section « Options de signature du code de l'outil ADT » à la page 189. L'utilisation de l'indicateur `-migrate` permet de créer une mise à jour d'une application de bureau à programme d'installation natif qui utilise une extension native et de modifier le certificat de signature du code de l'application lors de l'expiration du certificat d'origine, par exemple. Pour plus d'informations, voir « Signature d'une version mise à jour d'une application AIR » à la page 211.

L'indicateur `-migrate` de la commande `-package` est disponible dans AIR 3.6 et ultérieur.

-target Type de package à créer. Les types de package pris en charge sont les suivants :

- air : package AIR. « air » est la valeur par défaut et il est inutile de spécifier l'indicateur -target lors de la création de fichiers AIR ou AIRI.
- airn : package d'application natif pour périphériques associés au profil de télévision étendu.
- ane : package d'extensions natives AIR
- Cibles des packages Android
  - apk : package Android. Un package créé à l'aide de cette cible ne peut être installé que sur un périphérique Android et non sur un émulateur.
  - apk-captive-runtime : package Android qui inclut l'application et une version captive du moteur d'exécution AIR. Un package créé à l'aide de cette cible ne peut être installé que sur un périphérique Android et non sur un émulateur.
  - apk-debug : package Android contenant des informations de débogage complémentaires. (Les fichiers SWF de l'application doivent également être compilés avec une prise en charge du débogage.)
  - apk-emulator : package Android réservé à un émulateur sans prise en charge du débogage. (La cible apk-debug permet d'autoriser le débogage sur les émulateurs et les périphériques.)
  - apk-profile : package Android qui prend en charge le profilage des performances et de la mémoire.
- Cibles des packages iOS :
  - ipa-ad-hoc : package iOS destiné à une distribution ad hoc.
  - ipa-app-store : package iOS destiné à une distribution via l'App Store d'Apple.
  - ipa-debug : package iOS contenant des informations de débogage complémentaires. (Les fichiers SWF de l'application doivent également être compilés avec une prise en charge du débogage.)
  - ipa-test : package iOS compilé sans information de débogage ou d'optimisation.
  - ipa-debug-interpreter : fonctionnalité qui équivaut à un package de débogage, mais qui permet de compiler plus rapidement. Néanmoins, le pseudo-code ActionScript est interprété et non traduit en code machine. L'exécution du code est donc ralentie dans un package interpreter.
  - ipa-debug-interpreter-simulator : fonctionnalité équivalente à ipa-debug-interpreter, mais mise en package pour le simulateur iOS. Macintosh uniquement. Si vous utilisez cette option, vous devez inclure également l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.
  - ipa-test-interpreter : fonctionnalité qui équivaut à un package de test, mais qui permet de compiler plus rapidement. Néanmoins, le pseudo-code ActionScript est interprété et non traduit en code machine. L'exécution du code est donc ralentie dans un package interpreter.
  - ipa-test-interpreter-simulator : fonctionnalité équivalente à ipa-test-interpreter, mais mise en package pour le simulateur iOS. Macintosh uniquement. Si vous utilisez cette option, vous devez inclure également l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.
- native : programme d'installation d'application de bureau natif. Le type de fichier produit correspond au format d'installation natif du système d'exploitation sur lequel est exécutée la commande :
  - EXE : Windows
  - DMG : Mac
  - DEB : Ubuntu Linux (AIR 2.6 ou versions antérieures)
  - RPM — Fedora ou OpenSuse Linux (AIR 2.6 ou versions antérieures)

Pour plus d'informations, voir « Mise en package d'un programme d'installation natif de bureau » à la page 58.

-sampler (iOS uniquement, AIR 3.4 et ultérieur) Active l'échantillonneur ActionScript basé sur la télémétrie dans les applications iOS. Cet indicateur permet de profiler l'application avec Adobe Scout. Bien que Scout puisse profiler le contenu de n'importe quelle plate-forme Flash, l'activation de la télémétrie détaillée vous permet de mieux connaître la durée des fonctions ActionScript, les rendus DisplayList et Stage3D, etc. Notez que l'utilisation de cet indicateur peut avoir une incidence sur les performances ; par conséquent, ne l'utilisez pas pour les applications de production.

-hideAneLibSymbols (iOS uniquement, AIR 3.4 et les versions ultérieures) Les développeurs d'applications peuvent utiliser plusieurs extensions natives provenant de sources diverses, et si les fichiers ANE partagent un nom de symbole commun, l'outil ADT génère une erreur de type « symbole dupliqué dans le fichier d'objet ». Dans certains cas, cette erreur peut même se manifester sous forme de blocage au moment de l'exécution. Vous pouvez utiliser l'option `hideAneLibSymbols` pour indiquer si vous souhaitez rendre visibles les symboles de la bibliothèque ANE uniquement aux sources de cette bibliothèque (yes) ou à toutes les sources (no) :

- **yes** : les symboles ANE sont masqués, ce qui résout tous les conflits de symboles indésirables.
- **no** : (valeur par défaut) les symboles ANE ne sont pas masqués. Ce comportement est celui des versions antérieures à AIR 3.4.

-embedBitcode (iOS uniquement, AIR 25 et versions ultérieures) Les développeurs d'application peuvent utiliser l'option `embedBitcode` pour indiquer s'il faut inclure le bitcode dans leur application iOS en spécifiant oui ou non. La valeur par défaut de cette option est non.

DEBUGGER_CONNECTION_OPTIONS Les options de connexion au débogueur indiquent si un package à déboguer doit tenter de se connecter à un débogueur distant qui s'exécute sur un autre ordinateur ou écouter une connexion issue d'un débogueur distant. Cet ensemble d'options est réservé aux packages mobiles à déboguer (cibles apk-debug et ipa-debug). Il est décrit à la section « Options de connexion au débogueur » à la page 192.

-airDownloadURL Spécifie une autre URL de téléchargement et d'installation du moteur d'exécution d'AIR sur un périphérique Android. Si vous ne spécifiez pas d'autre URL, une application AIR redirige l'utilisateur vers le moteur d'exécution d'AIR dans Android Market, le cas échéant.

Si l'application est distribuée via un site autre qu'Android Market géré par Google, il peut s'avérer nécessaire de spécifier l'URL de téléchargement du moteur d'exécution d'AIR à partir de ce site. Certains autres sites n'autorisent pas les applications à demander un téléchargement externe. Cette option est réservée aux packages Android.

NATIVE_SIGNING_OPTIONS Les options de signature natives identifient le certificat requis pour signer un fichier de package natif. Ces options de signature permettent d'appliquer une signature utilisée par le système d'exploitation natif, plutôt que le moteur d'exécution d'AIR. Les options sont sinon identiques à AIR_SIGNING_OPTIONS et font l'objet d'une description détaillée à la section « Options de signature du code de l'outil ADT » à la page 189.

Les signatures natives sont prises en charge sous Windows et Android. Sous Windows, vous devez spécifier à la fois les options de signature AIR et les options de signature natives. Sous Android, vous ne pouvez spécifier que les options de signature natives.

Dans la plupart des cas, un même certificat de signature du code peut appliquer à la fois une signature AIR et une signature native. Certains cas de figure font toutefois exception à la règle. Google stipule, par exemple, que toutes les applications destinées à Android Market doivent impérativement être signées à l'aide d'un certificat valide jusqu'à l'année 2033 au moins. Cela signifie qu'un certificat délivré par une autorité de certification reconnue, recommandée lors de l'application d'une signature AIR, ne doit pas être utilisé pour signer une application Android. (Aucune autorité de certification ne délivre de certificat de signature du code dont la période de validité est aussi longue.)

output Nom du fichier de package à créer. La définition de l'extension du fichier est facultative. Si vous ne l'indiquez pas, une extension adaptée à la valeur -target et au système d'exploitation en cours est ajoutée.

app_descriptor Chemin d'accès au fichier descripteur de l'application. Il est possible de spécifier un chemin relatif (défini par rapport au répertoire actif) ou un chemin absolu. (Le fichier descripteur de l'application est renommé *application.xml* dans le fichier AIR.)

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible :

- Android : le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes de l'outil ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)
- iOS : le kit SDK d'AIR est livré avec un kit SDK iOS captif. L'option -platformsdk permet la mise en package d'applications avec un kit SDK externe ; vous n'êtes donc pas obligé d'utiliser le SDK iOS captif. Par exemple, si vous avez créé une extension avec le dernier kit SDK iOS, vous pouvez spécifier ce kit SDK lors de la mise en package de votre application. En outre, lorsque vous utilisez l'outil ADT avec le simulateur iOS, vous devez toujours inclure l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.

Les développeurs d'applications **-arch** peuvent utiliser cet argument pour créer un package APK pour les plates-formes x86, qui prend les valeurs suivantes :

- armv7 - ADT met en package APK pour la plate-forme Android armv7.
- x86 - ADT met en package APK pour la plate-forme Android x86.

armv7 est la valeur par défaut lorsqu'aucune valeur n'est spécifiée

FILE_OPTIONS Identifie les fichiers d'application à inclure dans le package. Les options de fichier font l'objet d'une description détaillée à la section « Options associées aux fichiers et chemins » à la page 191. Ne spécifiez pas d'options de fichier si vous créez un package natif à partir d'un fichier AIR ou AIRI.

input_airi A spécifier lors de la création d'un package natif à partir d'un fichier AIRI. Les options AIR_SIGNING_OPTIONS sont obligatoires si la cible correspond à *air* (ou qu'aucune cible n'est spécifiée).

input_air A spécifier lors de la création d'un package natif à partir d'un fichier AIR. Ne spécifiez pas d'option AIR_SIGNING_OPTIONS.

ANE_OPTIONS Identifie les options et les fichiers permettant de créer un package d'extensions natives. Les options du package d'extensions sont décrites de façon exhaustive dans « Options d'extension native » à la page 193.

Exemples de commandes ADT -package

Création d'un package de fichiers d'application spécifiques figurant dans le répertoire actif pour une application AIR de type SWF :

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Création d'un package de fichiers d'application spécifiques figurant dans le répertoire actif pour une application AIR de type HTML :

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js
image.gif
```

Création d'un package de tous les fichiers et sous-répertoires inclus dans le répertoire de travail actif :

```
adt –package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Remarque : *le fichier keystore contient la clé privée utilisée pour signer l'application. Veillez à ne jamais inclure le certificat de signature dans le package AIR. Si vous utilisez des caractères génériques dans la commande ADT, déplacez le fichier keystore afin qu'il ne soit pas inclus dans le package. Dans cet exemple, le fichier keystore (cert.p12) réside dans le répertoire parent.*

Création d'un package contenant uniquement les fichiers principaux et un sous-répertoire d'images :

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Création d'un package contenant une application HTML et tous les fichiers contenus dans les sous-répertoires HTML, de scripts et d'images sous-jacents :

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js
html scripts images
```

Création d'un package du fichier application.xml et du fichier SWF principal se trouvant dans un répertoire de travail (release/bin) :

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml –C
release/bin myApp.swf
```

Création d'un package des ressources provenant de plusieurs emplacements du système de fichiers de développement. Dans cet exemple, les actifs de l'application résident dans les dossiers suivants avant la création du package :

```
/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
```

Exécution de la commande ADT suivante à partir du répertoire /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
```

Résultats dans la structure de package suivante :

```
/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
```

Exécution de l'outil ADT en tant que programme Java pour une application SWF simple (sans définir le chemin de classe) :

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Exécution de l'outil ADT en tant que programme Java pour une application HTML simple (sans définir le chemin de classe) :

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Exécution de l'outil ADT en tant que programme Java (le chemin de classe Java étant défini de sorte à inclure le package ADT.jar) :

```
java -com.adobe.air.ADT –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

Exécution de l'outil ADT en tant que tâche Java dans Apache Ant (bien qu'il soit plutôt conseillé d'utiliser la commande ADT directement dans les scripts Ant). Les chemins indiqués dans l'exemple correspondent à un système 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>
```

***Remarque :** sur certains ordinateurs, les caractères à double octet figurant dans les chemins d'accès de système de fichiers risquent d'être incorrectement interprétés. Si tel est le cas, tentez de définir le JRE utilisé pour exécuter l'outil ADT de sorte à utiliser le jeu de caractères UTF-8. Tel est le cas par défaut dans le script de lancement de l'outil ADT sous Mac et Linux. Dans le fichier Windows adt.bat ou si vous exécutez l'outil ADT directement à partir de Java, spécifiez l'option* `-Dfile.encoding=UTF-8` *sur la ligne de commande Java.*

Commande ADT prepare

La commande -prepare crée un package AIRI non signé. Il est impossible d'utiliser seul un package AIRI. Utilisez la commande -sign pour convertir un fichier AIRI en package AIR signé ou la commande package pour convertir le fichier AIRI en package natif.

La commande -prepare gère la syntaxe suivante :

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output Nom du fichier AIRI créé.

app_descriptor Chemin d'accès au fichier descripteur de l'application. Il est possible de spécifier un chemin relatif (défini par rapport au répertoire actif) ou un chemin absolu. (Le fichier descripteur de l'application est renommé *application.xml* dans le fichier AIR.)

FILE_OPTIONS Identifie les fichiers d'application à inclure dans le package. Les options de fichier font l'objet d'une description détaillée à la section « Options associées aux fichiers et chemins » à la page 191.

Commande ADT sign

La commande -sign permet de signer les fichiers AIRI et ANE.

Elle gère la syntaxe suivante :

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Les options de signature AIR identifient le certificat requis pour signer un fichier de package. Les options de signature font l'objet d'une description détaillée à la section « Options de signature du code de l'outil ADT » à la page 189.

input Nom du fichier AIRI ou ANE à signer.

output Nom du package signé à créer.

Si un fichier ANE est déjà signé, l'ancienne signature est éliminée. (Il est impossible de signer à nouveau un fichier AIR. Pour utiliser une nouvelle signature en cas de mise à jour d'une application, utilisez la commande `-migrate`.)

Commande ADT migrate

La commande -migrate applique une signature de migration à un fichier AIR. Vous devez utiliser une signature de migration lorsque vous renouvelez ou modifiez le certificat numérique et devez mettre à jour une application signée à l'aide de l'ancien certificat.

Pour plus d'informations sur la mise en package d'applications AIR avec signature de migration, voir « Signature d'une version mise à jour d'une application AIR » à la page 211.

Remarque : *le certificat de migration doit être appliqué dans les 365 jours qui suivent l'expiration du certificat. Au terme de ce délai, il devient impossible de signer les mises à jour d'une application par le biais d'une signature de migration. Les utilisateurs peuvent commencer par mettre à jour l'application en installant une version associée à une signature de migration, puis installer la mise à jour la plus récente ou désinstaller l'application d'origine et installer le nouveau package AIR.*

Pour utiliser une signature de migration, commencez par signer l'application AIR à l'aide du nouveau certificat ou du certificat renouvelé (à l'aide des commandes -package ou -sign), puis appliquez la signature de migration par le biais de l'ancien certificat et de la commande -migrate.

La commande `-migrate` utilise la syntaxe suivante :

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Options de signature AIR qui identifient le certificat d'origine utilisé pour signer les versions existantes de l'application AIR. Les options de signature font l'objet d'une description détaillée à la section « Options de signature du code de l'outil ADT » à la page 189.

input Fichier AIR déjà signé par le biais du NOUVEAU certificat d'application.

output Nom du package final qui contient les signatures issues du nouveau et de l'ancien certificats.

les fichiers d'entrée et de sortie AIR doivent porter un nom différent.

Remarque : *La commande ADT migrate ne peut pas être utilisée avec les applications de bureau AIR incluant des extensions natives, car ces applications ont été mises en package comme des programmes d'installation natifs, pas comme des fichiers .air. Pour modifier les certificats pour une application AIR qui inclut une extension native, mettez l'application en package en utilisant la « Commande ADT package » à la page 175 avec l'indicateur -migrate.*

Commande ADT checkstore

La commande -checkstore permet de vérifier la validité d'un keystore. La commande gère la syntaxe suivante :

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS Options de signature qui identifient le keystore à valider. Les options de signature font l'objet d'une description détaillée à la section « Options de signature du code de l'outil ADT » à la page 189.

Commande ADT certificate

La commande -certificate permet de créer un certificat de signature du code numérique auto-signé. La commande gère la syntaxe suivante :

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type
output password
```

-cn Chaîne assignée en tant que nom commun du nouveau certificat.

-ou Chaîne assignée en tant qu'unité organisationnelle chargée de délivrer le certificat. (Facultatif)

`-o` Chaîne assignée en tant qu'organisation qui délivre le certificat. (Facultatif)

`-c` Code pays ISO-3166 de deux lettres. Un certificat n'est pas généré si un code incorrect est indiqué. (Facultatif)

`-validityPeriod` Nombre d'années de validité du certificat. Si aucune valeur n'est spécifiée, une validité de cinq ans est définie. (Facultatif)

`key_type` Type de clé à utiliser pour que le certificat soit *2048-RSA*.

output Chemin et nom du fichier de certificat à générer.

password Mot de passe d'accès au nouveau certificat. Le mot de passe est obligatoire si un fichier AIR est signé par le biais de ce certificat.

Commande ADT installApp

La commande -installApp installe une application sur un périphérique ou un émulateur.

Vous devez désinstaller une application existante avant de la réinstaller à l'aide de cette commande.

La commande gère la syntaxe suivante :

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package
fileName
```

-platform Nom de la plate-forme du périphérique. Spécifiez *ios* ou *android*.

-platformsdk Chemin vers le kit SDK de la plate-forme pour le périphérique cible (facultatif) :

- Android : le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes de l'outil ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)
- iOS : le kit SDK d'AIR est livré avec un kit SDK iOS captif. L'option -platformsdk permet la mise en package d'applications avec un kit SDK externe ; vous n'êtes donc pas obligé d'utiliser le SDK iOS captif. Par exemple, si vous avez créé une extension avec le dernier kit SDK iOS, vous pouvez spécifier ce kit SDK lors de la mise en package de votre application. En outre, lorsque vous utilisez l'outil ADT avec le simulateur iOS, vous devez toujours inclure l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.

-device Spécifie le *ios_simulator*, le numéro de série (Android) ou le handle (iOS) du périphérique raccordé. Sous iOS, ce paramètre est obligatoire ; sous Android, ce paramètre doit être spécifié uniquement lorsque plusieurs périphériques ou émulateurs Android sont raccordés à votre ordinateur et en cours d'exécution. Si le périphérique spécifié n'est pas connecté, ADT renvoie le code de sortie 14 : Erreur de périphérique (Android) ou Périphérique spécifié non valide (iOS). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Remarque : *l'installation d'un fichier IPA directement sur un périphérique iOS est possible sur AIR 3.4 et les versions ultérieures, et nécessite iTunes 10.5.0 ou une version ultérieure.*

Utilisez la commande `adt -devices` (disponible dans AIR 3.4 et les versions ultérieures) pour déterminer le handle ou le numéro de série des périphériques raccordés. Notez que sous iOS, vous utilisez le handle et non le UUID du périphérique. Pour plus d'informations, voir « Commande des périphériques ADT » à la page 188.

Sous Android, vous pouvez aussi utiliser l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

-package Nom de fichier du package à installer. Sous iOS, ce fichier doit être un fichier IPA. Sur Android, il doit s'agir d'un package APK. Si le package spécifié est déjà installé, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique).

Commande ADT appVersion

La commande -appVersion indique la version d'une application installée sur un périphérique ou un émulateur. La commande gère la syntaxe suivante :

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Nom de la plate-forme du périphérique. Spécifiez *ios* ou *android*.

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible :

- Android : le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes de l'outil ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)
- iOS : le kit SDK d'AIR est livré avec un kit SDK iOS captif. L'option -platformsdk permet la mise en package d'applications avec un kit SDK externe ; vous n'êtes donc pas obligé d'utiliser le SDK iOS captif. Par exemple, si vous avez créé une extension avec le dernier kit SDK iOS, vous pouvez spécifier ce kit SDK lors de la mise en package de votre application. En outre, lorsque vous utilisez l'outil ADT avec le simulateur iOS, vous devez toujours inclure l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.

-device Spécifiez *ios_simulator* ou le numéro de série du périphérique. Ne spécifiez le périphérique que si plusieurs périphériques ou émulateurs Android sont connectés à l'ordinateur et sont en cours d'exécution. Si le périphérique spécifié n'est pas connecté, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Sous Android, utilisez l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

-appid ID de l'application AIR installée. Si aucune application dotée de l'ID spécifié n'est installée sur le périphérique, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique).

Commande ADT launchApp

La commande -launchApp exécute une application installée sur un périphérique ou un émulateur. La commande gère la syntaxe suivante :

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Nom de la plate-forme du périphérique. Spécifiez *ios* ou *android*.

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible :

- Android : le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes de l'outil ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)
- iOS : le kit SDK d'AIR est livré avec un kit SDK iOS captif. L'option -platformsdk permet la mise en package d'applications avec un kit SDK externe ; vous n'êtes donc pas obligé d'utiliser le SDK iOS captif. Par exemple, si vous avez créé une extension avec le dernier kit SDK iOS, vous pouvez spécifier ce kit SDK lors de la mise en package de votre application. En outre, lorsque vous utilisez l'outil ADT avec le simulateur iOS, vous devez toujours inclure l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.

-device Spécifiez *ios_simulator* ou le numéro de série du périphérique. Ne spécifiez le périphérique que si plusieurs périphériques ou émulateurs Android sont connectés à l'ordinateur et sont en cours d'exécution. Si le périphérique spécifié n'est pas connecté, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Sous Android, utilisez l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

-appid ID de l'application AIR installée. Si aucune application dotée de l'ID spécifié n'est installée sur le périphérique, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique).

Commande ADT uninstallApp

La commande -uninstallApp supprime entièrement une application installée sur un émulateur ou périphérique distant. La commande gère la syntaxe suivante :

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Nom de la plate-forme du périphérique. Spécifiez *ios* ou *android*.

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible :

- Android : le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes de l'outil ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)
- iOS : le kit SDK d'AIR est livré avec un kit SDK iOS captif. L'option -platformsdk permet la mise en package d'applications avec un kit SDK externe ; vous n'êtes donc pas obligé d'utiliser le SDK iOS captif. Par exemple, si vous avez créé une extension avec le dernier kit SDK iOS, vous pouvez spécifier ce kit SDK lors de la mise en package de votre application. En outre, lorsque vous utilisez l'outil ADT avec le simulateur iOS, vous devez toujours inclure l'option -platformsdk en spécifiant le chemin vers le kit SDK du simulateur iOS.

-device Spécifiez *ios_simulator* ou le numéro de série du périphérique. Ne spécifiez le périphérique que si plusieurs périphériques ou émulateurs Android sont connectés à l'ordinateur et sont en cours d'exécution. Si le périphérique spécifié n'est pas connecté, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Sous Android, utilisez l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

-appid ID de l'application AIR installée. Si aucune application dotée de l'ID spécifié n'est installée sur le périphérique, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique).

Commande ADT installRuntime

La commande -installRuntime installe le moteur d'exécution d'AIR sur un périphérique.

Vous devez désinstaller toute version existante du moteur d'exécution d'AIR avant de le réinstaller par le biais de cette commande.

La commande gère la syntaxe suivante :

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package
fileName
```

-platform Nom de la plate-forme du périphérique. Cette commande est actuellement gérée par la plate-forme Android uniquement. Utilisez le nom, *android*.

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible. A l'heure actuelle, l'unique kit SDK de plate-forme pris en charge est Android. Le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)

-device Numéro de série du périphérique. Ne spécifiez le périphérique que si plusieurs périphériques ou émulateurs sont connectés à l'ordinateur et sont en cours d'exécution. Si le périphérique spécifié n'est pas connecté, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Sous Android, utilisez l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

-package Nom de fichier du moteur d'exécution à installer. Sur Android, il doit s'agir d'un package APK. Si aucun package n'est spécifié, le moteur d'exécution adapté au périphérique ou à l'émulateur est sélectionné parmi les options proposées par le kit SDK d'AIR. Si le moteur d'exécution est déjà installé, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique).

Commande ADT runtimeVersion

La commande -runtimeVersion indique la version installée du moteur d'exécution d'AIR sur un périphérique ou un émulateur. La commande gère la syntaxe suivante :

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Nom de la plate-forme du périphérique. Cette commande est actuellement gérée par la plate-forme Android uniquement. Utilisez le nom, *android*.

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible. A l'heure actuelle, l'unique kit SDK de plate-forme pris en charge est Android. Le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)

-device Numéro de série du périphérique. Ne spécifiez le périphérique que si plusieurs périphériques ou émulateurs sont connectés à l'ordinateur et sont en cours d'exécution. Si le moteur d'exécution n'est pas installé ou que le périphérique spécifié n'est pas connecté, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Sous Android, utilisez l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

Commande ADT uninstallRuntime

La commande -uninstallRuntime supprime entièrement le moteur d'exécution d'AIR d'un périphérique ou d'un émulateur. La commande gère la syntaxe suivante :

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Nom de la plate-forme du périphérique. Cette commande est actuellement gérée par la plate-forme Android uniquement. Utilisez le nom, *android*.

-platformsdk Chemin d'accès au kit SDK de la plate-forme du périphérique cible. A l'heure actuelle, l'unique kit SDK de plate-forme pris en charge est Android. Le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Il est également inutile d'indiquer le chemin du kit SDK de plate-forme sur la ligne de commande si la variable d'environnement AIR_ANDROID_SDK_HOME est déjà définie. (Si les deux variables sont définies, le chemin indiqué sur la ligne de commande est utilisé.)

-device Numéro de série du périphérique. Ne spécifiez le périphérique que si plusieurs périphériques ou émulateurs sont connectés à l'ordinateur et sont en cours d'exécution. Si le périphérique spécifié n'est pas connecté, l'outil ADT renvoie le code de sortie 14 (erreur de périphérique). Si plusieurs périphériques ou émulateurs sont connectés et qu'aucun périphérique n'est spécifié, l'outil ADT renvoie le code de sortie 2 (erreur d'utilisation).

Sous Android, utilisez l'outil ADB intégré pour dresser la liste des numéros de série des périphériques et émulateurs en cours d'exécution connectés :

```
adb devices
```

Commande des périphériques ADT

La commande -help de l'outil ADT affiche les ID des périphériques et émulateurs actuellement raccordés :

```
adt -devices -platform iOS|android
```

-platform Nom de la plate-forme à vérifier. Spécifiez `android` ou `iOS`.

Remarque : *sous iOS, cette commande requiert l'installation d'iTunes 10.5.0 ou d'une version ultérieure.*

Commande ADT help

La commande ADT -help affiche un rappel succinct des options de ligne de commande :

```
adt -help
```

La sortie d'aide utilise les conventions relatives aux symboles suivantes :

- <> : les éléments entre crochets indiquent les informations à fournir.
- () : les éléments entre parenthèses indiquent les options assimilées à un groupe dans la sortie de la commande d'aide.
- ALL_CAPS : les éléments en majuscules indiquent un ensemble d'options décrit séparément.
- | :OU. Par exemple, (A | B), signifie élément A ou élément B.

- ? : 0 ou 1. Un point d'interrogation suivi d'un élément indique que ce dernier est facultatif et qu'une seule occurrence est autorisée, le cas échéant.
- * : 0 ou plus. Un astérisque suivi d'un élément indique que ce dernier est facultatif et qu'un nombre illimité d'occurrences est autorisé.
- + : 1 ou plus. Un signe plus suivi d'un élément indique que ce dernier est obligatoire et que plusieurs occurrences sont autorisées.
- aucun symbole : si un élément ne possède pas de symbole en suffixe, il est obligatoire et une seule occurrence est autorisée.

Ensembles d'options ADT

Plusieurs commandes ADT partagent des ensembles d'options.

Options de signature du code de l'outil ADT

ADT utilise l'architecture JCA (Java Cryptography Architecture) pour accéder aux clés privées et aux certificats pour les applications AIR de signature. Les options de signature permettent d'identifier le keystore ainsi que la clé privée et le certificat qu'il contient.

Le keystore doit comprendre à la fois la clé privée et la chaîne de certificat associée. Si le certificat de signature est lié à un certificat approuvé qui réside sur l'ordinateur, c'est le contenu du champ de nom commun du certificat qui s'affiche en tant que nom d'éditeur dans la boîte de dialogue d'installation d'AIR.

L'outil ADT exige que le certificat soit conforme à la norme x509v3 (RFC3280) et qu'il comprenne l'extension d'utilisation avancée de la clé dotée des valeurs appropriées à la signature du code. Les contraintes inhérentes au certificat sont respectées et pourraient empêcher l'utilisation de certains certificats pour signer une application AIR.

Remarque : *le cas échéant, l'outil ADT fait appel aux paramètres proxy de l'environnement d'exécution Java (JRE, Java Runtime Environment) pour se connecter à des ressources Internet afin de vérifier les listes de révocation de certificats et d'obtenir des horodatages. Si vous rencontrez des problèmes lors de la connexion à ces ressources Internet avec l'outil ADT alors que le réseau requiert des paramètres proxy spécifiques, vous devrez peut-être configurer les paramètres proxy de JRE.*

Syntaxe des options de signature AIR

Les options de signature font appel à la syntaxe suivante (sur une seule ligne de commande) :

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias Alias d'une clé dans le keystore. Il est inutile de spécifier un alias lorsqu'un keystore contient uniquement un certificat. Si aucun alias n'est précisé, l'outil ADT utilise la première clé du keystore.

Toutes les applications de gestion de keystores ne permettent pas d'affecter un alias à des certificats. Avec le keystore du système Windows par exemple, vous devez utiliser le nom unique du certificat comme alias. L'utilitaire Java Keytool vous permet de dresser la liste des certificats disponibles afin de déterminer les alias pertinents. Par exemple, si vous exécutez la commande :

```
keytool -list -storetype Windows-MY
```

la sortie suivante sera générée pour un certificat :

```
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
```

Pour faire référence à ce certificat sur la ligne de commande d'ADT, définissez l'alias sur :

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

Sous Mac OS X, l'alias d'un certificat dans la chaîne de clé correspond au nom affiché dans l'application Trousseau d'accès.

-storetype Type de keystore déterminé par l'implémentation du keystore. L'implémentation par défaut du keystore comprise dans la plupart des installations Java prend en charge les types `JKS` et `PKCS12`. Java 5.0 assure la prise en charge du type `PKCS11` (permettant d'accéder aux keystores situés sur des jetons matériels) et du type `Keychain` (permettant d'accéder à la chaîne de clé Mac OS X). Java 6.0 prend en charge le type `MSCAPI` (sous Windows). Si d'autres fournisseurs JCA ont été installés et configurés, il se peut que d'autres types de keystores soient disponibles. Si aucun type de keystore n'est spécifié, c'est le type par défaut correspondant au fournisseur JCA défini par défaut qui sera utilisé.

Type de magasin	Format du keystore	Version Java minimale
JKS	Fichier keystore Java (.keystore)	1.2
PKCS12	Fichier PKCS12 (.p12 ou .pfx)	1.4
PKCS11	Jeton matériel	1.5
KeychainStore	Chaîne de clé Mac OS X	1.5
Windows-MY ou Windows-ROOT	MSCAPI	1.6

-keystore Chemin d'accès au fichier keystore pour les types de magasins basés sur un fichier.

-storepass Mot de passe requis pour accéder au keystore. Si vous n'avez pas spécifié de mot de passe, l'outil ADT vous invite à en entrer un.

-keypass Mot de passe requis pour accéder à la clé privée permettant de signer l'application AIR. Si vous n'avez pas spécifié de mot de passe, l'outil ADT vous invite à en entrer un.

Remarque : *si vous entrez un mot de passe dans le cadre de la commande ADT, les caractères sont enregistrés dans l'historique de ligne de commande. Il est par conséquent recommandé d'éviter l'utilisation des options -keypass ou -storepass si la sécurité du certificat est importante. Notez également que si vous omettez les options de mot de passe, les caractères saisis dans les invites de saisie de mot de passe ne s'affichent pas (pour les mêmes raisons de sécurité). Saisissez simplement le mot de passe et appuyez sur la touche Entrée.*

-providerName Fournisseur JCA conçu pour le type de keystore spécifié. Si vous ne spécifiez pas de fournisseur, l'outil ADT utilise le fournisseur associé par défaut au type de keystore.

-tsa Indique l'URL d'un serveur d'horodatage compatible RFC3161 à des fins d'horodatage de la signature numérique. Si vous n'avez pas spécifié d'URL, un serveur d'horodatage par défaut fourni par Geotrust est utilisé. Dès lors que la signature d'une application AIR est horodatée, il est encore possible d'installer l'application après l'expiration du certificat de signature, car l'horodatage vérifie la validité du certificat au moment de l'apposition de la signature.

Si l'outil ADT ne parvient pas à se connecter au serveur d'horodatage, la signature est annulée et le package n'est pas créé. Pour désactiver l'horodatage, spécifiez `-tsa none`. Toutefois, il devient impossible d'installer une application AIR mise en package sans horodatage une fois le certificat de signature arrivé à expiration.

Remarque : *la plupart des options de signature correspondent à l'option équivalente de l'utilitaire Java Keytool. Vous pouvez utiliser ce dernier afin d'examiner et de gérer des keystores sous Windows. L'utilitaire de sécurité Apple® s'acquitte également de ces tâches sous Mac OS X.*

-provisioning-profile Fichier de configuration Apple iOS (requis par la mise en package d'applications iOS uniquement).

Exemples d'options de signature
Signature à l'aide d'un fichier .p12 :

```
-storetype pkcs12 -keystore cert.p12
```

Signature à l'aide du keystore Java par défaut :

```
-alias AIRcert -storetype jks
```

Signature à l'aide d'un keystore Java spécifique :

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Signature à l'aide de la chaîne de clé Mac OS X :

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Signature à l'aide du keystore du système Windows :

```
-alias cn=AIRCert -storeype Windows-MY
```

Signature à l'aide d'un jeton matériel (voir les instructions du fabricant du jeton concernant la configuration de Java en vue d'utiliser le jeton et de définir la valeur `providerName` pertinente) :

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Signature sans horodatage incorporé :

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Options associées aux fichiers et chemins

Les options associées aux fichiers et chemins spécifient tous les fichiers inclus dans le package. Elles gèrent la syntaxe suivante :

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs Fichiers et répertoires à inclure dans le package du fichier AIR. Le nombre de fichiers et de répertoires spécifié n'est pas limité, mais les noms doivent être séparés par un espace. Si vous indiquez un répertoire, tous les fichiers et sous-répertoires qu'il contient, à l'exception des fichiers masqués, sont ajoutés au package. (En outre, si le fichier descripteur d'application est spécifié, soit directement soit via le développement de caractères génériques ou de répertoires, il est ignoré et exclu du package la seconde fois.) Les fichiers et répertoires spécifiés doivent se trouver dans le répertoire actif ou l'un de ses sous-répertoires. Utilisez l'option `-C` pour changer de répertoire actif.

Important : *les caractères génériques ne sont pas admis dans les arguments `file_or_dir` suivant l'option `-C`. (Les shells de commande développent les caractères génériques avant de transmettre les arguments à l'outil ADT, ce qui incite ce dernier à rechercher des fichiers à un emplacement erroné.) Vous pouvez néanmoins continuer à utiliser le caractère point (.) pour indiquer le répertoire actif. l'élément `-C assets .` copie, par exemple, le contenu intégral du répertoire des actifs, sous-répertoires compris, dans le niveau racine du package de l'application.*

`-C dir files_and_dirs` Modifie le répertoire de travail de la valeur de *dir* avant de traiter les fichiers et les répertoires successifs ajoutés au package de l'application (spécifié dans **`files_and_dirs`**). Les fichiers ou les répertoires sont ajoutés à la racine du package de l'application. Il est possible d'utiliser l'option `–C` de manière illimitée afin d'inclure des fichiers provenant de divers emplacements du système de fichiers. Si un chemin relatif est défini pour `dir`, il est toujours résolu à partir du répertoire de travail initial.

Comme l'outil ADT traite les fichiers et répertoires inclus dans le package, les chemins relatifs définis entre le répertoire actif et les fichiers cible sont stockés. Ces chemins sont développés dans la structure de répertoires de l'application au moment de l'installation du package. Par conséquent, en indiquant `-C release/bin lib/feature.swf`, vous placez le fichier `release/bin/lib/feature.swf` dans le sous-répertoire `lib` du dossier racine de l'application.

`-e file_or_dir dir` Place le fichier ou le répertoire dans le répertoire spécifié du package. Il est impossible d'utiliser cette option lors de la mise en package d'un fichier ANE.

Remarque : *l'élément `<content>` du fichier descripteur de l'application doit indiquer l'emplacement final du fichier principal de l'application au sein de l'arborescence de répertoires du package de l'application.*

-extdir *dir* La valeur du paramètre *`dir`* correspond au nom du répertoire dans lequel rechercher les extensions natives (fichiers ANE). Spécifiez soit un chemin absolu soit un chemin relatif au répertoire actuel. Vous pouvez spécifier plusieurs fois l'option `-extdir`.

Le répertoire spécifié contient les fichiers ANE des extensions natives que l'application utilise. Chaque fichier ANE de ce répertoire possède l'extension de nom de fichier .ane. Il n'est toutefois *pas* impératif que le nom de fichier qui précède l'extension .ane soit identique à la valeur de l'élément `extensionID` du fichier descripteur de l'application.

Par exemple, si vous utilisez `-extdir ./extensions`, le répertoire `extensions` peut avoir l'aspect suivant :

```
extensions/
    extension1.ane
    extension2.ane
```

Remarque : *l'outil ADT et l'outil ADL traitent différemment l'utilisation de l'option -extdir. Dans ADL, cette option spécifie un répertoire contenant des sous-répertoires, chacun contenant un fichier ANE extrait du package. Dans ADT, cette option spécifie un répertoire contenant les fichiers ANE.*

Options de connexion au débogueur

Lorsque la cible du package est apk-debug, ipa-debug ou ipa-debug-interpreter, il est possible d'utiliser les options de connexion pour indiquer si l'application mobile va tenter de se connecter à un débogueur à distance (normalement utilisé pour le débogage wi-fi) ou écouter une connexion entrante à partir d'un débogueur à distance (normalement utilisé pour le débogage USB). Utilisez l'option `-connect` pour établir une connexion à un débogueur, l'option `-listen` pour accepter une connexion émanant d'un débogueur via une connexion USB. Ces options s'excluent mutuellement, c'est-à-dire qu'il est impossible de les utiliser ensemble.

L'option -connect gère la syntaxe suivante :

```
-connect hostString
```

-connect Si cet indicateur est spécifié, l'application tente de se connecter à un débogueur distant.

hostString Chaîne d'identification de l'ordinateur qui exécute l'outil de débogage Flash, FDB. Si cet indicateur n'est pas spécifié, l'application tente de se connecter à un débogueur qui s'exécute sur l'ordinateur de création du package. La chaîne hostString peut être un nom de domaine complet, *NomMachine.SousGroupe.exemple.com*, ou une adresse IP, *192.168.4.122*. Si la machine spécifiée (ou définie par défaut) est introuvable, le moteur d'exécution affiche une boîte de dialogue permettant de saisir un nom d'hôte valide.

L'option `-listen` gère la syntaxe suivante :

```
-listen port
```

-listen Si cet élément est spécifié, le moteur d'exécution attend une connexion émanant d'un débogueur distant.

port (Facultatif) port d'écoute. Par défaut, le moteur d'exécution écoute sur le port 7936. Pour plus d'informations sur l'utilisation de l'option `-listen`, voir « Débogage à distance avec le programme FDB via USB » à la page 112.

Options de profilage d'application Android

Si le package a pour cible apk-profile, les options de profilage permettent de spécifier le fichier SWF préchargé requis à des fins de profilage de performance et de mémoire. Les options de profilage gèrent la syntaxe suivante :

```
-preloadSWFPath directory
```

-preloadSWFPath Si cet indicateur est spécifié, l'application tente de trouver le fichier SWF préchargé dans le répertoire indiqué. S'il n'est pas spécifié, l'outil ADT inclut le fichier SWF préchargé issu du kit SDK d'AIR.

directory Répertoire contenant le fichier SWF préchargé de profilage.

Options d'extension native

Les options d'extension native spécifient les options et fichiers requis pour mettre en package un fichier ANE associé à une extension native. Utilisez ces options avec une commande de package ADT dans laquelle l'option `-target` est définie sur `ane`.

```
extension-descriptor -swc swcPath
   -platform platformName
  -platformoptions path/platform.xml
   FILE_OPTIONS
```

extension-descriptor Fichier descripteur de l'extension native.

-swc Fichier SWC contenant les ressources et le code ActionScript associés à l'extension native.

-platform Nom de la plate-forme prise en charge par le fichier ANE. Vous pouvez inclure plusieurs options `-platform`, chacune avec son paramètre `FILE_OPTIONS`.

-platformoptions Chemin vers le fichier d'options d'une plate-forme (platform.xml). Utilisez ce fichier pour spécifier les options personnalisées de l'éditeur de liens, les bibliothèques partagées et les bibliothèques statiques tierces utilisées par l'extension. Pour plus d'informations et d'exemples, voir Bibliothèques iOS natives.

FILE_OPTIONS Identifie les fichiers de la plate-forme native à inclure au package, notamment les bibliothèques statiques à inclure au package de l'extension native. Les options de fichier font l'objet d'une description détaillée à la section « Options associées aux fichiers et chemins » à la page 191. (Notez qu'il est impossible d'utiliser l'option -e lors de la mise en package d'un fichier ANE.)

Voir aussi

Mise en package d'une extension native

Messages d'erreur du programme ADT

Les tableaux ci-après recensent les erreurs pouvant être signalées par le programme ADT et leurs causes probables.

Erreurs de validation du fichier descripteur d'application

Code d'erreur	Description	Remarques
100	Le fichier descripteur d'application ne peut pas être analysé.	Recherchez les erreurs de syntaxe XML éventuelles dans le fichier descripteur d'application, par exemple des balises non fermées.
101	Espace de nom manquant	Ajoutez l'espace de nom nécessaire.
102	Espace de nom non valide	Vérifiez l'orthographe de l'espace de nom.
103	Élément ou attribut inattendu	Supprimez les éléments ou les attributs fautifs. Les valeurs personnalisées ne sont pas autorisées dans le fichier descripteur. Vérifiez l'orthographe du nom des éléments ou des attributs. Assurez-vous que ces éléments soient placés dans l'élément parent approprié et que les attributs soient utilisés avec les éléments corrects.
104	Élément ou attribut manquant	Ajoutez l'élément ou l'attribut requis.
105	L'élément ou l'attribut contient une valeur non valide.	Corrigez la valeur fautive.
106	Combinaison d'attributs de fenêtre non valide	Certains paramètres de fenêtre, tels que `transparency = true` et `systemChrome = standard` ne peuvent pas être utilisés simultanément. Modifiez l'un des paramètres incompatibles.
107	La taille minimale de la fenêtre est supérieure à sa taille maximale.	Modifiez l'un des paramètres de taille.
108	Attribut déjà utilisé dans un élément précédent	
109	Doublon	Supprimez le doublon.
110	Un élément au moins du type spécifié est obligatoire.	Ajoutez l'élément requis.
111	Aucun des profils stipulés dans le fichier descripteur de l'application ne prend en charge les extensions natives.	Ajoutez un profil à la liste supportedProfies qui prend en charge les extensions natives.
112	La cible AIR ne prend pas en charge les extensions natives.	Sélectionnez une cible qui prend en charge les extensions natives.
113	<nativeLibrary> et <initializer> doivent être spécifiés conjointement.	Vous devez spécifier une fonction d'initialisation pour chaque bibliothèque native de l'extension native.
114	<finalizer> détecté sans <nativeLibrary>.	Ne spécifiez pas d'élément finalizer, sauf si la plate-forme utilise une bibliothèque native.

Code d'erreur	Description	Remarques
115	La plate-forme par défaut ne doit pas contenir d'implémentation native.	Ne spécifiez pas de bibliothèque native dans l'élément de plate-forme par défaut.
116	L'invocation du navigateur n'est pas prise en charge pour cette cible.	Il est impossible de définir l'élément `<allowBrowserInvocation>` sur `true` pour la cible de mise en package spécifiée.
117	Cette cible requiert au moins n espaces de noms pour mettre en package les extensions natives.	Modifiez l'espace de noms AIR dans le descripteur de l'application en définissant une valeur prise en charge.

Pour plus d'informations sur les espaces de noms, les éléments, les attributs et leurs valeurs valides, voir « Fichiers descripteurs d'applications AIR » à la page 217.

Erreurs des icônes d'application

Code d'erreur	Description	Remarques
200	Le fichier d'icône ne peut pas être ouvert.	Vérifiez la présence du fichier à l'emplacement spécifié. Servez-vous d'une autre application pour vérifier que le fichier peut être ouvert.
201	La taille de l'icône n'est pas correcte.	La taille de l'icône (en pixels) doit correspondre à la balise XML. Par exemple, pour l'élément de descripteur d'application : `<image32x32>icon.png</image32x32>` La taille de l'image `icon.png` doit être exactement de 32x32 pixels.
202	Le fichier d'icône contient un format d'image non pris en charge.	Seul le format PNG est pris en charge. Convertissez les images en un autre format avant de mettre votre application en package.

Erreurs du fichier d'application

Code d'erreur	Description	Remarques
300	Le fichier est manquant ou ne peut pas être ouvert.	Un fichier spécifié sur la ligne de commande est introuvable ou ne peut pas être ouvert.
301	Le fichier descripteur d'application est manquant ou ne peut pas être ouvert.	Le fichier descripteur d'application est introuvable à l'emplacement spécifié ou ne peut pas être ouvert.
302	Le fichier de contenu racine n'est pas dans le package.	Le fichier SWF ou HTML référencé dans l'élément `<content>` du descripteur d'application doit être ajouté dans le package via son inclusion dans la liste des fichiers sur la ligne de commande d'ADT.

Code d'erreur	Description	Remarques
303	Le fichier d'icône n'est pas dans le package.	Les fichiers d'icône spécifiés dans le descripteur d'application doivent être ajoutés dans le package via leur inclusion dans la liste des fichiers sur la ligne de commande d'ADT. Les fichiers d'icône ne sont pas ajoutés automatiquement.
304	Le contenu initial de la fenêtre n'est pas valide.	Le fichier référencé dans l'élément `<content>` du descripteur d'application n'est pas reconnu comme un fichier HTML ou SWF valide.
305	La version SWF du contenu initial de la fenêtre dépasse la version de l'espace de nom.	La version SWF du fichier référencé dans l'élément `<content>` du descripteur d'application n'est pas prise en charge par la version d'AIR spécifiée dans l'espace de noms du descripteur. Par exemple, tenter de créer un package avec un fichier SWF10 (Flash Player 10) en tant que contenu initial d'une application AIR 1.1 génère cette erreur.
306	Profil non pris en charge	Le profil que vous spécifiez dans le fichier descripteur d'application n'est pas pris en charge. Voir « supportedProfiles » à la page 252.
307	L'espace de noms doit correspondre au moins à *nnn*.	Utilisez l'espace de noms adapté aux fonctionnalités de l'application (tel que l'espace de noms 2.0).

Codes de sortie des autres erreurs

Code de sortie	Description	Remarques
2	Erreur d'utilisation	Vérifiez si les arguments de ligne de commande contiennent des erreurs.
5	Erreur inconnue	Cette erreur désigne une situation que les conditions d'erreur habituelles ne peuvent pas expliquer. Les causes potentielles comprennent une incompatibilité entre ADT et l'environnement d'exécution Java (JRE), des installations ADT ou JRE corrompues et des erreurs de programmation dans ADT.
6	Impossible d'écrire dans le répertoire de sortie	Assurez-vous que le répertoire de sortie spécifié (ou implicite) soit accessible et que le lecteur qui l'héberge dispose de suffisamment d'espace disque.
7	Impossible d'accéder au certificat	Assurez-vous que le chemin d'accès au magasin de clés soit correctement spécifié. Vérifiez que le certificat est accessible dans le magasin de clés. L'utilitaire Java 1.6 Keytool peut être utilisé pour dépanner les problèmes d'accès au certificat.
8	Certificat non valide	Le fichier du certificat n'a pas été créé correctement, a été modifié, est arrivé à expiration ou a été révoqué.

Code de sortie	Description	Remarques
9	Impossible de signer un fichier AIR	Vérifiez les options de signature transmises à ADT.
10	Impossible de créer l'horodatage	ADT n'a pas pu se connecter au serveur d'horodatage. Si vous vous connectez à Internet via un serveur proxy, vous devrez peut-être configurer les paramètres proxy de JRE.
11	Erreur de création de certificat	Vérifiez les arguments de ligne de commande utilisés pour la création des signatures.
12	Entrée non valide	Vérifiez les chemins de fichier et les autres arguments transmis à ADT sur la ligne de commande.
13	Kit SDK de périphérique introuvable	Vérifiez la configuration du kit SDK du périphérique. L'outil ADT ne trouve pas le kit SDK du périphérique requis pour exécuter la commande spécifiée.
14	Erreur de périphérique	L'outil ADT ne peut pas exécuter la commande en raison d'un problème ou d'une restriction liés au périphérique. Ce code de sortie est, par exemple, généré lors d'une tentative de désinstallation d'une application qui n'a pas été installée.
15	Pas de périphérique	Vérifiez qu'un périphérique est connecté et sous tension ou qu'un émulateur est en cours d'exécution.
16	Composants GPL introuvables	Le kit SDK d'AIR actif ne contient pas tous les composants requis pour exécuter l'opération.
17	Échec de l'outil de mise en package du périphérique.	Le package n'a pas pu être créé, car les composants attendus du système d'exploitation sont manquants.

Erreurs Android

Code de sortie	Description	Remarques
400	La version actuelle du kit SDK d'Android ne prend pas en charge l'attribut.	Vérifiez l'orthographe du nom de l'attribut et assurez-vous qu'il correspond à un attribut valide de l'élément dans lequel il apparaît. Il sera peut-être nécessaire de définir l'indicateur -platformsdk dans la commande ADT si l'attribut a été introduit après Android 2.2.
401	La version actuelle du kit SDK d'Android ne prend pas en charge la valeur de l'attribut.	Vérifiez l'orthographe de la valeur de l'attribut et assurez-vous que celle-ci est gérée par l'attribut. Il sera peut-être nécessaire de définir l'indicateur -platformsdk dans la commande ADT si la valeur de l'attribut a été introduite après Android 2.2.
402	La version actuelle du kit SDK d'Android ne prend pas en charge la balise XML.	Vérifiez l'orthographe du nom de la balise XML et assurez-vous qu'il s'agit d'un élément de document manifeste Android valide. Il sera peut-être nécessaire de définir l'indicateur -platformsdk dans la commande ADT si l'élément a été introduit après Android 2.2.
403	Il est interdit de remplacer une balise Android	L'application tente de remplacer un élément de manifeste Android dont l'utilisation est réservée à AIR. Voir « Paramètres Android » à la page 79.
404	Il est interdit de remplacer un attribut Android	L'application tente de remplacer un attribut de manifeste Android dont l'utilisation est réservée à AIR. Voir « Paramètres Android » à la page 79.
405	La balise Android %1 doit être le premier élément dans la balise manifestAdditions.	Déplacez la balise spécifiée à l'emplacement requis.
406	L'attribut %1 de la balise Android %2 possède la valeur non valide %3.	Fournissez une valeur valide pour l'attribut.

Variables d'environnement ADT

Le cas échéant, l'outil ADT lit la valeur des variables d'environnement suivantes :

AIR_ANDROID_SDK_HOME spécifie le chemin d'accès au répertoire racine du kit SDK d'Android (le répertoire qui contient le dossier des outils). Le kit SDK d'AIR 2.6+ inclut les outils du kit SDK d'Android SDK nécessaires à l'implémentation des commandes ADT correspondantes. Ne définissez cette valeur que si vous souhaitez utiliser une autre valeur du kit SDK d'Android. Si cette variable est définie, il est inutile de spécifier l'option -platformsdk lors de l'exécution de commandes ADT qui en ont besoin. Si cette variable et l'option de ligne de commande sont définies, le chemin spécifié sur la ligne de commande est utilisé.

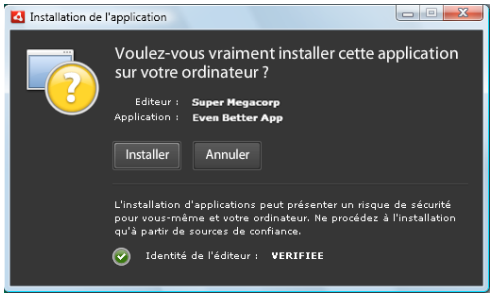
AIR_EXTENSION_PATH spécifie une liste de répertoires dans lesquels rechercher les extensions natives requises par une application. Une recherche logique est effectuée dans cette liste de répertoires après avoir spécifié les répertoires d'extensions natives sur la ligne de commande d'ADT. La commande ADL a également recours à cette variable d'environnement.

Remarque : *sur certains ordinateurs, les caractères à double octet figurant dans les chemins d'accès de système de fichiers stockés dans ces variables d'environnement risquent d'être incorrectement interprétés. Si tel est le cas, tentez de définir le JRE utilisé pour exécuter l'outil ADT de sorte à utiliser le jeu de caractères UTF-8. Tel est le cas par défaut dans le script de lancement de l'outil ADT sous Mac et Linux. Dans le fichier Windows adt.bat ou si vous exécutez l'outil ADT directement à partir de Java, spécifiez l'option* `-Dfile.encoding=UTF-8` *sur la ligne de commande Java.*

Chapitre 13 : Signature d'applications AIR

Signature numérique d'un fichier AIR

Le fait de signer numériquement vos fichiers d'installation AIR à l'aide d'un certificat délivré par une autorité de certification reconnue rassure pleinement vos utilisateurs sur l'état de l'application qu'ils installent : cette signature garantit que l'application n'a pas été modifiée accidentellement, ou intentionnellement dans le but de nuire, et vous identifie en tant signataire (éditeur). AIR affiche le nom de l'éditeur pendant l'installation lorsque l'application AIR est signée à l'aide d'un certificat approuvé ou *lié par une chaîne de certificats* à un certificat approuvé sur l'ordinateur utilisé pour l'installation :



Boîte de dialogue de confirmation de l'installation d'une application signée par un certificat approuvé

Si vous signez une application à l'aide d'un certificat auto-signé (ou d'un certificat qui n'est pas lié par une chaîne à un certificat approuvé), l'utilisateur doit accepter que les risques sécuritaires découlant de l'installation de l'application sont plus importants. La boîte de dialogue d'installation fait état de ces risques supplémentaires :

Boîte de dialogue de confirmation de l'installation d'une application signée par un certificat auto-signé

***Important :** des personnes mal intentionnées peuvent falsifier un fichier AIR en usurpant votre identité si elles obtiennent le fichier magasin de signatures ou découvrent votre clé privé.*

Certificats développeur

Les obligations légales, restrictions et garanties de sécurité impliquant l'utilisation de certificats de signature de code sont brièvement exposées dans les déclarations des pratiques de certification – ou déclarations CPS (Certificate Practice Statements) – et les contrats d'abonnés publiés par l'autorité de certification émettrice. Pour plus d'informations sur les contrats des autorités de certification publiant actuellement des certificats développeurs AIR, voir les documents suivants :

ChosenSecurity (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

ChosenSecurity CPS (http://www.chosensecurity.com/resource_center/repository.htm)

GlobalSign (http://www.globalsign.com/code-signing/index.html)

GlobalSign CPS (http://www.globalsign.com/repository/index.htm)

Déclaration CPS de Thawte (http://www.thawte.com/cps/index.html) en anglais

Enoncé des pratiques de certification de VeriSign (EPC)4 (http://www.verisign.com/repository/CPS/)

Contrat d'abonnement de VeriSign (https://www.verisign.fr/repository/index.html)

A propos de la signature de code AIR

Lorsqu'un fichier AIR est signé, une signature numérique est ajoutée dans le fichier d'installation. La signature comprend un résumé du package, utilisé pour vérifier que le fichier AIR n'a pas été modifié depuis qu'il a été signé, et englobe des informations sur le certificat de signature qui permettent de vérifier l'identité de l'éditeur.

L'environnement AIR utilise l'infrastructure de clé publique (PKI) qui est prise en charge par le biais du magasin de certificats du système d'exploitation afin de déterminer la fiabilité d'un certificat. Pour que les informations de l'éditeur soient contrôlées, l'ordinateur sur lequel une application AIR est installée doit faire confiance directement au certificat utilisé pour signer l'application AIR, ou faire confiance à une chaîne de certificats liant ce certificat à une autorité de certification approuvée.

Si un fichier AIR est signé à l'aide d'un certificat n'appartenant pas à une chaîne le rattachant à l'un des certificats racines approuvés (et en règle générale ceci englobe tous les certificats auto-signés), les informations de l'éditeur ne peuvent pas être vérifiées. Si l'environnement AIR peut déterminer que le package AIR n'a pas été modifié depuis qu'il a été signé, il n'y a par contre aucun moyen de savoir qui est l'auteur et le signataire de ce fichier.

Remarque : *un utilisateur peut choisir de faire confiance à un certificat auto-signé, dans ce cas toute application AIR signée à l'aide de ce certificat affiche le nom de l'éditeur en guise de valeur du champ de nom commun dans le certificat. L'environnement AIR ne fournit aucune possibilité à l'utilisateur de désigner un certificat comme étant approuvé. Le certificat (sans la clé privée) doit être fourni séparément à l'utilisateur et celui-ci doit utiliser l'un des moyens proposés par le système d'exploitation, ou tout autre outil approprié, pour importer le certificat à l'emplacement approprié dans le magasin de certificats du système.*

A propos des identifiants d'éditeur AIR

Important : *depuis la version 1.5.3 d'AIR, l'utilisation de l'identifiant d'éditeur est déconseillée et ce dernier n'est plus calculé à partir du certificat développeur. Les nouvelles applications ne requièrent plus d'identifiant d'éditeur et ne devraient pas l'utiliser. Lors de la mise à jour d'applications existantes, vous devez stipuler l'identifiant d'éditeur original dans le fichier descripteur d'application.*

Avant la version 1.5.3 d'AIR, le programme d'installation d'une application AIR générait un identifiant d'éditeur lors de l'installation d'un fichier AIR. Cet identifiant était propre au certificat de signature du fichier AIR. Si vous réutilisiez le même certificat pour plusieurs applications AIR, le même identifiant d'éditeur leur était appliqué. La signature d'une mise à jour d'application par le biais d'un autre certificat, voire d'une occurrence renouvelée du certificat original entraînait la modification de l'identifiant d'éditeur.

Depuis la version 1.5.3 d'AIR, aucun identifiant d'éditeur n'est affecté par AIR. Le fichier descripteur d'une application publiée avec AIR 1.5.3 peut stipuler une chaîne d'identifiant d'éditeur. Ne stipulez d'identifiant d'éditeur que lors de la publication de mises à jour d'applications originellement associées aux versions d'AIR antérieures à 1.5.3. Si vous ne stipulez pas d'identifiant original dans le fichier descripteur d'application, le nouveau package AIR n'est pas traité comme une mise à jour de l'application existante.

Pour déterminer l'identifiant d'éditeur original, recherchez le fichier `publisherid` dans le sous-répertoire META-INF/AIR du répertoire d'installation de l'application originale. La chaîne que contient ce fichier correspond à l'identifiant d'éditeur. Le fichier descripteur de l'application doit stipuler le moteur d'exécution d'AIR 1.5.3 (ou version ultérieure) dans la déclaration d'espace de noms pour que vous puissiez stipuler manuellement l'identifiant d'éditeur.

S'il existe, l'identifiant d'éditeur est utilisé comme suit :

- Dans la clé de chiffrement destinée au magasin local chiffré
- Dans le chemin du répertoire de stockage d'application
- Dans la chaîne de connexion associée aux connexions locales
- Dans la chaîne d'identité destinée à appeler une application par le biais de l'API intégrée au navigateur d'AIR
- Dans l'OSID (définition d'interface de service ouverte) utilisée lors de la création de programmes d'installation/de désinstallation personnalisés

Toute modification de l'identifiant d'éditeur entraîne un changement de comportement de toute fonctionnalité AIR basée sur celui-ci. Il devient, par exemple, impossible d'accéder aux données stockées dans le magasin local chiffré et la chaîne de connexion associée à toute occurrence de Flash ou d'AIR qui crée une connexion locale à l'application doit contenir le nouvel identifiant. L'identifiant d'éditeur d'une application installée ne doit pas être modifiée dans AIR 1.5.3 ou ultérieur. Si vous utilisez un autre identifiant d'éditeur lorsque vous publiez un package AIR, le programme d'installation traite le nouveau package comme une autre application plutôt qu'une mise à jour.

A propos des formats de certificats

Les outils de signature AIR acceptent tous les magasins de clés accessibles par l'intermédiaire de l'architecture de chiffrement Java, ou architecture JCA (Java Cryptography Architecture). Ceci englobe les fichiers de magasins de clés, comme les fichiers au format PKCS12 (utilisant généralement une extension de fichier .pfx ou p12), les fichiers .keystore de Java, les magasins de clés matériels PKCS11 et les magasins de clés système. Les formats de magasins de clés auxquels l'outil ADT peut accéder dépendent de la version et de la configuration du moteur d'exécution Java qui est utilisé pour exécuter cet outil. L'accès à certains types de magasins de clés, comme les jetons (périphériques de sécurité) matériels PKCS11, peut nécessiter l'installation et la configuration de pilotes logiciels et de modules d'extension JCA supplémentaires.

Pour signer des fichiers AIR, vous pouvez utiliser la plupart des certificats développeurs existants ou en obtenir un nouveau publié expressément pour la signature des applications AIR. Vous pouvez par exemple utiliser l'un des types de certificats suivants publiés par VeriSign, Thawte, GlobalSign ou ChosenSecurity :

- ChosenSecurity
  - ID d'éditeur TC pour Adobe AIR

- GlobalSign
  - Certificat développeur ObjectSign
- Thawte :
  - Certificat de développeur AIR
  - Certificat de développeur Apple
  - Certificat de développeur JavaSoft
  - Certificat Microsoft Authenticode
- VeriSign :
  - ID numérique Adobe AIR
  - ID numérique Microsoft Authenticode
  - ID numérique Sun Java Signing

Remarque : *le certificat doit être créé pour la signature de code. Vous ne pouvez pas utiliser de certificat SSL ou d'autres types de certificats pour signer des fichiers AIR.*

Horodatages

Lorsque vous signez un fichier AIR, l'outil de création de packages fait une demande auprès du serveur d'une autorité d'horodatage en vue d'obtenir une date et une heure de signature pouvant être vérifiées indépendamment. L'horodatage obtenu est incorporé au fichier AIR. Tant que le certificat utilisé pour la signature est valable au moment de cette signature, il est toujours possible d'installer le fichier AIR, même après l'expiration du certificat. Par contre, si aucun horodatage n'est obtenu, le fichier AIR perd toute possibilité d'installation lorsque le certificat expire ou est révoqué.

Par défaut, les outils de création de packages AIR obtiennent un horodatage. Cependant, afin de permettre la mise en package des applications malgré une indisponibilité du service d'horodatage, vous pouvez désactiver l'option d'horodatage. Adobe recommande la présence d'un horodatage dans tout fichier AIR distribué publiquement.

L'autorité d'horodatage par défaut qui est utilisée par les outils de création de package AIR est GeoTrust.

Obtention d'un certificat

Pour obtenir un certificat, il suffit généralement de se rendre sur le site Web de l'autorité de certification et de suivre la procédure d'acquisition indiquée par la société. Les outils utilisés pour élaborer le fichier de magasin de clés nécessaire aux outils AIR dépendent du type de certificat acheté, de la façon dont le certificat est stocké sur l'ordinateur receveur et, dans certains cas, du navigateur utilisé pour obtenir ce certificat. Par exemple, pour obtenir et exporter un certificat Adobe Developer publié par Thawte, vous devez utiliser Mozilla Firefox. Le certificat peut ensuite être exporté directement sous forme de fichier .p12 ou .pfx à partir de l'interface utilisateur de Firefox.

Remarque : *la version 1.5 et les versions ultérieures de Java ne prennent pas en charge les caractères ASCII étendus dans les mots de passe de protection des fichiers de certificats PKCS12. Les outils de développement AIR font appel à Java pour créer les packages AIR signés. Si vous exportez le certificat au format .p12 ou .pfx, le mot de passe ne doit contenir que des caractères ASCII standard.*

Vous pouvez générer un certificat auto-signé par le biais de l'outil ADT utilisé pour la mise en package des fichiers d'installation AIR. Il est également possible d'utiliser d'autres outils tiers.

Pour consulter la procédure de création d'un certificat auto-signé, ainsi que les instructions de signature d'un fichier AIR, voir « Outil AIR Developer (ADT) » à la page 174. Vous pouvez également exporter et signer des fichiers AIR par l'intermédiaire de Flash Builder, Dreamweaver et de la mise à jour AIR pour Flash.

L'exemple suivant décrit la procédure d'obtention d'un certificat de développeur AIR auprès de l'autorité de certification Thawte, et sa préparation pour l'utiliser avec l'outil ADT.

Exemple : obtention d'un certificat de développeur AIR auprès de l'autorité Thawte

Remarque : *cet exemple n'est qu'une illustration parmi les nombreuses possibilités disponibles pour acquérir et préparer un certificat de signature de code. Chaque autorité de certification possède ses propres stratégies et procédures.*

Pour acquérir un certificat de développeur AIR, le site Web de Thawte nécessite l'utilisation du navigateur Firefox de Mozilla. La clé privée pour le certificat est stockée dans le magasin de clés du navigateur. Assurez-vous que le magasin de clés de Firefox est sécurisé à l'aide d'un mot de passe principal et que l'ordinateur lui-même est installé dans un endroit sûr et protégé. (Vous pouvez exporter et supprimer le certificat et la clé privée depuis le magasin de clés du navigateur une fois la procédure d'acquisition achevée.)

Au cours de la procédure d'inscription du certificat, une paire de clé privée/publique est générée. La clé privée est automatiquement stockée dans le magasin de clés Firefox. Vous devez utiliser le même navigateur sur le même ordinateur pour faire la demande d'un certificat sur le site Web de Thawte et procéder à sa récupération.

1. Visitez le site Web de Thawte et affichez la Page Produits – Certificats développeur (signature de code).
2. A partir de la liste des certificats proposés, sélectionnez le certificat du développeur Adobe® Air™.
3. Suivez les trois étapes de la procédure d'inscription. Vous devez fournir des informations sur votre organisation et les coordonnées de la personne à contacter. Thawte procède ensuite à l'opération du contrôle d'identité, ce qui peut faire l'objet d'une demande de renseignements complémentaires. Cette vérification achevée, un courrier électronique vous est adressé, il contient des instructions sur la procédure de récupération du certificat.

   Remarque : *vous trouverez des informations supplémentaires sur la nature des documents à fournir en suivant ce lien : https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.*
4. Récupérez le certificat émis depuis le site Thawte. Le certificat est automatiquement enregistré dans le magasin de clés Firefox.
5. Exportez un fichier de magasin de clés contenant la clé privée et le certificat à partir du magasin de clés Firefox en procédant comme suit :

   Remarque : *lorsqu'ils sont exportés à partir de Firefox, la clé privée et le certificat sont transmis dans un format p12 (pfx) exploitable par l'outil ADT, Flex, Flash et Dreamweaver.*

   a. Ouvrez la boîte de dialogue *Gestionnaire de certificats* :
   b. Sous Windows : ouvrez Outils > Options > Avancé > Chiffrement > Afficher les certificats.
   c. Sous Mac OS : ouvrez Firefox > Préférences > Avancé > Chiffrement > Afficher les certificats.
   d. Sous Linux : ouvrez Edition > Préférences > Avancé > Chiffrement > Afficher les certificats.
   e. Sélectionnez le certificat développeur Adobe AIR à partir de la liste des certificats et cliquez sur le bouton **Sauvegarder**.
   f. Entrez un nom de fichier et indiquez l'emplacement vers lequel exporter le fichier de magasin de clés, puis cliquez sur **Enregistrer**.
   g. Si vous utilisez la fonction de mot de passe principal de Firefox et que vous voulez exporter le fichier, vous êtes invité à préciser votre mot de passe pour le dispositif logiciel de sécurité. (Ce mot de passe n'est utilisé que par Firefox.)
   h. Dans la boîte de dialogue *Choisir un mot de passe de sauvegarde du certificat*, créez un mot de passe pour le fichier de magasin de clés.

Important : *ce mot de passe protège le fichier de magasin de clés et il est requis lorsque le fichier est utilisé pour signer des applications AIR. Votre choix devrait se porter de préférence sur un mot de passe sécurisé.*

i Cliquez sur OK. Vous devriez recevoir un message de sauvegarde du mot de passe vous informant que l'opération a réussi. Le fichier de magasin de clés, qui contient la clé privée et le certificat, est sauvegardé au moyen d'une extension de fichier .p12 (au format PKCS12).

6 Utilisez le fichier de magasin de clés exporté avec l'outil AIR Developer (ADT), Flash Builder, Flash Professional ou Dreamweaver. Le mot de passe créé pour le fichier est demandé chaque fois qu'une application AIR est signée.

Important : *la clé privée et le certificat demeurent stockés dans le magasin de clés Firefox. Si cette procédure vous permet d'exporter un exemplaire supplémentaire du fichier de certificat, vous obtenez également un autre point d'accès qu'il faut absolument protéger pour préserver la sécurité de votre certificat et de votre clé privée.*

Changement de certificats

Il s'avère parfois nécessaire de changer de certificat pour signer les mises à jour de l'application AIR. Cette opération est ainsi requise dans les circonstances suivantes :

- Renouvellement du certificat développeur original
- Mise à niveau à partir d'un certificat auto-signé vers un certificat émis par une autorité de certification
- Changement d'un certificat auto-signé sur le point d'expirer
- Changement d'un certificat commercial, par exemple lorsque les informations relatives à l'identité de votre société sont modifiées

Pour qu'AIR traite un fichier AIR comme une mise à jour, vous devez signer à la fois les fichiers AIR originaux et les fichiers AIR de mise à jour à l'aide du même certificat ou appliquer une signature de migration de certificat à la mise à jour. Une signature de migration est une seconde signature appliquée au package AIR de mise à jour à l'aide du certificat original. La signature de migration se base sur le certificat original pour attester que le signataire est bien l'éditeur original de l'application.

Lorsqu'un fichier AIR doté d'une signature de migration est installé, le nouveau certificat devient le certificat principal. Les mises à jour suivantes ne nécessitent pas de signature de migration. Dans la mesure du possible, vous devez toutefois continuer à appliquer des signatures de migration pour prendre en compte les utilisateurs qui n'effectuent pas toutes les mises à jour.

Important : *vous devez changer de certificat et appliquer une signature de migration à la mise à jour avec le certificat original avant son expiration. Si vous n'effectuez pas cette procédure, les utilisateurs doivent désinstaller la version existante de l'application avant d'en installer une nouvelle version. Pour AIR 1.5.3 ou ultérieur, vous pouvez appliquer une signature de migration à l'aide d'un certificat arrivé à expiration dans les 365 jours qui suivent la date d'expiration. Il est toutefois impossible d'appliquer la signature d'application principale à l'aide du certificat arrivé à expiration.*

Pour changer de certificats :

1 Créez une mise à jour de votre application.

2 Créez le package et signez le fichier AIR de mise à jour à l'aide du **nouveau** certificat.

3 Signez le fichier AIR de nouveau au moyen du certificat **original** (grâce à la commande `-migrate` de l'outil ADT).

Un fichier AIR doté d'une signature de migration est, à d'autres égards, un fichier AIR tout à fait normal. Si l'application est installée sur un système vierge de toute version originale, l'environnement AIR installe la nouvelle version de la façon habituelle.

Remarque : *avant la version 1.5.3 d'AIR, signer une application AIR à l'aide d'un certificat renouvelé ne nécessitait pas toujours une signature de migration. Depuis la version 1.5.3 d'AIR, les certificats renouvelés requièrent impérativement une signature de migration.*

Pour appliquer une signature de migration, utilisez la « Commande ADT migrate » à la page 182, comme le décrit la section « Signature d'une version mise à jour d'une application AIR » à la page 211.

Remarque : *La commande ADT migrate ne peut pas être utilisée avec les applications de bureau AIR incluant des extensions natives, car ces applications ont été mises en package comme des programmes d'installation natifs, pas comme des fichiers .air. Pour modifier les certificats pour une application AIR qui inclut une extension native, mettez l'application en package en utilisant la « Commande ADT package » à la page 175 avec l'indicateur -migrate.*

Changement d'identité dans une application

Dans les versions d'AIR antérieures à 1.5.3, l'identité d'une application AIR changeait lors de l'installation d'une mise à jour à laquelle était appliquée une signature de migration. Modifier l'identité d'une application a diverses conséquences, à savoir :

- La nouvelle version de l'application ne peut pas accéder aux données contenues dans le magasin local chiffré existant.
- L'emplacement du répertoire de stockage de l'application change. Les données situées à l'ancien emplacement ne sont pas copiées dans le nouveau répertoire. (Par contre, la nouvelle application peut localiser le répertoire d'origine en fonction de l'ancien ID d'éditeur).
- L'application ne peut plus ouvrir une seule connexion locale au moyen de l'ancien ID d'éditeur.
- La chaîne d'identité permettant d'accéder à une application à partir d'une page Web change.
- L'OSID (définition d'interface de service ouverte) de l'application change. (L'OSID est utilisée lors de la création de programmes d'installation/de désinstallation personnalisés.)

L'identité de l'application ne doit pas changer lors de la publication d'une mise à jour avec AIR 1.5.3 ou ultérieur. Le descripteur d'application du fichier AIR de mise à jour doit comporter les identifiants d'application et d'éditeur originaux. Le nouveau package n'est sinon pas traité comme une mise à jour.

Remarque : *lorsque vous publiez une nouvelle application avec AIR 1.5.3 ou une version ultérieure, ne stipulez pas d'identifiant d'éditeur.*

Terminologie

Cette section propose un glossaire des termes essentiels à votre compréhension lorsque vous décidez de signer une application pour la distribuer publiquement.

Terme	Description
Autorité de certification	Entité dans un réseau à infrastructure de clé publique qui sert de tiers de confiance et qui, en définitive, atteste de l'identité du propriétaire d'une clé publique. Normalement, une autorité de certification émet des certificats numériques, signés par sa propre clé privée, pour attester qu'elle a effectivement contrôlé l'identité du détenteur du certificat.
Déclaration des pratiques de certification (Déclaration CPS)	Présente les différentes pratiques et stratégies de l'autorité de certification dans l'émission et la vérification des certificats. La déclaration des pratiques de certification, ou déclaration CPS (Certification Practice Statement), fait partie intégrante du contrat qui lie l'autorité de certification avec ses abonnés et parties de confiance. Elle décrit également dans les grandes lignes les stratégies élaborées pour la vérification d'identité et le niveau des garanties offertes par les certificats fournis.
Liste de révocation de certificats	Liste des certificats émis qui ont été révoqués et qui ne devraient plus être considérés comme dignes de confiance. L'environnement d'exécution AIR vérifie la liste de révocation des certificats à la signature d'une application AIR et, si aucun horodatage n'est présent, renouvelle l'opération lorsque l'application est installée.
Chaîne de certificats	Une chaîne de certificats est une séquence de certificats dans laquelle chaque certificat présent a été signé par le certificat qui lui succède.
Certificat numérique	Document numérique contenant des informations sur l'identité du propriétaire, la clé publique du propriétaire et l'identité du certificat lui-même. Un certificat émis par une autorité de certification est lui-même signé par un certificat appartenant à l'autorité de certification émettrice.
Signature numérique	Résumé ou message chiffré ne pouvant être déchiffré qu'à l'aide de la paire clé publique et moitié de la clé publique-privée. Dans une infrastructure de clé publique (PKI), une signature numérique contient un ou plusieurs certificats numériques qui, en bout de chaîne, remontent jusqu'à l'autorité de certification. Une signature numérique peut servir à garantir qu'un message (ou un fichier informatique) n'a subi aucune modification depuis sa signature (dans les limites de la garantie fournie par l'algorithme de chiffrement utilisé) et, en admettant que l'autorité de certification émettrice soit jugée digne de confiance, à attester de l'identité du signataire.
Magasin de clés	Base de données contenant des certificats numériques et, dans certains cas, les clés privées associées.
architecture de chiffrement Java (architecture JCA)	Architecture extensible propre à la gestion et à l'accès des magasins de clés. Pour plus d'informations, voir le Guide de référence de l'architecture JCA.
PKCS #11	Norme de chiffrement d'interface pour jeton élaborée par RSA Laboratories. Un magasin de clés sur jeton (périphérique de sécurité) matériel.
PKCS #12	Norme décrivant la syntaxe des échanges d'informations personnelles élaborée par RSA Laboratories. Fichier de magasin de clés ; il contient habituellement une clé privée et son certificat numérique associé.
Clé privée	Système de chiffrement asymétrique de la moitié privée de la clé à deux composants public-privé. La clé privée doit être conservée dans un endroit secret et ne devrait jamais être transmise via un réseau. Les messages signés numériquement sont chiffrés par le signataire au moyen de la clé privée.
Clé publique	Système de chiffrement asymétrique de la moitié publique de la clé à deux composants public-privé. La clé publique est disponible sans réserve : elle sert à déchiffrer des messages chiffrés à l'aide de la clé privée.

Terme	Description
Infrastructure de clé publique (PKI)	Système basé sur l'approbation dans lequel les autorités de certification garantissent l'identité des propriétaires de clés publiques. Les clients du réseau font confiance aux certificats numériques émis par une autorité de certification approuvée pour vérifier l'identité du signataire d'un message numérique (ou d'un fichier).
Horodatage	Donnée signée numériquement qui contient la date et l'heure auxquelles un événement est survenu. L'outil ADT peut englober un horodatage à partir d'un serveur de temps RFC 3161 (en anglais) dans un package AIR. Lorsqu'il est présent, l'environnement AIR utilise l'horodatage pour établir la validité d'un certificat au moment de la signature. Ceci permet l'installation d'une application AIR après l'expiration du certificat de signature.
Autorité d'horodatage	Organisation ayant autorité pour émettre des horodatages. Pour être reconnu par l'environnement AIR, l'horodatage doit être conforme au protocole RFC 3161 et la signature de cet horodatage doit être liée par une chaîne de certificats à un certificat racine de confiance sur l'ordinateur d'installation.

Certificats iOS

Les certificats développeurs délivrés par Apple permettent de signer les applications iOS, notamment celles qui sont développées dans Adobe AIR. Il est obligatoire d'appliquer une signature à l'aide d'un certificat de développement Apple pour installer une application sur un périphérique de test. L'application d'une signature par le biais d'un certificat de distribution est obligatoire pour distribuer l'application finalisée.

Pour signer une application, l'outil ADT doit pouvoir accéder au certificat développeur et à la clé privée associée. Le fichier de certificat en tant que tel ne comprend pas de clé privée. Vous devez créer un keystore sous forme de fichier d'échange d'informations personnelles (.p12 or .pfx) qui contient à la fois le certificat et la clé privée. Voir « Conversion d'un certificat de développement en fichier de keystore P12 » à la page 209.

Génération d'une demande de signature de certificat

Pour obtenir un certificat de développement, vous générez une demande de signature de certificat, que vous envoyez au portail iOS Provisioning Portal d'Apple.

Le processus de demande de signature de certificat génère une paire clé publique-clé privée. La clé privée demeure sur l'ordinateur. Vous envoyez la demande de signature de certificat contenant la clé publique et les informations d'identification à Apple, qui fait office d'autorité de certification. Apple signe le certificat par le biais de son propre certificat WWDR (World Wide Developer Relations).

Génération d'une demande de signature de certificat sous Mac OS

Sous Mac OS, vous disposez de l'application Trousseau d'accès pour générer une demande de signature de code. L'application Trousseau d'accès réside dans le sous-répertoire Utilitaires du répertoire Applications. Vous trouverez des instructions de génération de la demande de signature de certificat sur le portail iOS Provisioning Portal d'Apple.

Génération d'une demande de signature de certificat sous Windows

Il est recommandé aux développeurs Windows d'obtenir le certificat de développement iPhone sur un ordinateur Mac. Il leur est toutefois possible d'obtenir ce certificat sous Windows. Commencez par créer une demande de signature de certificat (fichier CSR) par le biais d'OpenSSL en procédant comme suit :

1. Installez OpenSSL sur l'ordinateur Windows. (Accédez à http://www.openssl.org/related/binaries.html.)

   Il sera peut-être nécessaire d'installer également les fichiers redistribuables Visual C++ 2008, recensés sur la page de téléchargement OpenSSL. (Il est toutefois *inutile* d'installer Visual C++ sur l'ordinateur.)

2. Ouvrez une session de commande Windows et accédez au répertoire bin d'OpenSSL (c:\OpenSSL\bin\, par exemple).

3 Créez la clé privée en entrant le texte ci-dessous sur la ligne de commande :

```
openssl genrsa -out mykey.key 2048
```

Enregistrez cette clé privée. Vous en aurez besoin ultérieurement.

Lorsque vous utilisez OpenSSL, tenez compte de tous les messages d'erreur. Même si OpenSSL génère un message d'erreur, il est possible que la sortie de fichiers continue. Ces fichiers risquent toutefois d'être inutilisables. Si des messages d'erreur sont générés, vérifiez la syntaxe et exécutez à nouveau la commande.

4 Créez le fichier CSR en entrant le texte ci-dessous sur la ligne de commande :

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest  -subj
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

Remplacez l'adresse électronique, la valeur CN (nom du certificat) et la valeur C (pays) par vos coordonnées.

5 Téléchargez le fichier CSR sur le site du centre des développeurs iPhone d'Apple. (Voir « Demande de certificat de développement iPhone et création d'un profil de configuration ».)

Conversion d'un certificat de développement en fichier de keystore P12

Pour créer un keystore P12, vous devez combiner le certificat de développement Apple à la clé privée associée dans un fichier unique. Le processus de création du fichier de keystore varie selon la méthode utilisée pour générer la demande de signature du certificat original et l'emplacement de stockage de la clé privée.

Conversion du certificat de développement iPhone en fichier P12 sous Mac OS

Une fois le certificat iPhone téléchargé d'Apple, exportez-le au format keystore P12. Procédez comme suit sous Mac® OS :

1 Ouvrez l'application Trousseau d'accès (qui réside dans le dossier Applications/Utilitaires).

2 Si vous n'avez pas encore ajouté le certificat au trousseau, sélectionnez Fichier > Importer. Accédez ensuite au fichier de certificat (fichier .cer) que vous avez obtenu d'Apple.

3 Sélectionnez la catégorie Clés dans Trousseau d'accès.

4 Sélectionnez la clé privée associée au certificat de développement iPhone.

La clé privée est identifiée par le certificat public du développeur iPhone : <Prénom> <Nom> auquel elle est associée.

5 Cliquez sur le certificat de développement iPhone en appuyant sur Commande et sélectionnez *Export "iPhone Developer: Name..."*.

6 Enregistrez le keystore au format de fichier Échange d'informations personnelles (.p12).

7 Vous serez invité à créer un mot de passe utilisé lorsque vous signez des applications par le biais du keystore ou lorsque vous transférez la clé et le certificat stockés dans ce keystore vers un autre keystore.

Conversion d'un certificat de développement Apple en fichier P12 sous Windows

Pour développer des applications AIR for iOS, vous devez disposer d'un fichier de certificat P12. Vous générez ce certificat à partir du fichier de certificat de développement iPhone envoyé par Apple.

1 Convertissez le certificat de développement reçu d'Apple en certificat PEM. Exécutez l'instruction de ligne de commande suivante à partir du répertoire bin d'OpenSSL :

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```

2 Si vous utilisez la clé privée extraite du trousseau sur un ordinateur Mac, convertissez-la en clé PEM :

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```

3 Vous pouvez maintenant générer un fichier P12 valide basé sur la clé et la version PEM du certificat de développement iPhone :

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

Si vous utilisez une clé issue du trousseau Mac OS, utilisez la version PEM générée à l'étape précédente. Si tel n'est pas le cas, utilisez la clé OpenSSL générée précédemment (sous Windows).

Création d'un fichier AIR intermédiaire non signé à l'aide de l'outil ADT

La commande `-prepare` permet de créer un fichier AIR intermédiaire non signé. Un fichier AIR intermédiaire doit être signé à l'aide de la commande `-sign` d'ADT pour pouvoir générer un fichier d'installation AIR valide.

La commande `-prepare` admet les mêmes indicateurs et paramètres que la commande `-package` (à l'exception des options de signature). Ces deux commandes ne diffèrent que par le fait que le fichier de sortie est signé ou non. Le fichier intermédiaire est généré avec la même extension de nom de fichier : `airi`.

Pour signer un fichier intermédiaire AIR, utilisez la commande `-sign` d'ADT (Voir « Commande ADT prepare » à la page 182.)

Exemple d'utilisation de la commande ADT -prepare

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Signature d'un fichier intermédiaire AIR à l'aide de l'outil ADT

Pour signer un fichier intermédiaire AIR à l'aide d'ADT, utilisez la commande `-sign` d'ADT. La commande sign fonctionne uniquement avec les fichiers intermédiaires AIR (dotés de l'extension `airi`). Il est impossible de signer un fichier AIR une seconde fois.

Pour créer un fichier intermédiaire AIR, utilisez la commande `-prepare` d'ADT (Voir « Commande ADT prepare » à la page 182.)

Signature d'un fichier AIRI

❖ Faites appel à la commande `-sign` d'ADT en respectant la syntaxe suivante :

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS Les options de signature permettent d'identifier la clé privée et le certificat utilisés pour signer le fichier AIR. Ces options sont décrites à la section « Options de signature du code de l'outil ADT » à la page 189.

airi_file Chemin d'accès au fichier intermédiaire AIR non signé à signer.

air_file Nom du fichier AIR à créer.

Exemple d'utilisation de la commande ADT -sign

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Pour plus d'informations, voir « Commande ADT sign » à la page 182.

Signature d'une version mise à jour d'une application AIR

Chaque fois que vous créez une version mise à jour d'une application AIR existante, vous signez l'application mise à jour. Dans le meilleur des cas, vous pouvez signer la version mise à jour à l'aide du même certificat que celui que vous aviez utilisé pour la version précédente. La procédure est alors parfaitement identique à celle de la première signature de l'application.

Si le certificat utilisé pour signer la version précédente de l'application est arrivé à expiration ou a été remplacé, vous pouvez utiliser le certificat renouvelé ou le certificat de remplacement pour signer la version mise à jour. Pour ce faire, signez l'application avec le nouveau certificat *et* appliquez une signature de migration en utilisant le certificat d'origine. La signature de migration garantit que le propriétaire du certificat original a publié la mise à jour.

Avant d'appliquer une signature de migration, tenez compte des éléments suivants :

- Pour pouvoir appliquer une signature de migration, le certificat original doit être valide ou être arrivé à expiration il y a moins de 365 jours. La durée de ce délai est susceptible d'être modifiée ultérieurement.

  Remarque : *jusqu'à la version 2.6 d'AIR, ce délai était de 180 jours.*

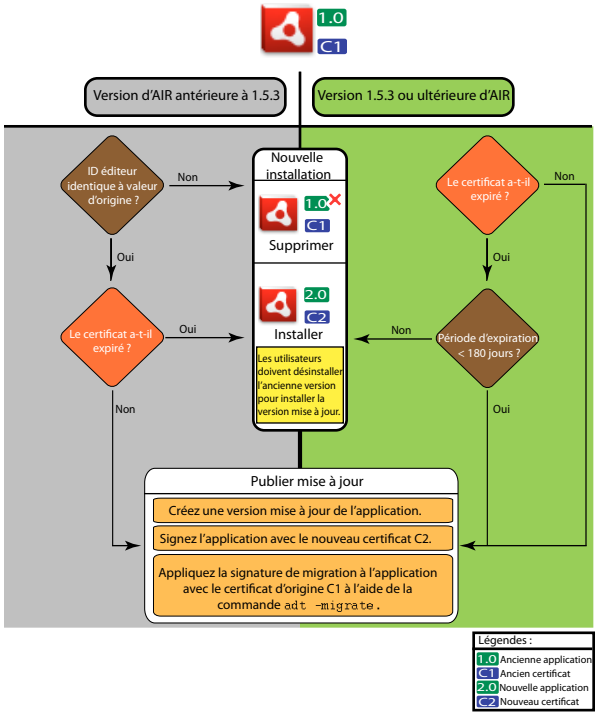
- Il est impossible d'appliquer une signature de migration au terme du délai de 365 jours qui suit l'expiration du certificat. Dans ce cas, les utilisateurs doivent désinstaller la version existante avant d'installer la version mise à jour.
- Le délai de 365 jours ne s'applique qu'aux applications pour lesquelles l'espace de noms spécifie AIR 1.5.3 ou une version ultérieure dans le fichier descripteur.

Important : *la signature de mises à jour par le biais de signatures de migration issues de certificats arrivés à expiration ne constitue qu'une solution temporaire. Pour une solution à long terme, créez un flux de travail de signature standardisé destiné à gérer le déploiement de mises à jour d'application. Par exemple, signez chaque mise à jour avec le certificat le plus récent et appliquez un certificat de migration en utilisant le certificat avec lequel la dernière version a été signée (le cas échéant). Chargez chaque mise à jour vers sa propre URL, depuis laquelle les utilisateurs peuvent télécharger l'application. Pour plus d'informations, voir «* *Flux de travail de signature associé aux mises à jour d'application » à la page 275.*

Le tableau et la figure qui suivent résument le flux de travail associé aux signatures de migration :

Scénario	État du certificat d'origine	Action du développeur	Action de l'utilisateur
Application basée sur le moteur d'exécution d'Adobe AIR 1.5.3 ou ultérieur	Valide	Publiez la version la plus récente de l'application AIR.	Aucune action requise L'application est automatiquement mise à niveau.
	Arrivé à expiration, mais le délai de 365 jours n'est pas dépassé	Signez l'application à l'aide du nouveau certificat. Appliquez une signature de migration en utilisant le certificat expiré.	Aucune action requise L'application est automatiquement mise à niveau.
	Arrivé à expiration et délai dépassé	Il est impossible d'appliquer la signature de migration à la mise à jour de l'application AIR. Vous devez plutôt publier une autre version de l'application AIR à l'aide d'un nouveau certificat. Les utilisateurs peuvent installer la nouvelle version après avoir désinstallé la version existante de l'application AIR.	Désinstallez la version actuelle de l'application AIR et installez la version la plus récente.

Scénario	État du certificat d'origine	Action du développeur	Action de l'utilisateur
• Application basée sur le moteur d'exécution d'Adobe AIR 1.5.2 ou antérieur • L'identifiant d'éditeur figurant dans le descripteur d'application de la mise à jour correspond à celui de la version précédente	Valide	Publiez la version la plus récente de l'application AIR.	Aucune action requise L'application est automatiquement mise à niveau.
	Arrivé à expiration et délai dépassé	Il est impossible d'appliquer la signature de migration à la mise à jour de l'application AIR. Vous devez plutôt publier une autre version de l'application AIR à l'aide d'un nouveau certificat. Les utilisateurs peuvent installer la nouvelle version après avoir désinstallé la version existante de l'application AIR.	Désinstallez la version actuelle de l'application AIR et installez la version la plus récente.
• Application basée sur le moteur d'exécution d'Adobe AIR 1.5.2 ou antérieur • L'identifiant d'éditeur figurant dans le descripteur d'application de la mise à jour ne correspond **pas** à celui de la version précédente	Tous les états	Signez l'application AIR à l'aide d'un certificat valide et publiez la version la plus récente de l'application AIR	Désinstallez la version actuelle de l'application AIR et installez la version la plus récente.



Flux de travail de signature associé aux mises à jour

Migration d'une application AIR en vue d'utiliser un nouveau certificat

Pour effectuer la migration d'une application AIR vers un nouveau certificat lors de la mise à jour de l'application :

1. Créez une mise à jour de votre application.
2. Créez le package et signez le fichier AIR de mise à jour à l'aide du **nouveau** certificat.
3. Signez à nouveau le fichier AIR au moyen du certificat **d'origine** en utilisant la commande `-migrate`.

Un fichier AIR signé avec la commande `-migrate` permet de mettre à jour de toute version précédente de l'application signée à l'aide de l'ancien certificat, mais aussi d'installer une nouvelle version de l'application.

Remarque : *lors de la mise à jour d'une application publiée pour une version d'AIR antérieure à 1.5.3, stipulez l'identifiant d'éditeur original dans le fichier descripteur de l'application. Les utilisateurs de l'application devront sinon désinstaller la version antérieure avant d'installer la mise à jour.*

Faites appel à la commande `-migrate` d'ADT en respectant la syntaxe suivante :

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** Les options de signature permettent d'identifier la clé privée et le certificat utilisés pour signer le fichier AIR. Ces options doivent identifier le certificat de signature **d'origine**. Elles sont décrites à la section « Options de signature du code de l'outil ADT » à la page 189.
- **air_file_in** Fichier AIR destiné à la mise à jour, signé au moyen du **nouveau** certificat.
- **air_file_out** Fichier AIR à créer.

Remarque : *les fichiers d'entrée et de sortie AIR doivent porter un nom différent.*

L'exemple montre l'appel d'ADT avec l'indicateur `-migrate` pour appliquer une signature de migration à une version mise à jour d'une application AIR :

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Remarque : *la commande* `-migrate` *a été ajoutée à l'outil ADT dans la version 1.1 d'AIR.*

Migration d'une application AIR avec programme d'installation natif afin d'utiliser un nouveau certificat

Une application AIR publiée en tant que programme d'installation natif (par exemple, une application qui utilise l'API d'extension native) ne peut pas être signée à l'aide de la commande ADT `-migrate` car il s'agit d'une application native spécifique à la plate-forme, pas d'un fichier .air. Pour effectuer la migration d'une application AIR publiée en tant qu'extension native vers un nouveau certificat, mieux vaut procéder comme suit :

1. Créez une mise à jour de votre application.
2. Assurez-vous que, dans votre fichier de descripteur d'application (app.xml), la balise `<supportedProfiles>` inclut le profil de bureau et le profil de bureau étendu (extendedDesktop) (ou retirez la balise `<supportedProfiles>` du descripteur d'application).
3. Mettez en package et signez l'application mise à jour **en tant que fichier .air** en utilisant la commande ADT `-package` avec le **nouveau** certificat.
4. Appliquez le certificat de migration au fichier .air à l'aide de la commande ADT `-migrate` en utilisant le certificat **d'origine** (comme décrit plus haut dans « Migration d'une application AIR en vue d'utiliser un nouveau certificat » à la page 213).
5. Mettez le fichier .air en package dans un programme d'installation natif en utilisant la commande ADT `-package` avec l'indicateur `-target native`. Comme l'application est déjà signée, vous n'avez pas besoin de spécifier un certificat de signature à cette étape.

L'exemple suivant illustre les étapes 3 à 5 de ce processus. Le code appelle ADT avec la commande `-package`, puis avec `-migrate` et enfin une nouvelle fois avec `-package` afin de mettre en package une version mise à jour d'une application AIR sous forme de programme d'installation natif :

```
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
```

Migration d'une application AIR qui utilise une extension native en vue d'utiliser un nouveau certificat

Une application AIR qui utilise une extension native ne peut pas être signée à l'aide de la commande ADT `-migrate`. Il est également impossible d'effectuer sa migration à l'aide de la procédure de migration d'une application AIR avec programme d'installation natif, car elle ne peut pas être publiée en tant que fichier .air intermédiaire. Pour effectuer la migration d'une application AIR qui utilise une extension native vers un nouveau certificat, mieux vaut procéder comme suit :

1. Créez une mise à jour de votre application.
2. Mettez en package et signez le programme d'installation natif de la mise à jour en utilisant la commande ADT `-package`. Mettez l'application en package avec le **nouveau** certificat, et incluez l'indicateur `-migrate` désignant le certificat **d'origine**.

Utilisez la syntaxe suivant pour appeler la commande ADT `-package` avec l'indicateur `-migrate` :

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** Les options de signature permettent d'identifier la clé privée et le certificat utilisés pour signer le fichier AIR. Ces options identifient le **nouveau** certificat de signature. Elles sont décrites à la section « Options de signature du code de l'outil ADT » à la page 189.
- **MIGRATION_SIGNING_OPTIONS** Les options de signature permettent d'identifier la clé privée et le certificat utilisés pour signer le fichier AIR. Ces options identifient le certificat de signature **d'origine**. Elles sont décrites à la section « Options de signature du code de l'outil ADT » à la page 189.
- Les autres options sont les mêmes que celles utilisées pour la mise en package d'une application AIR avec programme d'installation natif et sont décrites à la section « Commande ADT package » à la page 175.

L'exemple suivant montre comment appeler ADT avec la commande `-package` et l'indicateur `-migrate` en vue de mettre en package une version mise à jour d'une application AIR qui utilise une extension native et d'appliquer une signature de migration à la mise à jour :

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore
original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Remarque : *l'indicateur `-migrate` de la commande `-package` est disponible dans ADT dans AIR 3.6 et ultérieur.*

Création d'un certificat auto-signé à l'aide de l'outil ADT

Les certificats auto-signés permettent de produire un fichier d'installation AIR valide. Ils n'assurent toutefois qu'une sécurité limitée aux utilisateurs. Il est impossible de vérifier l'authenticité des certificats auto-signés. Lorsqu'un fichier AIR auto-signé est installé, les informations relatives à l'éditeur sont présentées à l'utilisateur comme étant inconnues. Un certificat généré par ADT est valable pendant cinq ans.

Si vous créez une mise à jour d'une application AIR signée à l'aide d'un certificat généré automatiquement, vous devez utiliser le même certificat pour signer les fichiers AIR d'origine et de mise à jour. Les certificats générés par ADT sont toujours uniques, même si les paramètres définis sont identiques pour plusieurs d'entre eux. Par conséquent, si vous souhaitez auto-signer des mises à jour au moyen d'un certificat généré par l'outil ADT, conservez en lieu sûr le certificat d'origine. De plus, vous ne pourrez pas créer de fichier AIR mis à jour après l'expiration du certificat initial généré par ADT. (Vous pouvez certes éditer de nouvelles applications dotées d'un autre certificat, mais pas de nouvelles versions de la même application.)

Important : *en raison des limites des certificats auto-signés, Adobe vous recommande vivement d'utiliser un certificat commercial émis par une autorité de certification réputée pour signer des applications AIR destinées au public.*

Le certificat et la clé privée associée générés par l'outil ADT sont stockés dans un fichier keystore de type PKCS12. Le mot de passe spécifié est défini sur la clé proprement dite, pas dans le keystore.

Exemples de génération de certificats

```
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
```

Pour signer des fichiers AIR à l'aide de ces certificats, utilisez les options de signature avec les commandes `-package` ou `-prepare` d'ADT :

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

Remarque : *la version 1.5 et les versions ultérieures de Java ne prennent pas en charge les caractères ASCII étendus dans les mots de passe de protection des fichiers de certificats PKCS12. Le mot de passe ne doit contenir que des caractères ASCII standard.*

Chapitre 14 : Fichiers descripteurs d'applications AIR

Toute application AIR requiert un fichier descripteur d'application. Le fichier descripteur d'une application est un document XML qui définit les propriétés de base de cette dernière.

De nombreux environnements de développement prenant AIR en charge génèrent automatiquement un descripteur d'application lorsque vous créez un projet. Dans le cas contraire, vous devez créer votre propre fichier descripteur. Un exemple de fichier descripteur, `descriptor-sample.xml`, se trouve dans le répertoire `samples` des kits SDK AIR et Flex.

Vous pouvez attribuer n'importe quel nom à un fichier descripteur d'application. Lorsque vous mettez en package l'application, le nom du fichier descripteur d'application est remplacé par `application.xml`. Le fichier est ensuite placé dans un répertoire spécial au sein du package AIR.

Exemple de fichier descripteur d'application

Le document descripteur d'application suivant définit les propriétés de base utilisées par la plupart des applications AIR :

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <id>example.HelloWorld</id>
    <versionNumber>1.0.1</versionNumber>
    <filename>Hello World</filename>
    <name>Example Co. AIR Hello World</name>
     <description>
        <text xml:lang="en">This is an example.</text>
        <text xml:lang="fr">C'est un exemple.</text>
        <text xml:lang="es">Esto es un ejemplo.</text>
    </description>
    <copyright>Copyright (c) 2010 Example Co.</copyright>
    <initialWindow>
        <title>Hello World</title>
        <content>
            HelloWorld.swf
        </content>
    </initialWindow>
    <icon>
        <image16x16>icons/smallIcon.png</image16x16>
        <image32x32>icons/mediumIcon.png</image32x32>
        <image48x48>icons/bigIcon.png</image48x48>
        <image128x128>icons/biggerIcon.png</image128x128>
    </icon>
</application>
```

Si l'application utilise un fichier HTML comme contenu racine plutôt qu'un fichier SWF, seul l'élément `<content>` est différent :

```
<content>
    HelloWorld.html
</content>
```

Modifications apportées au fichier descripteur d'application

Le fichier descripteur d'application AIR a été modifié dans les versions suivantes d'AIR.

Modifications apportées au fichier descripteur dans AIR 1.1

Éléments d'application autorisés `name` et `description` à localiser par le biais de l'élément `text`.

Modifications apportées au fichier descripteur dans AIR 1.5

L'élément `contentType` est devenu un enfant obligatoire de l'élément `fileType`.

Modifications apportées au fichier descripteur dans AIR 1.5.3

L'élément `publisherID` a été ajouté pour autoriser les applications à spécifier une valeur d'identifiant d'éditeur.

Modifications apportées au fichier descripteur dans AIR 2.0

Éléments ajoutés :

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (voir imageNxN)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

Modifications apportées au fichier descripteur dans AIR 2.5

Élément supprimé : `version`

Éléments ajoutés :

- `android`
- `extensionID`
- `extensions`
- `image36x36` (voir imageNxN)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

Modifications apportées au fichier descripteur dans AIR 2.6

Éléments ajoutés :

- `image114x114` (voir `imageNxN`)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Modifications apportées au fichier descripteur dans AIR 3.0

Éléments ajoutés :

- `colorDepth`
- *direct* comme valeur valide de `renderMode`
- L'élément renderMode n'est plus ignoré pour les plates-formes de bureau
- Il est possible de spécifier l'élément `<uses-sdk>` d'Android (alors que cela était impossible auparavant).

Modifications apportées au fichier descripteur dans AIR 3.1

Éléments ajoutés :

- « Entitlements » à la page 231

Modifications apportées au fichier descripteur dans AIR 3.2

Éléments ajoutés :

- `depthAndStencil`
- `supportedLanguages`

Modifications apportées au fichier descripteur dans AIR 3.3

Éléments ajoutés :

- `aspectRatio` inclut désormais l'option `ANY`.

Modifications apportées au fichier descripteur dans AIR 3.4

Éléments ajoutés :

- `image50x50` (voir « `imageNxN` » à la page 239)
- `image58x58` (voir « `imageNxN` » à la page 239)
- `image100x100` (voir « `imageNxN` » à la page 239)
- `image1024x1024` (voir « `imageNxN` » à la page 239)

Modifications apportées au fichier descripteur dans AIR 3.6

Éléments ajoutés :

- « requestedDisplayResolution » à la page 250 dans l'élément « iPhone » à la page 243 inclut désormais un attribut `excludeDevices`, vous permettant de spécifier si les cibles iOS utilisent une résolution élevée ou standard.

- Le nouvel élément « requestedDisplayResolution » à la page 250 dans « initialWindow » à la page 241 indique s'il faut utiliser une résolution élevée ou standard sur les plates-formes telles que les Mac équipés d'écrans haute résolution.

Modifications apportées au descripteur d'AIR 3.7

Éléments ajoutés :

- L'élément « iPhone » à la page 243 propose maintenant un élément « externalSwfs » à la page 233, qui permet d'indiquer une liste de fichiers SWF à charger à l'exécution.
- L'élément « iPhone » à la page 243 propose maintenant un élément « forceCPURenderModeForDevices » à la page 236, qui permet de forcer le mode de rendu UC pour un ensemble de périphériques spécifié.

Structure du fichier descripteur d'application

Le fichier descripteur d'application est un document XML dont la structure est la suivante :

```
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <allowBrowserInvocation>...<allowBrowserInvocation>
    <android>
        <colorDepth>...</colorDepth>
        <manifestAdditions
                <manifest>...</manifest>
            ]]>
        </manifestAdditions
    </android>
    <copyright>...</copyright>
    customUpdateUI>...</
    <description>
        <text xml:lang="...">...</text>
    </description>
    <extensions>
        <extensionID>...</extensionID>
    </extensions>
    <filename>...</filename>
    <fileTypes>
        <fileType>
            <contentType>...</contentType>
            <description>...</description>
            <extension>...</extension>
            <icon>
                <imageNxN>...</imageNxN>
            </icon>
            <name>...</name>
        </fileType>
    </fileTypes>
    <icon>
        <imageNxN>...</imageNxN>
    </icon>
    <id>...</id>
    <initialWindow>
        <aspectRatio>...</aspectRatio>
        <autoOrients>...</autoOrients>
```

```
        <content>...</content>
        <depthAndStencil>...</depthAndStencil>
        <fullScreen>...</fullScreen>
        <height>...</height>
        <maximizable>...</maximizable>
        <maxSize>...</maxSize>
        <minimizable>...</minimizable>
        <minSize>...</minSize>
        <renderMode>...</renderMode>
        <requestedDisplayResolution>...</requestedDisplayResolution>
        <resizable>...</resizable>
        <softKeyboardBehavior>...</softKeyboardBehavior>
        <systemChrome>...</systemChrome>
        <title>...</title>
        <transparent>...</transparent>
        <visible>...</visible>
        <width>...</width>
        <x>...</x>
        <y>...</y>
    </initialWindow>
    <installFolder>...</installFolder>
    <iPhone>
        <Entitlements>...</Entitlements>
        <InfoAdditions>...</InfoAdditions>
        <requestedDisplayResolution>...</requestedDisplayResolution>
        <forceCPURenderModeForDevices>...</forceCPURenderModeForDevices>
        <externalSwfs>...</externalSwfs>
    </iPhone>
    <name>
        <text xml:lang="...">...</text>
    </name>
    <programMenuFolder>...</programMenuFolder>
    <publisherID>...</publisherID>
    <« supportedLanguages » à la page 252>...</« supportedLanguages » à la page 252>
    <supportedProfiles>...</supportedProfiles>
    <versionNumber>...</versionNumber>
    <versionLabel>...</versionLabel>
</application>
```

Éléments du fichier descripteur d'application AIR

Le dictionnaire d'éléments suivant décrit chaque élément valide d'un fichier descripteur d'application AIR.

allowBrowserInvocation

Adobe AIR 1.0 et versions ultérieures : facultatif

Permet à l'API intégrée au navigateur AIR de détecter et lancer l'application.

Si vous définissez cette valeur sur `true`, tenez compte des implications au niveau de la sécurité. Ces implications sont décrites dans Appel d'une application AIR à partir du navigateur (développeurs ActionScript) et Appel d'une application AIR à partir du navigateur (développeurs HTML).

Pour plus d'informations, voir « Lancement d'une application AIR installée à partir du navigateur » à la page 271.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
`true` ou `false` (valeur par défaut)

Exemple
```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 et versions ultérieures : facultatif

Permet d'ajouter des éléments au fichier manifeste Android. AIR crée un fichier AndroidManifest.xml par package APK. L'élément android du fichier descripteur d'application AIR permet de lui ajouter d'autres éléments. Cet élément est ignoré sur toutes les plates-formes, à l'exception d'Android.

Élément parent :« application » à la page 222

Éléments enfants :

- « colorDepth » à la page 226
- « manifestAdditions » à la page 244

Contenu
Éléments qui définissent les propriétés propres à Android à ajouter au fichier manifeste de l'application Android.

Exemple
```
<android>
    <manifestAdditions>
        ...
    </manifestAdditions>
</android>
```

Voir aussi

« Paramètres Android » à la page 79

The AndroidManifest.xml File (disponible en anglais uniquement)

application

Adobe AIR 1.0 et versions ultérieures : obligatoire

Élément racine d'un document descripteur d'application AIR.

Élément parent : aucun

Éléments enfants :

- « allowBrowserInvocation » à la page 221
- « android » à la page 222

- « copyright » à la page 228
- « customUpdateUI » à la page 229
- « description » à la page 230
- « extensions » à la page 233
- « filename » à la page 234
- « fileTypes » à la page 235
- « icon » à la page 238
- « id » à la page 239
- « initialWindow » à la page 241
- « installFolder » à la page 242
- « iPhone » à la page 243
- « name » à la page 247
- « programMenuFolder » à la page 248
- « publisherID » à la page 248
- « supportedLanguages » à la page 252
- « supportedProfiles » à la page 252
- « version » à la page 255
- « versionLabel » à la page 255
- « versionNumber » à la page 256

Attributs

minimumPatchLevel : niveau de correctif minimal du moteur d'exécution d'AIR requis par l'application.

xmlns : l'attribut d'espace de noms XML détermine la version du moteur d'exécution d'AIR requise par l'application.

L'espace de noms est modifié pour chaque nouvelle version majeure d'AIR (mais non pour les correctifs mineurs). Le dernier segment de l'espace de noms, tel que « 3.0 », indique la version du moteur d'exécution requise par l'application.

Les valeurs xmlns associées aux versions majeures d'AIR sont les suivantes :

```
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"
```

Dans le cas des applications de type SWF, la version du moteur d'exécution d'AIR spécifiée dans le fichier descripteur d'application détermine la version SWF maximale pouvant être chargée en tant que contenu initial de l'application. Les applications qui spécifient AIR 1.0 ou AIR 1.1 ne peuvent utiliser que des fichiers SWF9 (Flash Player 9) en tant que contenu initial, même en cas d'exécution avec le moteur d'exécution d'AIR 2. Les applications qui spécifient AIR 1.5 (ou une version ultérieure) peuvent utiliser des fichiers SWF9 ou SWF10 (Flash Player 10) en tant que contenu initial.

La version SWF détermine la version des API AIR et Flash Player disponibles. Lorsqu'un fichier SWF9 est utilisé en tant que contenu initial d'une application AIR 1.5, cette application n'a accès qu'aux API AIR 1.1 et Flash Player 9. De plus, les modifications de comportement apportées aux API existantes dans AIR 2.0 ou Flash Player 10.1 n'ont aucun effet. (Les modifications de sécurité importantes apportées aux API sont une exception à ce principe et peuvent être appliquées de façon rétroactive aux correctifs présents ou futurs du moteur d'exécution.)

Pour les applications HTML, la version du moteur d'exécution spécifiée dans le descripteur d'application détermine la version des API AIR et Flash Player accessible à l'application. Les comportements HTML, CSS et JavaScript sont toujours déterminés par la version de WebKit utilisée dans le moteur d'exécution d'AIR installé, et non par le fichier descripteur d'application.

Lorsqu'une application AIR charge du contenu SWF, la version des API AIR et Flash Player disponibles pour ce contenu dépend de la manière dont le contenu est chargé. Il arrive que la version soit déterminée par l'espace de noms du descripteur d'application, par la version du contenu en cours de chargement ou par la version du contenu chargé. Le tableau suivant montre comment la version de l'API est déterminée en fonction de la méthode de chargement :

Méthode de chargement du contenu	**Méthode de détermination de la version de l'API**
Contenu initial, application de type SWF	Version SWF du fichier **loaded**
Contenu initial, application de type HTML	Espace de nom du descripteur d'application
SWF chargé par le contenu SWF	Version du contenu **loading**
Bibliothèque SWF chargée par le contenu HTML via la balise <script>	Espace de nom du descripteur d'application
Fichier SWF chargé par le contenu HTML, via les API AIR ou Flash Player (flash.display.Loader, par exemple)	Espace de nom du descripteur d'application
SWF chargé par le contenu HTML via les balises <object> ou <embed> (ou les API JavaScript équivalentes)	Version SWF du fichier **loaded**

Lors du chargement d'un fichier SWF d'une version différente de celle du contenu en cours de chargement, deux problèmes peuvent survenir :

- Chargement d'un fichier SWF d'une version récente par un fichier SWF d'une version antérieure : les références aux API ajoutées versions récentes d'AIR et de Flash Player dans le contenu chargé ne sont pas résolues.
- Chargement d'un fichier SWF d'une ancienne version par un fichier SWF d'une version récente : les API modifiées dans les versions récentes d'AIR et de Flash Player peuvent adopter un comportement auquel le contenu chargé ne s'attend pas.

Contenu

L'élément de l'application contient des éléments enfants qui définissent les propriétés d'une application AIR.

Exemple

```
<?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 et versions ultérieures, iOS et Android : facultatif

Définit le format de l'application.

Si le format n'est pas spécifié, l'application s'ouvre dans le format et l'orientation « naturels » du périphérique. L'orientation naturelle varie selon le périphérique. En règle générale, il s'agit du format portrait sur les périphériques équipés d'un écran de petite taille, tels que les téléphones. Sur certains périphériques, tels que la tablette iPad, l'application s'ouvre dans l'orientation active. Dans AIR 3.3 et les versions ultérieures, cela s'applique à la totalité de l'application et pas simplement à l'affichage initial.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

`portrait, landscape` ou `any`

Exemple

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

autoOrients

Adobe AIR 2.0 et versions ultérieures, iOS et Android : facultatif

Indique si l'orientation du contenu de l'application est automatiquement modifiée lorsque le périphérique pivote. Pour plus d'informations, voir Orientation de la scène.

Si vous utilisez l'orientation automatique, envisagez de définir les propriétés `align` et `scaleMode` de l'objet Stage sur les valeurs suivantes :

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Ces paramètres permettent à l'application de pivoter autour de l'angle supérieur gauche et interdisent la mise à l'échelle automatique de son contenu. Bien que les autres modes de mise à l'échelle ajustent le contenu en fonction des dimensions de la scène au terme de sa rotation, ils entraînent également un découpage, une distorsion ou une réduction excessive du contenu. Il est quasiment toujours préférable de redessiner le contenu ou d'en modifier à nouveau la disposition manuellement.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

`true` ou `false` (valeur par défaut)

Exemple

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

colorDepth

Adobe AIR 3 et versions ultérieures : facultatif

Indique quand utiliser la couleur 16 bits ou la couleur 32 bits.

L'utilisation de la couleur 16 bits peut augmenter les performances de rendu, au détriment de la fidélité des couleurs. Dans les versions antérieures à AIR 3, la couleur 16 bits est toujours utilisée sur Android. Dans AIR 3, la couleur 32 bits est utilisée par défaut.

Remarque : *si votre application utilise la classe StageVideo, vous devez utiliser la couleur 32 bits.*

Élément parent :« android » à la page 222

Éléments enfants : aucun

Contenu
Utilisez l'une des valeurs suivantes :

- 16 bits
- 32 bits

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

containsVideo

Indique si l'application contiendra ou non tout contenu vidéo.

Élément parent :« android » à la page 222

Éléments enfants : aucun

Contenu
Utilisez l'une des valeurs suivantes :

- true
- false

Exemple
```
<android>
    <containsVideo>true</containsVideo>
    <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 et versions ultérieures : obligatoire

La valeur indiquée de l'élément `content` correspond à l'URL du fichier de contenu principal de l'application. Il s'agit soit d'un fichier SWF, soit d'un fichier HTML. L'URL est définie relativement à la racine du dossier d'installation de l'application. (Si une application AIR est exécutée avec ADL, l'URL est relative au dossier contenant le fichier descripteur d'application. Vous disposez du paramètre `root-dir` d'ADL pour indiquer un autre répertoire racine.)

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
URL relative au répertoire de l'application. La valeur de l'élément content étant assimilée à une URL, les caractères qui composent le nom du fichier de contenu doivent respecter le codage URL, conformément aux règles définies dans RFC 1738. Les caractères espace doivent par exemple être codés sous la forme `%20`.

Exemple
```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0 à 1.1 : facultatif. AIR 1.5 et versions ultérieures : obligatoire

`contentType` est obligatoire depuis AIR 1.5 (il était facultatif dans AIR 1.0 et 1.1). Cette propriété aide certains systèmes d'exploitation à trouver l'application la plus adaptée pour ouvrir un fichier. Sa valeur doit correspondre au type MIME du contenu du fichier. Notez que la valeur est ignorée sous Linux si le type de fichier est déjà enregistré et a un type MIME attribué.

Élément parent :« fileType » à la page 234

Éléments enfants : aucun

Contenu
Type et sous-type MIME. Pour plus d'informations sur les types MIME, voir RFC2045 (disponible en anglais uniquement).

Exemple
```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 et versions ultérieures : facultatif

Informations de copyright relatives à l'application AIR. Sous Mac OS, le texte de copyright s'affiche dans la boîte de dialogue A propos de de l'application installée. Sous Mac OS, les informations de copyright sont également utilisées dans le champ NSHumanReadableCopyright du fichier Info.plist de l'application.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
Chaîne contenant les informations de copyright de l'application.

Exemple
```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 et versions ultérieures : facultatif

Indique si une application fournit ses propres boîtes de dialogue de mise à jour. Si l'élément est défini sur `false`, AIR propose des boîtes de dialogue de mise à jour standard à l'utilisateur. Seules les applications distribuées en tant que fichiers AIR disposent du système de mise à jour intégré d'AIR.

Si l'élément `customUpdateUI` est défini sur `true` dans la version installée de l'application et que l'utilisateur double-clique sur le fichier AIR pour installer une nouvelle version ou procéder à la mise à jour de l'application par le biais de la fonction d'installation transparente, le moteur d'exécution ouvre la version installée de l'application. Le moteur d'exécution n'ouvre pas le programme d'installation de l'application AIR par défaut. La logique de l'application peut alors déterminer la marche à suivre pour l'opération de mise à jour. (L'ID d'application et l'ID d'éditeur stockés dans le fichier AIR doivent être identiques aux valeurs de l'application installée pour que la mise à jour aboutisse.)

Remarque : *le mécanisme `customUpdateUI` n'intervient que lorsque l'application est déjà installée et que l'utilisateur double-clique sur le fichier d'installation AIR contenant une mise à jour ou qu'il installe une mise à jour de l'application par le biais de la fonction d'installation transparente. Que le paramètre `customUpdateUI` soit défini sur `true` ou non, vous pouvez télécharger et démarrer une mise à jour par le biais de la logique de l'application en affichant l'interface utilisateur personnalisée si besoin est.*

Pour plus d'informations, voir « Mise à jour des applications AIR » à la page 273.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu

`true` ou `false` (valeur par défaut)

Exemple

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 et versions ultérieures : facultatif

Indique que l'application requiert l'utilisation du tampon de profondeur ou du tampon de modèle. Ces tampons s'utilisent normalement avec du contenu 3D. Par défaut, la valeur de cet élément est `false` en vue de désactiver les tampons de profondeur et de modèle. Cet élément est nécessaire, car les tampons doivent être alloués au démarrage de l'application, c'est-à-dire avant le chargement de contenu.

Le paramètre de cet élément doit correspondre à la valeur transmise pour l'argument `enableDepthAndStencil` à la méthode `Context3D.configureBackBuffer()`. Si les valeurs ne correspondent pas, AIR renvoie une erreur.

Cet élément est applicable uniquement lorsque `renderMode = direct`. Si `renderMode` n'est pas égal à `direct`, ADT renvoie l'erreur 118 :

```
<depthAndStencil> element unexpected for render mode cpu.  It requires "direct" render mode.
```

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

`true` ou `false` (valeur par défaut)

Exemple

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

description

Adobe AIR 1.0 et versions ultérieures : facultatif

Description de l'application, affichée dans le programme d'installation d'une application AIR.

Si vous indiquez un nœud texte unique (au lieu de plusieurs éléments text), le programme d'installation de l'application AIR utilise cette description, quelle que soit la langue système activée. Dans le cas contraire, le programme d'installation d'une application AIR utilise la description la plus proche de la langue de l'interface utilisateur du système d'exploitation de l'utilisateur. Prenons l'exemple d'une installation dans laquelle l'élément `description` du fichier descripteur d'application comprend une valeur pour le jeu de paramètres régionaux en (anglais). Le programme d'installation d'une application AIR utilise la description en si le système de l'utilisateur identifie en (anglais) en tant que langue de l'interface utilisateur. Il utilise également la description en si la langue de l'interface utilisateur système correspond à en-US (anglais américain). Toutefois, si la langue de l'interface utilisateur système correspond à en-US et que le fichier descripteur d'application définit à la fois les noms en-US et en-GB, le programme d'installation d'AIR utilise la valeur en-US. Si l'application ne définit pas de description qui corresponde à la langue de l'interface utilisateur système, le programme d'installation d'une application AIR utilise la première valeur `description` stipulée dans le fichier descripteur de l'application.

Pour plus d'informations sur le développement d'applications multilingues, voir « Localisation d'applications AIR » à la page 309.

Élément parent :« application » à la page 222

Éléments enfants :« text » à la page 254

Contenu

Le modèle de descripteur d'application AIR 1.0 ne permet de définir qu'un seul nœud texte simple en tant que nom (plutôt que plusieurs éléments `text`).

Dans AIR 1.1 (ou version ultérieure), il est possible de spécifier plusieurs langues dans l'élément `description`. L'attribut `xml:lang` de chaque élément text définit un code de langue, comme indiqué à l'adresse suivante RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

Exemple

Description avec nœud texte simple :

```
<description>This is a sample AIR application.</description>
```

Description avec éléments text localisés en anglais, espagnol et français (valide dans AIR 1.1 et ultérieur) :

```
<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 et versions ultérieures : obligatoire

Description du type de fichier affichée à l'intention de l'utilisateur par le système d'exploitation. Il est impossible de la localiser.

Voir aussi : « description » à la page 230 en tant qu'enfant de l'élément application

Élément parent :« fileType » à la page 234

Éléments enfants : aucun

Contenu
Chaîne qui décrit le contenu du fichier.

Exemple

```
<description>PNG image</description>
```

embedFonts

Vous permet d'utiliser des polices personnalisées sur StageText dans l'application AIR. Cet élément est facultatif.

Élément parent :« application » à la page 222

Éléments enfants :« font » à la page 236

Contenu
L'élément embedFonts peut contenir un nombre illimité d'éléments font.

Exemple

```
<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 et versions ultérieures, iOS uniquement : facultatif

iOS utilise des propriétés appelées entitlements pour que l'application puisse accéder à des ressources et fonctionnalités supplémentaires. Utilisez l'élément Entitlements pour spécifier ces informations sur une application iOS mobile.

Élément parent :« iPhone » à la page 243

Éléments enfants : éléments Entitlements.plist iOS

Contenu
Contient les éléments enfants qui stipulent les paires clé-valeur faisant office de paramètres Entitlements.plist dans l'application. Le contenu de l'élément Entitlements doit être entouré d'un bloc CDATA. Pour plus d'informations, voir le document Entitlement Key Reference dans iOS Developer Library (disponible en anglais uniquement).

Exemple
```
<iphone>
...
   <Entitlements>
      <![CDATA[
         <key>aps-environment</key>
         <string>development</string>
      ]]>
   </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 et versions ultérieures : obligatoire

Chaîne correspondant à l'extension d'un type de fichier.

Élément parent :« fileType » à la page 234

Éléments enfants : aucun

Contenu
Chaîne identifiant les caractères qui composent l'extension du fichier (sans le point, « . »).

Exemple
```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 et versions ultérieures

Indique l'ID d'une extension ActionScript utilisée par l'application. L'ID est défini dans le document descripteur de l'extension.

Élément parent :« extensions » à la page 233

Éléments enfants : aucun

Contenu
Chaîne identifiant l'ID de l'extension ActionScript.

Exemple
```
<extensionID>com.example.extendedFeature</extensionID>
```

extensions

Adobe AIR 2.5 et versions ultérieures : facultatif

Identifie les extensions ActionScript utilisées par une application.

Élément parent :« application » à la page 222

Éléments enfants :« extensionID » à la page 232

Contenu
Éléments enfant `extensionID` contenant les ID d'extension ActionScript issus du fichier descripteur de l'extension.

Exemple

```
<extensions>
    <extensionID>extension.first</extensionID>
    <extensionID>extension.next</extensionID>
    <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 et versions ultérieures, iOS uniquement : facultatif

Indique le nom d'un fichier texte contenant une liste de fichiers SWF à configurer par ADT pour l'hébergement à distance. Vous pouvez réduire la taille de téléchargement initiale de l'application en compressant un sous-ensemble des fichiers SWF utilisées par votre application et en chargeant les fichiers SWF externes restants (actif uniquement) lors de l'exécution à l'aide de la méthode `Loader.load()`. Pour utiliser cette fonction, vous devez compresser l'application de telle sorte qu'ADT déplace tout le pseudo-code ActionScript (ABC) depuis les fichiers SWF chargés en externe vers le fichier SWF principal de l'application, laissant un fichier SWF qui contient uniquement des actifs. Cela est nécessaire pour respecter la règle de l'Apple Store qui interdit le téléchargement tout code une fois qu'une application est installée.

Pour plus d'informations, reportez-vous à « Réduire la taille des téléchargements en chargeant des fichiers SWF externes d'actifs uniquement » à la page 91.

Élément parent :« iPhone » à la page 243, « initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Nom d'un fichier texte contenant une liste délimitée par des virgules de ligne des fichiers SWF pour être à distance hébergé.

Attributs :
aucun.

Exemples
iOS :

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

filename

Adobe AIR 1.0 et versions ultérieures : obligatoire

Chaîne à utiliser en tant que nom de fichier de l'application (sans extension) lors de l'installation de cette dernière. Le fichier d'application démarre l'application AIR dans le moteur d'exécution. Si aucune valeur `name` n'est définie, l'élément `filename` fait également office de nom du dossier d'installation.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu

La propriété `filename` peut contenir tout caractère Unicode (UTF-8), à l'exception des caractères suivants, dont l'utilisation est interdite dans les noms de fichier dans divers systèmes de fichiers :

Caractère	Code hexadécimal
divers	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

Le dernier caractère de la valeur `filename` ne doit pas correspondre à un point.

Exemple

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 et versions ultérieures : facultatif

Décrit un type de fichier unique auquel peut être associée l'application.

Élément parent :« fileTypes » à la page 235

Éléments enfants :

- « contentType » à la page 228
- « description » à la page 231
- « extension » à la page 232
- « icon » à la page 238
- « name » à la page 248

Contenu
Éléments qui décrivent un type de fichier.

Exemple
```
<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 et versions ultérieures : facultatif

L'élément `fileTypes` permet de déclarer les types de fichiers auxquels peut être associée une application AIR.

Lors de l'installation d'une application AIR, tout type de fichier déclaré est enregistré dans le système d'exploitation. Si ces types de fichiers ne sont pas encore associés à une autre application, ils sont associés à l'application AIR. Pour remplacer une association existante entre un type de fichier et une autre application, utilisez la méthode `NativeApplication.setAsDefaultApplication()` à l'exécution (de préférence avec l'accord de l'utilisateur).

Remarque : *les méthodes d'exécution ne gèrent que les associations de types de fichiers déclarés dans le descripteur d'application.*

L'élément `fileTypes` est facultatif.

Élément parent :« application » à la page 222

Éléments enfants : « fileType » à la page 234

Contenu
L'élément `fileTypes` peut contenir un nombre illimité d'éléments `fileType`.

Exemple
```
<fileTypes>
    <fileType>
        <name>adobe.VideoFile</name>
        <extension>avf</extension>
        <description>Adobe Video File</description>
        <contentType>application/vnd.adobe.video-file</contentType>
        <icon>
            <image16x16>icons/AIRApp_16.png</image16x16>
            <image32x32>icons/AIRApp_32.png</image32x32>
            <image48x48>icons/AIRApp_48.png</image48x48>
            <image128x128>icons/AIRApp_128.png</image128x128>
        </icon>
    </fileType>
</fileTypes>
```

font

Décrit une police personnalisée unique pouvant être utilisée dans l'application AIR.

Élément parent :« embedFonts » à la page 231

Éléments enfants :« fontName » à la page 236, « fontPath » à la page 236

Contenu
Éléments spécifiant le nom de la police personnalisée et son chemin d'accès.

Exemple

```
<font>
   <fontPath>ttf/space age.ttf</fontPath>
   <fontName>space age</fontName>
</font>
```

fontName

Spécifie le nom de la police personnalisée.

Élément parent :« font » à la page 236

Éléments enfants : aucun

Contenu
Nom de la police personnalisée à spécifier dans StageText.fontFamily

Exemple

```
<fontName>space age</fontName>
```

fontPath

Indique l'emplacement du fichier de police personnalisée.

Élément parent :« font » à la page 236

Éléments enfants : aucun

Contenu
Chemin d'accès au fichier de police personnalisé (en fonction de la source).

Exemple

```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 et versions ultérieures, iOS uniquement : facultatif

Forcer le mode de rendu UC pour un ensemble spécifié de périphériques. Cette fonction vous permet d'activer de manière sélective le mode de rendu GPU pour les périphériques iOS restants.

Vous ajoutez cette balise comme enfant de la balise `iPhone` et spécifiez une liste de noms de modèles séparés par des espaces. Voici des noms de modèle de périphérique valides :

iPad1,1	iPhone1,1	iPod1,1
iPad2,1	iPhone1,2	iPod2,1
iPad2,2	iPhone2,1	iPod3,3
iPad2,3	iPhone3.1	iPod4,1
iPad2,4	iPhone3,2	iPod 5,1
iPad2,5	iPhone4,1	
iPad3,1	iPhone5,1	
iPad3,2		
iPad3,3		
iPad3,4		

Élément parent :« iPhone » à la page 243, « initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Liste de noms de modèles de périphérique séparés par des espaces

Attributs :
aucun.

Exemples
iOS :

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

fullScreen

Adobe AIR 2.0 et versions ultérieures, iOS et Android : facultatif

Indique si l'application démarre en mode Plein écran.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
`true` ou `false` (valeur par défaut)

Exemple

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 et versions ultérieures : facultatif

Hauteur initiale de la fenêtre principale de l'application.

Si vous ne définissez pas de hauteur, elle est déterminée par les paramètres figurant dans le fichier SWF racine ou, dans le cas d'une application AIR de type HTML, par le système d'exploitation.

La hauteur maximale d'une fenêtre est passée de 2 048 à 4 096 pixels dans AIR 2.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Entier positif dont la valeur ne doit pas dépasser 4 095.

Exemple

```
<height>4095</height>
```

icon

Adobe AIR 1.0 et versions ultérieures : facultatif

La propriété `icon` définit un ou plusieurs fichiers d'icône à utiliser pour l'application. L'inclusion d'une icône est facultative. Si vous ne définissez pas la propriété `icon`, le système d'exploitation affiche une icône par défaut.

Le chemin défini est relatif au répertoire racine de l'application. Les fichiers d'icône doivent être au format PNG. Vous pouvez indiquer les tailles d'icône suivantes :

S'il existe un élément de taille donnée, l'image du fichier doit posséder exactement la taille définie. Si toutes les tailles ne sont pas disponibles, la taille la plus proche est mise à l'échelle en fonction de l'utilisation requise de l'icône par le système d'exploitation.

Remarque : *les icônes indiquées ne sont pas automatiquement ajoutées au package AIR. Les fichiers d'icône doivent être inclus à l'emplacement relatif correct lors de la mise en package de l'application.*

Pour obtenir de meilleurs résultats, fournissez une image dans chacune des tailles disponibles. Assurez-vous en outre que l'apparence des icônes est appropriée en mode de couleur 16 et 32 bits.

Élément parent :« application » à la page 222

Éléments enfants :« imageNxN » à la page 239

Contenu
Élément imageNxN défini pour chaque taille d'icône requise.

Exemple

```
<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 et versions ultérieures : obligatoire

Chaîne d'identifiant de l'application, appelée ID d'application. Un identifiant de type DNS inverse est souvent utilisé, mais ce type n'est pas obligatoire.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
La valeur de l'identifiant ne contient que les caractères suivants :

- 0–9
- a–z
- A–Z
- . (point)
- - (tiret)

La valeur doit comporter entre 1 et 212 caractères. Cet élément est obligatoire.

Exemple

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 et versions ultérieures : facultatif

Définit le chemin d'accès d'icône par rapport au répertoire de l'application.

Vous pouvez utiliser les images d'icône suivantes, chacune stipulant une taille d'icône distincte :

- image16x16
- image29x29 (AIR 2+)
- image32x32
- image36x36 (AIR 2.5+)
- image48x48
- image50x50 (AIR 3.4+)
- image57x57 (AIR 2+)
- image58x58 (AIR 3.4+)
- image72x72 (AIR 2+)
- image100x100 (AIR 3.4+)
- image114x114 (AIR 2.6+)
- image128x128
- image144x144 (AIR 3.4+)
- image512x512 (AIR 2+)

- image1024x1024 (AIR 3.4+)

L'icône doit être un fichier graphique PNG dont la taille correspond exactement à la valeur indiquée par l'élément image. Les fichiers d'icône doivent être inclus dans le package de l'application. Les icônes auxquelles fait référence le document descripteur de l'application ne sont pas automatiquement intégrées.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
Le chemin d'accès au fichier d'icône peut contenir tout caractère Unicode (UTF-8), à l'exception des caractères suivants, dont l'utilisation est interdite dans les noms de fichier dans divers systèmes de fichiers :

Caractère	Code hexadécimal
divers	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

Exemple

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 et versions ultérieures : facultatif

Permet de spécifier des propriétés supplémentaires d'une application iOS.

Élément parent :« iPhone » à la page 243

Éléments enfants : éléments Info.plist iOS

Contenu
Contient les éléments enfants qui stipulent les paires clé-valeur faisant office de paramètres Info.plist dans l'application. Le contenu de l'élément InfoAdditions doit être entouré d'un bloc CDATA.

Pour plus d'informations sur les paires clé-valeur et leur expression en XML, voir Information Property List Key Reference dans la bibliothèque de référence pour l'iPhone d'Apple (disponible en anglais uniquement).

Exemple

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

Voir aussi

« Paramètres iOS » à la page 86

initialWindow

Adobe AIR 1.0 et versions ultérieures : obligatoire

Définit le principal fichier de contenu et l'aspect initial de l'application.

Élément parent :« application » à la page 222

Éléments enfants : tous les éléments suivants peuvent servir d'enfants de l'élément initialWindow. Selon qu'AIR prend en charge ou non les fenêtres sur une plate-forme, certains éléments sont toutefois ignorés :

Élément	Bureau	Mobile
« aspectRatio » à la page 225	ignoré	utilisé
« autoOrients » à la page 226	ignoré	utilisé
« content » à la page 227	utilisé	utilisé
« depthAndStencil » à la page 229	utilisé	utilisé
« fullScreen » à la page 237	ignoré	utilisé
« height » à la page 238	utilisé	ignoré
« maximizable » à la page 245	utilisé	ignoré
« maxSize » à la page 246	utilisé	ignoré
« minimizable » à la page 246	utilisé	ignoré
« minSize » à la page 246	utilisé	ignoré
« renderMode » à la page 249	utilisé (AIR 3.0 et versions ultérieures)	utilisé
« requestedDisplayResolution » à la page 250	utilisé (AIR 3.6 et ultérieur)	ignoré
« resizable » à la page 251	utilisé	ignoré
« softKeyboardBehavior » à la page 251	ignoré	utilisé
« systemChrome » à la page 253	utilisé	ignoré

Élément	Bureau	Mobile
« title » à la page 254	utilisé	ignoré
« transparent » à la page 255	utilisé	ignoré
« visible » à la page 256	utilisé	ignoré
« width » à la page 257	utilisé	ignoré
« x » à la page 257	utilisé	ignoré
« y » à la page 257	utilisé	ignoré

Contenu

Éléments enfants qui définissent l'aspect et le comportement d'une application.

Exemple

```
<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 et versions ultérieures : facultatif

Identifie le sous-répertoire du répertoire d'installation par défaut.

Sous Windows, le sous-répertoire d'installation par défaut est le répertoire Program Files. Sous Mac OS, il s'agit du répertoire /Applications. Sous Linux, il s'agit du répertoire /opt/. Par exemple, si la propriété `installFolder` est définie sur « `Acme` » et qu'une application s'appelle « `ExempleApp` », l'application est installée dans C:\Program Files\Acme\ExempleApp sous Windows, dans /Applications/Acme/Exemple.app sous Mac OS et dans /opt/Acme/ExampleApp sous Linux.

La propriété `installFolder` est facultative. Si vous ne spécifiez pas la propriété `installFolder`, l'application est installée dans un sous-répertoire du répertoire d'installation par défaut, basé sur la propriété `name`.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
La propriété `installFolder` peut comporter n'importe quel caractère Unicode (UTF-8), à l'exception de ceux dont l'utilisation est interdite dans les noms de fichier dans divers systèmes de fichiers (voir la propriété `filename`, qui contient la liste d'exceptions).

La barre oblique (/) fait office de caractère de séparation de répertoire pour définir un sous-répertoire imbriqué.

Exemple
```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, iOS uniquement : facultatif

Définit les propriétés d'une application propres à iOS.

Élément parent :« application » à la page 222

Éléments enfants :

- « Entitlements » à la page 231
- « externalSwfs » à la page 233
- « forceCPURenderModeForDevices » à la page 236
- « InfoAdditions » à la page 240
- « requestedDisplayResolution » à la page 250

Voir aussi

« Paramètres iOS » à la page 86

manifest

Adobe AIR 2.5 et versions ultérieures, Android uniquement : facultatif

Spécifie les informations à ajouter au fichier manifeste Android associé à l'application.

Élément parent :« manifestAdditions » à la page 244

Éléments enfants : définis par le kit SDK d'Android.

Contenu
D'un point de vue technique, l'élément manifest ne fait pas partie du modèle de descripteur d'une application AIR. Il correspond à la racine du document XML manifeste Android. Tout contenu intégré à l'élément manifest doit impérativement respecter le modèle AndroidManifest.xml. Lorsque vous générez un fichier APK à l'aide des outils AIR, les informations que contient l'élément manifest sont copiées dans la section correspondante du fichier AndroidManifest.xml généré de l'application.

Si vous spécifiez des valeurs de l'élément manifest d'Android uniquement disponibles dans une version du kit SDK plus récente que celle directement prise en charge par AIR, vous devez définir l'indicateur `-platformsdk` sur ADT lors de la mise en package de l'application. Définissez l'indicateur sur le chemin du système de fichiers pointant vers une version du kit SDK d'Android qui prend en charge les valeurs que vous ajoutez.

L'élément manifest à proprement parler doit être délimité par un bloc CDATA dans le fichier descripteur d'application AIR.

Exemple

```
<![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>
]]>
```

Voir aussi

« Paramètres Android » à la page 79

The AndroidManifest.xml File (disponible en anglais uniquement)

manifestAdditions

Adobe AIR 2.5 et versions ultérieures, Android uniquement

Spécifie les informations à ajouter au fichier manifeste Android.

Chaque application Android comprend un fichier manifeste qui définit les propriétés de base de l'application. En tant que concept, le fichier manifeste Android est similaire au fichier descripteur d'application AIR. Une application AIR for Android comprend à la fois un fichier descripteur d'application et un fichier manifeste Android généré automatiquement. Lorsqu'une application AIR for Android est mise en package, les informations que contient l'élément `manifestAdditions` sont ajoutées aux parties correspondantes du document manifeste Android.

Élément parent :« android » à la page 222

Éléments enfants :« manifest » à la page 243

Contenu

Les informations que contient l'élément `manifestAdditions` sont ajoutées au document XML AndroidManifest.

AIR définit plusieurs entrées manifeste dans le document manifeste Android généré pour s'assurer que les fonctionnalités de l'application et du moteur d'exécution fonctionnent correctement. Il est impossible de remplacer les paramètres suivants :

Il est impossible de définir les attributs suivants de l'élément manifest :

- package
- android:versionCode
- android:versionName

Il est impossible de définir les attributs suivants de l'élément activity principal :

- android:label

- android:icon

Il est impossible de définir les attributs suivants de l'élément application :

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

Exemple

```
<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>
```

Voir aussi

« Paramètres Android » à la page 79

The AndroidManifest.xml File (disponible en anglais uniquement)

maximizable

Adobe AIR 1.0 et versions ultérieures : facultatif

Spécifie si la fenêtre peut être agrandie.

Remarque : *sur des systèmes d'exploitation qui assimilent l'agrandissement d'une fenêtre à une opération de redimensionnement, tel Mac OS X, les paramètres maximizable et resizable doivent être définis sur* `false` *pour interdire l'exécution d'un zoom ou d'un redimensionnement de la fenêtre.*

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

`true` (valeur par défaut) ou `false`

Exemple

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 et versions ultérieures : facultatif

Tailles maximales de la fenêtre. Si vous ne définissez pas de taille maximale, elle est déterminée par le système d'exploitation.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Deux entiers qui représentent la largeur et la hauteur maximales, séparés par un espace blanc.

Remarque : *la taille de fenêtre maximale prise en charge par AIR est passée de 2 048x2 048 à 4 096x4 096 pixels dans AIR 2. (Etant donné que les coordonnées de l'écran sont de base zéro, la valeur maximale gérée par la largeur et la hauteur est 4 095.)*

Exemple

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 et versions ultérieures : facultatif

Spécifie si la fenêtre peut être réduite.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
`true` (valeur par défaut) ou `false`

Exemple

```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 et versions ultérieures : facultatif

Spécifie la taille minimale autorisée de la fenêtre.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Deux entiers qui représentent la largeur et la hauteur minimales, séparés par un espace blanc. Notez que la taille minimale imposée par le système d'exploitation prime sur la valeur définie dans le fichier descripteur d'application.

Exemple

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 et versions ultérieures : facultatif

Titre de l'application affiché par le programme d'installation d'une application AIR.

Si aucun élément `name` n'est indiqué, le programme d'installation d'une application AIR utilise l'élément `filename` à titre de nom d'application.

Élément parent :« application » à la page 222

Éléments enfants :« text » à la page 254

Contenu

Si vous indiquez un nœud texte unique (au lieu de plusieurs éléments `<text>`), le programme d'installation de l'application AIR utilise ce nom, quelle que soit la langue système activée.

Le modèle de fichier descripteur d'application AIR 1.0 ne permet de définir qu'un seul nœud texte simple en tant que nom (plutôt que plusieurs éléments `text`). Dans AIR 1.1 (ou version ultérieure), il est possible de spécifier plusieurs langues dans l'élément `name`.

L'attribut `xml:lang` de chaque élément text définit un code de langue, comme indiqué à l'adresse suivante RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

Le programme d'installation d'une application AIR utilise le nom le plus proche de la langue de l'interface utilisateur du système d'exploitation de l'utilisateur. Prenons l'exemple d'une installation dans laquelle l'élément `name` du fichier descripteur d'application comprend une valeur pour le jeu de paramètres régionaux en (anglais). Le programme d'installation d'une application AIR utilise le nom en si le système d'exploitation identifie en (anglais) en tant que langue de l'interface utilisateur. Il utilise également le nom en si la langue de l'interface utilisateur système correspond à en-US (anglais américain). Toutefois, si la langue de l'interface utilisateur correspond à en-US et que le fichier descripteur d'application définit à la fois les noms en-US et en-GB, le programme d'installation d'AIR utilise la valeur en-US. Si l'application ne définit pas de nom qui corresponde aux langues prises en charge par le système d'exploitation, le programme d'installation d'une application AIR utilise la première valeur `name` stipulée dans le fichier descripteur d'application.

L'élément `name` ne définit que le titre d'application utilisé par le programme d'installation d'une application AIR. Le programme d'installation d'une application AIR prend en charge plusieurs langues : chinois traditionnel, chinois simplifié, tchèque, néerlandais, anglais, français, allemand, italien, japonais, coréen, polonais, portugais brésilien, russe, espagnol, suédois et turc. Le programme d'installation d'une application AIR sélectionne la langue affichée (pour le texte autre que le titre et la description de l'application) en fonction de la langue de l'interface utilisateur système. La langue sélectionnée n'est pas affectée par les paramètres du fichier descripteur d'application.

L'élément `name` ne définit *pas* les jeux de paramètres régionaux disponibles pour l'application installée en cours d'exécution. Pour plus d'informations sur le développement d'applications multilingues, voir « Localisation d'applications AIR » à la page 309.

Exemple

L'exemple suivant définit un nom associé à un nœud texte simple :

```
<name>Test Application</name>
```

L'exemple suivant, valide dans AIR 1.1 et ultérieur, spécifie le nom dans trois langues (anglais, espagnol et français) par le biais de nœuds d'élément <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 et versions ultérieures : obligatoire

Identifie le nom d'un type de fichier.

Élément parent :« fileType » à la page 234

Éléments enfants : aucun

Contenu
Chaîne représentant le nom d'un type de fichier.

Exemple

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 et versions ultérieures : facultatif

Identifie l'emplacement de stockage des raccourcis associés à l'application dans le menu Tous les programmes du système d'exploitation Windows ou dans le menu Applications sous Linux. (Ce paramètre n'est actuellement pas pris en charge par les autres systèmes d'exploitation.)

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
La chaîne utilisée par la valeur `programMenuFolder` peut comporter n'importe quel caractère Unicode (UTF-8), à l'exception de ceux dont l'utilisation est interdite dans les noms de fichier dans divers systèmes de fichiers (voir l'élément `filename`, qui contient la liste d'exceptions). Le dernier caractère de cette valeur ne doit *pas* correspondre à une barre oblique (/).

Exemple

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 et versions ultérieures : facultatif

Identifie l'identifiant d'éditeur requis pour mettre à jour une application AIR initialement créée dans AIR 1.5.2 ou une version antérieure.

Ne spécifiez un identifiant d'éditeur que si vous créez une mise à jour d'application. La valeur de l'élément `publisherID` doit correspondre à l'identifiant d'éditeur généré par AIR pour la version antérieure de l'application. L'identifiant d'éditeur d'une application installée est indiqué dans le fichier `META-INF/AIR/publisherid`, qui réside dans le dossier dans lequel est installée l'application.

Les nouvelles applications créées dans AIR 1.5.3 ou ultérieur ne doivent pas spécifier d'identifiant d'éditeur.

Pour plus d'informations, voir « A propos des identifiants d'éditeur AIR » à la page 201.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
Chaîne d'identifiant d'éditeur.

Exemple
```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 et versions ultérieures : facultatif

Indique si l'accélération par unité de processeur graphique (GPU) est utilisée, sous réserve d'être prise en charge par le périphérique actif.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Utilisez l'une des valeurs suivantes :

- `auto` (valeur par défaut) : active actuellement le mode unité centrale.
- `cpu` : l'accélération matérielle n'est pas utilisée.
- `direct` : la composition du rendu est effectuée dans l'unité centrale ; le blitting fait appel au processeur graphique. Disponible dans AIR 3+.

  Remarque : *pour tirer profit de l'accélération par processeur graphique du contenu Flash sur les plates-formes AIR mobiles, Adobe recommande d'utiliser renderMode="direct" (c'est-à-dire, Stage3D) plutôt que renderMode="gpu". Adobe prend officiellement en charge les structures d'application basées sur Stage3D suivantes et recommande leur utilisation : Starling (2D) et Away3D (3D). Pour plus d'informations sur Stage3D et Starling/Away3D, voir http://gaming.adobe.com/getstarted/.*

- `gpu` : l'accélération matérielle est utilisée, le cas échéant.

  Important : *N'utilisez pas le mode de rendu sur GPU pour les applications Flex.*

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

requestedDisplayResolution

Adobe AIR 2.6 et versions ultérieures, iOS uniquement ; Adobe AIR 3.6 et versions ultérieures, OS X (facultatif)

Indique si l'application souhaite utiliser une résolution standard ou haute sur un périphérique ou un ordinateur équipé d'un écran à haute résolution. Si l'élément est défini sur *standard* (valeur par défaut), l'écran est assimilé par l'application à un écran à résolution standard. Si l'élément est défini sur *high*, l'application peut adresser chaque pixel à haute résolution.

Par exemple, sur un écran d'iPhone haute résolution 640 x 960, si le paramètre est défini sur *standard*, la dimension de la scène en plein écran est de 320 x 480, et chaque pixel d'application est rendu en utilisant quatre pixels d'écran. Si le paramètre est défini sur *high*, la dimension de la scène en plein écran est 640 x 960.

Sur un périphérique équipé d'un écran à haute résolution, les dimensions de la scène correspondent à celles de l'écran, quel que soit le paramètre utilisé.

Si l'élément `requestedDisplayResolution` est imbriqué dans l'élément `iPhone`, il s'applique aux périphériques iOS. Dans ce cas, l'attribut `excludeDevices` peut être utilisé pour spécifier les périphériques auxquels le paramètre ne s'applique pas.

Si l'élément `requestedDisplayResolution` est imbriqué dans l'élément `initialWindow`, il s'applique aux applications AIR sur les ordinateurs MacBook Pro prenant en charge des écrans haute résolution. La valeur spécifiée s'applique à toutes les fenêtres natives utilisées dans l'application. L'imbrication de l'élément `requestedDisplayResolution` dans l'élément `initialWindow` est pris en charge dans AIR 3.6 et ultérieur.

Élément parent :« iPhone » à la page 243, « initialWindow » à la page 241

Éléments enfants : aucun

Contenu

Soit *standard* (valeur par défaut), soit *high*

Attribut

excludeDevices : liste séparée par des espaces de noms de modèles iOS ou de préfixes de noms de modèles. Cela permet au développeur de faire utiliser la haute résolution par certains périphériques et la résolution standard par d'autres. Cet attribut n'est disponible que sous iOS (l'élément `requestedDisplayResolution` est imbriqué dans l'élément `iPhone`). L'attribut `excludeDevices` est disponible dans AIR 3.6 et ultérieur.

Pour tout autre périphérique dont le nom de modèle est spécifié dans cet attribut, la valeur `requestedDisplayResolution` est l'inverse de la valeur spécifiée. En d'autres termes, si la valeur de `requestedDisplayResolution` est *high*, les périphériques exclus utilisent la résolution standard. Si la valeur de `requestedDisplayResolution` est *standard*, les périphériques exclus utilisent la résolution élevée.

Les valeurs sont des noms de modèles de périphériques iOS ou des préfixes de noms de modèles. Par exemple, la valeur iPad3,1 désigne spécifiquement un iPad de 3e génération avec Wi-Fi (et non un iPad de 3e génération avec GSM ou CDMA). Quant à la valeur iPad3, elle fait référence à tout iPad de 3e génération. Une liste non officielle des noms de modèles de périphériques iOS est disponible sur la page Models du iPhone wiki (en anglais uniquement).

Exemples

Bureau :

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

iOS :

```
<iPhone>
    <requestedDisplayResolution excludeDevices="iPad3
iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 et versions ultérieures : facultatif

Spécifie si la fenêtre peut être redimensionnée.

Remarque : *sur des systèmes d'exploitation qui assimilent l'agrandissement d'une fenêtre à une opération de redimensionnement, tel Mac OS X, les paramètres maximizable et resizable doivent être définis sur* `false` *pour interdire l'exécution d'un zoom ou d'un redimensionnement de la fenêtre.*

Élément parent :« initialWindow » à la page 241

Éléments enfants :

Contenu

`true` (valeur par défaut) ou `false`

Exemple

```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 et versions ultérieures, profil mobile : facultatif

Spécifie le comportement par défaut de l'application lorsqu'un clavier virtuel est affiché. Le comportement par défaut consiste à déplacer l'application vers le haut. Le moteur d'exécution conserve à l'écran le champ de texte ou l'objet interactif qui sont la cible d'action. Utilisez l'option *pan* si l'application ne gère pas sa propre logique de traitement du clavier.

Vous pouvez également désactiver le comportement par défaut en définissant l'élément `softKeyboardBehavior` sur *none*. Dans ce cas de figure, les champs de texte et les objets interactifs distribuent un événement SoftKeyboardEvent lors de l'affichage du clavier logiciel, mais le moteur d'exécution n'effectue pas de panoramique ou de redimensionnement de l'application. C'est à l'application qu'il incombe de s'assurer que la zone de saisie de texte n'est pas masquée.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

Soit *none*, soit *pan*. La valeur par défaut est *pan*.

Exemple

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Voir aussi

SoftKeyboardEvent

supportedLanguages

Adobe AIR 3.2 et versions ultérieures : facultatif

Identifie les langues prises en charge par l'application. Cet élément est utilisé uniquement par iOS, le moteur d'exécution captif de Mac et les applications Android. Cet élément est ignoré par tous les autres types d'applications.

Si vous ne spécifiez pas cet élément, le programme de mise en package exécute par défaut les actions suivantes en fonction du type d'application :

- iOS : les langues prises en charge par le moteur d'exécution d'AIR sont répertoriées dans l'App Store d'iOS comme langues prises en charges dans l'application.
- Moteur d'exécution captif de Mac : toute application mise en package avec un paquet captif ne possède aucune information de localisation.
- Android : le paquet d'application dispose de ressources pour toutes les langues prises en charge par le moteur d'exécution d'AIR.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu

Liste de langues prises en charge, séparée par des espaces. Les valeurs de langue valides correspondent aux valeurs ISO 639-1 des langues prises en charge par le moteur d'exécution d'AIR : `en`, `de`, `es`, `fr`, `it`, `ja`, `ko`, `pt`, `ru`, `cs`, `nl`, `pl`, `sv`, `tr`, `zh`, `da`, `nb`, `iw`.

Le programme de mise en package génère une erreur si une valeur vide est spécifiée pour l'élément `<supportedLanguages>`.

Remarque : *les balises localisées (telles que le nom de la balise) ignorent la valeur d'une langue si vous utilisez la balise* `<supportedLanguages>` *alors qu'elle ne contient pas cette langue. Si une extension native dispose de ressources pour une langue non spécifiée par la balise* `<supportedLanguages>`*, un avertissement est renvoyé et les ressources sont ignorées pour cette langue.*

Exemple

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 et versions ultérieures : facultatif

Identifie les profils pris en charge pour l'application.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu

Vous pouvez inclure n'importe lesquelles des valeurs suivantes dans l'élément `supportedProfiles` :

- `desktop` : le profil de bureau est associé aux applications AIR installées sur un ordinateur de bureau à l'aide d'un fichier AIR. Ces applications n'ont pas accès à la classe NativeProcess (qui assure la communication avec les applications natives).
- `extendedDesktop` : le profil de bureau étendu est associé aux applications AIR installées sur un ordinateur de bureau à l'aide d'un programme d'installation d'application natif. Ces applications ont accès à la classe NativeProcess (qui assure la communication avec les applications natives).
- `mobileDevice` : le profil de périphérique mobile est associé aux applications mobiles.
- `extendedMobileDevice` : le profil de périphérique mobile étendu n'est pas utilisé actuellement.

La propriété `supportedProfiles` est facultative. Lorsque vous l'omettez du fichier descripteur d'application, il est possible de compiler et de déployer l'application pour tout profil.

Pour spécifier plusieurs profils, séparez-les par un espace. Le paramètre suivant, par exemple, stipule que l'application est uniquement disponible dans les profils de bureau et les profils étendus :

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

Remarque : *si vous exécutez une application avec l'application ADL et que vous ne spécifiez pas de valeur pour l'option ADL* `-profile`*, le premier profil du fichier descripteur d'application est utilisé. (Si aucun profil n'est spécifié dans le fichier descripteur d'application, le profil de bureau est utilisé.)*

Exemple

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

Voir aussi

« Profils de périphérique » à la page 259

« Profils pris en charge » à la page 78

systemChrome

Adobe AIR 1.0 et versions ultérieures : facultatif

Indique si la fenêtre de l'application initiale contient la barre de titre, les bordures et les contrôles standard proposés par le système d'exploitation.

Il est impossible de modifier le paramètre systemChrome de la fenêtre à l'exécution.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

Utilisez l'une des valeurs suivantes :

- `none` : aucun chrome système n'est proposé. L'application (ou une structure d'application telle que Flex) est responsable de l'affichage du chrome de fenêtre.
- `standard` (valeur par défaut) : le chrome système est proposé par le système d'exploitation.

Exemple

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 et versions ultérieures : facultatif

Spécifie une chaîne localisée.

L'attribut `xml:lang` d'un élément text définit un code de langue, comme indiqué à l'adresse suivante RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

Le programme d'installation d'une application AIR utilise l'élément `text` associé à la valeur de l'attribut `xml:lang` la plus proche de la langue de l'interface utilisateur du système d'exploitation de l'utilisateur.

Prenons l'exemple d'une installation dans laquelle l'élément `text` comprend une valeur pour le jeu de paramètres régionaux en (anglais). Le programme d'installation d'une application AIR utilise le nom en si le système d'exploitation identifie en (anglais) en tant que langue de l'interface utilisateur. Il utilise également le nom en si la langue de l'interface utilisateur système correspond à en-US (anglais américain). Toutefois, si la langue de l'interface utilisateur correspond à en-US et que le fichier descripteur d'application définit à la fois les noms en-US et en-GB, le programme d'installation d'AIR utilise la valeur en-US.

Si l'application ne définit pas d'élément `text` qui corresponde aux langues prises en charge par le système d'exploitation, le programme d'installation d'une application AIR utilise la première valeur `name` stipulée dans le fichier descripteur d'application.

Éléments parents :

- « name » à la page 247
- « description » à la page 230

Éléments enfants : aucun

Contenu

Attribut `xml:lang` qui spécifie des paramètres régionaux et une chaîne de texte localisé.

Exemple

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 et versions ultérieures : facultatif

Indique le titre figurant dans la barre de titre de la fenêtre initiale de l'application.

Le titre ne s'affiche que si l'élément `systemChrome` est défini sur `standard`.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

Chaîne contenant le titre de la fenêtre.

Exemple

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 et versions ultérieures : facultatif

Indique si la fenêtre initiale de l'application est semi-transparente par rapport au bureau.

Une fenêtre transparente risque d'être tracée plus lentement et de mobiliser plus de mémoire. Il est impossible de modifier le paramètre transparent à l'exécution.

***Important :** vous ne pouvez définir `transparent` sur `true` que si `systemChrome` est défini sur `none`.*

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

`true` ou `false` (valeur par défaut)

Exemple

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

version

Adobe AIR 1.0 à 2.0 : obligatoire, élément interdit dans AIR 2.5 et versions ultérieures

Identifie la version de l'application.

La chaîne version est un indicateur défini par l'application. AIR n'interprète pas la chaîne de version de quelque façon que ce soit. De ce fait, la version « 3.0 » n'est pas considérée comme étant plus récente que la version « 2.0 ». Exemples : « `1.0` », « `.4` », « `0.5` », « `4.9` », « `1.3.4a` ».

Dans AIR 2.5 et ultérieur, l'élément `version` a été remplacé par les éléments `versionNumber` et `versionLabel`.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu

Chaîne contenant la version de l'application.

Exemple

```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 et versions ultérieures : facultatif

Spécifie une chaîne indiquant la version intelligible.

La valeur de l'élément versionLabel, plutôt que celle de l'élément `versionNumber`, est affichée dans les boîtes de dialogue d'installation. Si vous n'utilisez pas l'élément `versionLabel`, `versionNumber` est utilisé dans les deux cas.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
Chaîne contenant la version affichée publiquement.

Exemple
```
<versionLabel>0.9 Beta</versionlabel>
```

versionNumber

Adobe AIR 2.5 et versions ultérieures : obligatoire

Numéro de version de l'application.

Élément parent :« application » à la page 222

Éléments enfants : aucun

Contenu
Le numéro de version peut contenir une séquence de trois entiers au plus, séparés par un point. Chaque entier est un nombre compris entre 0 et 999 (inclus).

Exemples
```
<versionNumber>1.0.657</versionNumber>

<versionNumber>10</versionNumber>

<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 et versions ultérieures : facultatif

Indique si la fenêtre initiale de l'application est visible dès sa création.

Les fenêtres AIR, y compris la fenêtre initiale, sont par défaut invisibles lors de leur création. Pour afficher une fenêtre, appelez la méthode `activate()` de l'objet NativeWindow ou définissez la propriété `visible` sur `true`. Il peut s'avérer utile de masquer initialement la fenêtre principale, afin de ne pas afficher les modifications apportées à la position, à la taille ou à la disposition du contenu de la fenêtre.

Le composant Flex `mx:WindowedApplication` affiche et active automatiquement la fenêtre juste avant la distribution de l'événement `applicationComplete`, à moins que l'attribut `visible` ne soit défini sur `false` dans la définition MXML.

Sur les périphériques indiqués dans le profil mobile, qui ne prend pas en charge les fenêtres, le paramètre visible est ignoré.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
`true` ou `false` (valeur par défaut)

Exemple

```
<visible>true</visible>
```

width

Adobe AIR 1.0 et versions ultérieures : facultatif

Largeur initiale de la fenêtre principale de l'application.

Si vous ne définissez pas de largeur, elle est déterminée par les paramètres figurant dans le fichier SWF racine ou, dans le cas d'une application AIR de type HTML, par le système d'exploitation.

La largeur maximale d'une fenêtre est passée de 2 048 à 4 096 pixels dans AIR 2.

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

Entier positif dont la valeur ne doit pas dépasser 4 095.

Exemple

```
<width>1024</width>
```

x

Adobe AIR 1.0 et versions ultérieures : facultatif

Position horizontale de la fenêtre initiale d'une application.

Dans la plupart des cas, il est préférable de confier au système d'exploitation le soin de déterminer la position initiale de la fenêtre, plutôt que de définir une valeur fixe.

L'origine du système de coordonnées de l'écran (0,0) correspond à l'angle supérieur gauche de l'écran principal de l'ordinateur de bureau (déterminé par le système d'exploitation).

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu

Valeur entière

Exemple

```
<x>120</x>
```

y

Adobe AIR 1.0 et versions ultérieures : facultatif

Position verticale de la fenêtre initiale d'une application.

Dans la plupart des cas, il est préférable de confier au système d'exploitation le soin de déterminer la position initiale de la fenêtre, plutôt que de définir une valeur fixe.

L'origine du système de coordonnées de l'écran (0,0) correspond à l'angle supérieur gauche de l'écran principal de l'ordinateur de bureau (déterminé par le système d'exploitation).

Élément parent :« initialWindow » à la page 241

Éléments enfants : aucun

Contenu
Valeur entière

Exemple

```
<y>250</y>
```

Chapitre 15 : Profils de périphérique

Adobe AIR 2 et ultérieur

Les profils sont un mécanisme de définition des classes des périphériques informatiques sur lesquels s'exécute l'application. Un profil définit un ensemble d'API et de fonctionnalités généralement pris en charge par une classe déterminée de périphérique. Les profils disponibles sont les suivants :

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Vous pouvez définir les profils associés à l'application dans le fichier descripteur de l'application. Seuls les utilisateurs qui disposent d'un ordinateur ou d'un périphérique stipulé dans les profils inclus peuvent installer l'application. Si vous n'incluez, par exemple, que le profil desktop dans le fichier descripteur de l'application, les utilisateurs peuvent installer et exécuter l'application sur un ordinateur de bureau uniquement.

Si vous incluez un profil que l'application ne prend pas totalement en charge, l'expérience de l'utilisateur dans les environnements de ce type risque de ne pas être satisfaisante. Si vous ne spécifiez pas de profil dans le fichier descripteur de l'application, AIR ne restreint pas cette dernière. Vous pouvez mettre en package l'application dans tous les formats pris en charge et celle-ci peut être installée par les utilisateurs qui disposent d'un périphérique stipulé dans n'importe quel profil. Elle risque toutefois de ne pas fonctionner correctement à l'exécution.

Dans la mesure du possible, les restrictions associées aux profils sont imposées lorsque vous mettez en package l'application. Si, par exemple, vous incluez le profil extendedDesktop uniquement, il est impossible de mettre en package l'application en tant que fichier AIR, mais uniquement en tant que programme d'installation natif. De même, si vous n'incluez pas le profil mobileDevice, il est impossible de mettre en package l'application en tant que fichier APK Android.

Un périphérique informatique peut prendre en charge plusieurs profils. Sur un ordinateur de bureau, AIR prend, par exemple, en charge les applications associées aux profils desktop et extendedDesktop. Une application associée au profil extendedDesktop peut toutefois communiquer avec les processus natifs et vous devez IMPERATIVEMENT la mettre en package en tant que programme d'installation natif (exe, dmg, deb ou rpm). En revanche, une application associée au profil desktop ne peut pas communiquer avec un processus natif. Vous pouvez la mettre en package en tant que fichier AIR ou programme d'installation natif.

L'inclusion d'une fonctionnalité dans un profil indique qu'elle est généralement prise en charge par la classe de périphérique correspondante. Ce qui ne signifie pas que chaque périphérique associé à un profil prend en charge chaque fonctionnalité. La plupart des téléphones portables, mais non leur totalité, contiennent par exemple un accéléromètre. Les classes et fonctionnalité qui ne sont pas universellement prises en charge gèrent généralement une propriété booléenne que vous pouvez vérifier avant d'utiliser la fonctionnalité. Dans le cas d'un accéléromètre, par exemple, vous pouvez tester la propriété statique `Accelerometer.isSupported` pour déterminer si le périphérique actif en est équipé.

Vous pouvez assigner les profils suivants à l'application AIR à l'aide de l'élément supportedProfiles du fichier descripteur de l'application :

Bureau Le profil de bureau définit un ensemble de fonctionnalités réservé aux applications AIR installées en tant que fichiers AIR sur un ordinateur de bureau. Ces applications sont installées et s'exécutent sur les plates-formes de bureau prises en charge (Mac OS, Windows et Linux). Les applications AIR développées dans les versions d'AIR antérieures

à la version 2 peuvent être considérées comme faisant partie du profil de bureau. Certaines API ne sont pas prises en charge dans ce profil. Ainsi, les applications de bureau ne peuvent pas communiquer avec les processus natifs.

Bureau étendu Le profil de bureau étendu définit un ensemble de fonctionnalités réservé aux applications AIR qui sont mises en package avec un programme d'installation natif et installées par le biais de celui-ci. Ces programmes d'installation natifs sont des fichiers EXE sous Windows, DMG sous Mac OS et BIN, DEB ou RPM sous Linux. Les applications du profil de bureau étendu sont dotées de fonctionnalités que ne possèdent pas les applications du profil de bureau. Pour plus d'informations, voir « Mise en package d'un programme d'installation natif de bureau » à la page 58.

Périphérique mobile Le profil de périphérique mobile définit un ensemble de fonctionnalités réservé aux applications installées sur des périphériques mobiles (tels que téléphones portables et tablettes). Ces applications s'installent et s'exécutent sur les plates-formes mobiles prises en charge, notamment sur Android, Blackberry Tablet OS et iOS.

Périphérique mobile étendu Le profil de périphérique mobile étendu définit un ensemble de fonctionnalités étendu réservé aux applications installées sur des périphériques mobiles. Aucun périphérique ne prend actuellement en charge ce profil.

Limitation des profils cible dans le fichier descripteur de l'application

Adobe AIR 2 et ultérieur

Depuis AIR 2, le fichier descripteur de l'application contient l'élément `supportedProfiles`, qui permet de limiter les profils cible. Le paramètre suivant, par exemple, stipule que l'application est uniquement disponible dans le profil de bureau :

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

Lorsque cet élément est défini, l'application peut uniquement être mise en package dans les profils que vous indiquez. Les valeurs disponibles sont les suivantes :

- `desktop` : profil de bureau
- `extendedDesktop` : profil de bureau étendu
- `mobileDevice` : profil de périphérique mobile

L'élément `supportedProfiles` est facultatif. Lorsque vous l'omettez du fichier descripteur d'application, il est possible de mettre en package et de déployer l'application pour tout profil.

Pour spécifier plusieurs profils dans l'élément `supportedProfiles`, séparez-les par un espace, comme suit :

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

Fonctionnalités des différents profils

Adobe AIR 2 et ultérieur

Le tableau ci-dessous indique les classes et fonctionnalités qui ne sont pas prises en charge par tous les profils.

Classe ou fonctionnalité	desktop	extendedDesktop	mobileDevice
Accéléromètre (Accelerometer.isSupported)	Non	Non	Vérifier
Accessibilité (Capabilities.hasAccessibility)	Oui	Oui	Non
Annulation de l'écho acoustique (Microphone.getEnhancedMicrophone())	Oui	Oui	Non
ActionScript 2	Oui	Oui	Non
Matrice CacheAsBitmap	Non	Non	Oui
Caméra (Camera.isSupported)	Oui	Oui	Oui
CameraRoll	Non	Non	Oui
CameraUI (CameraUI.isSupported)	Non	Non	Oui
Paquets de moteur d'exécution	Oui	Oui	Oui
ContextMenu (ContextMenu.isSupported)	Oui	Oui	Non
DatagramSocket (DatagramSocket.isSupported)	Oui	Oui	Oui
DockIcon (NativeApplication.supportsDockIcon)	Vérifier	Vérifier	Non
Glisser-déposer (NativeDragManager.isSupported)	Oui	Oui	Vérifier
EncyptedLocalStore (EncyptedLocalStore.isSupported)	Oui	Oui	Oui
Accès à Flash (DRMManager.isSupported)	Oui	Oui	Non
GameInput (GameInput.isSupported)	Non	Non	Non
Géolocalisation (Geolocation.isSupported)	Non	Non	Vérifier
HTMLLoader (HTMLLoader.isSupported)	Oui	Oui	Non
Editeur IME (IME.isSupported)	Oui	Oui	Vérifier
LocalConnection (LocalConnection.isSupported)	Oui	Oui	Non
Microphone (Microphone.isSupported)	Oui	Oui	Vérifier
Son multicanaux (Capabilities.hasMultiChannelAudio())	Non	Non	Non
Extensions natives	Non	Oui	Oui
NativeMenu (NativeMenu.isSupported)	Oui	Oui	Non
NativeProcess (NativeProcess.isSupported)	Non	Oui	Non
NativeWindow (NativeWindow.isSupported)	Oui	Oui	Non
NetworkInfo (NetworkInfo.isSupported)	Oui	Oui	Vérifier
Ouverture de fichiers dans l'application par défaut	Prise en charge restreinte	Oui	Non

Classe ou fonctionnalité	desktop	extendedDesktop	mobileDevice
PrintJob (PrintJob.isSupported)	Oui	Oui	Non
SecureSocket (SecureSocket.isSupported)	Oui	Oui	Vérifier
ServerSocket (ServerSocket.isSupported)	Oui	Oui	Oui
Shader	Oui	Oui	Prise en charge restreinte
Stage3D (Stage.stage3Ds.length)	Oui	Oui	Oui
Orientation de la scène (Stage.supportsOrientationChange)	Non	Non	Oui
StageVideo	Non	Non	Vérifier
StageWebView (StageWebView.isSupported)	Oui	Oui	Oui
Lancement de l'application lors de l'établissement de la connexion (NativeApplication.supportsStartAtLogin)	Oui	Oui	Non
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Oui	Oui	Non
Mode inactif du système	Non	Non	Oui
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Vérifier	Vérifier	Non
Entrée Text Layout Framework	Oui	Oui	Non
Updater (Updater.isSupported)	Oui	Non	Non
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Oui	Oui	Non

La signification des entrées du tableau est la suivante :

- *Vérifier* : la fonctionnalité est prise en charge sur certains périphériques du profil, mais pas tous. Vérifiez à l'exécution si elle est prise en charge avant de l'utiliser.
- *Prise en charge restreinte* : la fonctionnalité est prise en charge, mais est sujette à d'importantes restrictions. Pour plus d'informations, voir la documentation appropriée.
- *Non* : la fonctionnalité n'est pas prise en charge par le profil.
- *Oui* : la fonctionnalité est prise en charge par le profil. Notez que certains périphériques ne sont pas équipés du matériel requis par une fonctionnalité. Tous les téléphones ne disposent par exemple pas d'une caméra.

Spécification de profils lors du débogage à l'aide de l'application de débogage du lanceur AIR (ADL)

Adobe AIR 2 et ultérieur

L'application de débogage du lanceur AIR (ADL) vérifie si vous avez spécifié les profils pris en charge dans l'élément `supportedProfiles` du fichier descripteur d'application. Si tel est le cas, l'application ADL utilise par défaut le premier profil pris en charge de la liste lors du débogage.

Vous pouvez spécifier le profil que doit utiliser l'outil ADL à l'aide de l'argument de ligne de commande `-profile` (Voir « Application de débogage du lanceur AIR (ADL) » à la page 168.) Vous pouvez utiliser cet argument même si vous ne spécifiez pas de profil dans l'élément `supportedProfiles` du fichier descripteur d'application. Toutefois, si vous définissez l'élément `supportedProfiles`, il doit inclure le profil indiqué dans la ligne de commande, sans quoi l'outil ADL génère une erreur.

Chapitre 16 : API intégrée au navigateur et stockée dans le fichier AIR.SWF

Personnalisation du fichier badge.swf de l'installation transparente

Outre l'utilisation du fichier badge.swf fourni dans le SDK (kit de développement logiciel), vous avez également la possibilité de créer votre propre fichier SWF pour l'utiliser dans une page de navigateur. Le fichier SWF que vous avez personnalisé peut interagir avec le moteur d'exécution de différentes façons :

- Il peut installer une application AIR. Voir « Installation d'une application AIR à partir du navigateur » à la page 270.
- Il peut vérifier si une application AIR particulière est installée. Voir « Vérification à partir d'une page Web de la présence d'une application AIR installée » à la page 269.
- Il peut vérifier si le moteur d'exécution est installé. Voir « Vérification de la présence du moteur d'exécution » à la page 268.
- Il peut lancer une application AIR installée sur le système d'un utilisateur. Voir « Lancement d'une application AIR installée à partir du navigateur » à la page 271.

Ces possibilités sont toutes fournies en appelant des API dans un fichier SWF, nommé air.swf, et hébergé sur adobe.com. air.swf. Vous pouvez personnaliser le fichier badge.swf et appeler les API air.swf à partir de votre propre fichier SWF.

En outre, un fichier SWF s'exécutant dans le navigateur peut communiquer avec une application AIR en cours d'exécution grâce à l'utilisation de la classe LocalConnection. Pour plus d'informations, voir Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs ActionScript) ou Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs HTML).

Important : *les fonctionnalités décrites dans cette section (et les API du fichier air.swf) impliquent l'installation préalable d'Adobe® Flash® Player 9 mise à jour 3 ou ultérieur par l'utilisateur final dans le navigateur Web sous Windows ou Mac OS. Sous Linux, la fonction d'installation transparente requiert Flash Player 10 (version 10,0,12,36 ou ultérieure). Vous pouvez écrire du code pour vérifier la version installée de Flash Player et fournir une autre interface à l'utilisateur si la version requise de Flash Player n'est pas installée. Par exemple, si une ancienne version de Flash Player est installée, il est possible de fournir un lien vers la version à télécharger du fichier AIR (plutôt que d'utiliser le fichier badge.swf ou l'API air.swf pour installer une application).*

Utilisation du fichier badge.swf pour installer une application AIR

Le fichier badge.swf, qui est intégré aux kits SDK AIR et Flex, facilite l'utilisation de la fonctionnalité d'installation transparente. Ce fichier peut installer le moteur d'exécution et une application AIR à partir d'un lien dans une page Web. Le fichier badge.swf et son code source vous sont fournis pour que vous en assuriez la distribution sur votre site Web.

Intégration du fichier badge.swf dans une page Web

1 Recherchez les fichiers suivants, qui résident dans le répertoire samples/badge du kit SDK AIR ou Flex, et ajoutez-les au serveur Web.

- badge.swf
- default_badge.html
- AC_RunActiveContent.js

2 Ouvrez la page default_badge.html dans un éditeur de texte.

3 Dans la page default_badge.html, au niveau de la fonction JavaScript `AC_FL_RunContent()`, ajustez le paramétrage `FlashVars` pour les paramètres suivants :

Paramètre	Description
`appname`	Nom de l'application, affiché par le fichier SWF lorsque le moteur d'exécution n'est pas installé.
`appurl`	(Obligatoire). URL du fichier AIR à télécharger. Vous devez utiliser une URL absolue, et non relative.
`airversion`	(Obligatoire). Pour la version 1.0 du moteur d'exécution, définissez ce paramètre sur 1.0.
`imageurl`	URL de l'image (facultative) à afficher dans le badge.
`buttoncolor`	Couleur du bouton Télécharger (spécifiée sous forme de valeur hexadécimale, telle que `FFCC00`).
`messagecolor`	Couleur du message textuel affiché sous le bouton lorsque le moteur d'exécution n'est pas installé (spécifiée sous la forme d'une valeur hexadécimale, telle que `FFCC00`).

4 La taille minimale du fichier badge.swf est de 217 pixels de large sur 180 de haut. Ajustez les valeurs des paramètres `width` et `height` de la fonction `AC_FL_RunContent()` à votre convenance.

5 Renommez le fichier default_badge.html et ajustez son code (ou englobez-le dans une autre page HTML) selon vos besoins.

Remarque : *ne définissez pas l'attribut `wmode` de la balise HTML `embed` qui charge le fichier badge.swf ; conservez la valeur par défaut (`"window"`). Les autres paramètres `wmode` empêchent l'installation sur certains systèmes et leur utilisation donne lieu à une erreur : « Erreur n° 2044 : ErrorEvent non traité :. text=Erreur n° 2074 : La scène est trop petite pour contenir l'interface utilisateur téléchargée. »*

Vous pouvez également modifier et recompiler le fichier badge.swf. Pour plus d'informations, voir « Modification du fichier badge.swf » à la page 266.

Installation de l'application AIR à partir d'un lien d'installation transparente proposé dans une page Web

Une fois que vous avez ajouté le lien de l'installation transparente à une page, l'utilisateur peut installer l'application AIR en cliquant sur le lien dans le fichier SWF.

1 Naviguez jusqu'à la page HTML dans un navigateur Web sur lequel Flash Player (version 9 mise à jour 3 ou ultérieure sous Windows et Mac OS, ou version 10 sous Linux) est installé.

2 Dans cette page Web, cliquez sur le lien contenu dans le fichier badge.swf.

- Si le moteur d'exécution est installé, passez à l'étape suivante.
- Si le moteur d'exécution n'est pas installé, une boîte de dialogue s'affiche pour vous proposer de l'installer. Installez le moteur d'exécution (voir « Installation d'Adobe AIR » à la page 3), puis passez à l'étape suivante.

3 Dans la fenêtre d'installation, conservez les paramètres par défaut sélectionnés, puis cliquez sur Continuer.

Sous Windows, l'environnement d'exécution AIR effectue les opérations suivantes :

- Installation de l'application dans c:\Program Files\
- Création d'un raccourci sur le bureau pour ouvrir l'application
- Création d'un raccourci dans le menu Démarrer
- Ajout d'une entrée dans Ajout/Suppression de programmes du Panneau de configuration

Sous Mac OS, le programme d'installation ajoute l'application au répertoire Applications (par exemple, dans le répertoire /Applications pour Mac OS).

Sur un ordinateur Linux, AIR effectue automatiquement les opérations suivantes :

- Installation de l'application dans /opt
- Création d'un raccourci sur le bureau pour ouvrir l'application
- Création d'un raccourci dans le menu Démarrer
- Ajout d'une entrée dans le gestionnaire de package du système

4 Sélectionnez les options qui vous intéressent, puis cliquez sur le bouton Installer.

5 Une fois que vous avez terminé l'installation, cliquez sur Terminer.

Modification du fichier badge.swf

Les kits SDK de Flex et d'AIR contiennent les fichiers sources associés au fichier badge.swf. Ces fichiers sont situés dans le dossier samples/badge du kit SDK :

Fichiers sources	Description
badge.fla	Fichier Flash source, utilisé pour compiler le fichier badge.swf. Le fichier badge.fla est compilé dans un fichier SWF 9 (pouvant être chargé dans Flash Player).
AIRBadge.as	Classe ActionScript 3.0 définissant la classe de base utilisée dans le fichier basdge.fla.

Vous pouvez utiliser Flash Professional pour modifier la conception de l'interface visuelle du fichier badge.fla.

La fonction constructeur `AIRBadge()`, définie dans la classe AIRBadge, charge le fichier air.swf hébergé à l'adresse suivante http://airdownload.adobe.com/air/browserapi/air.swf. Le fichier air.swf contient le code pour l'utilisation de la fonctionnalité d'installation transparente.

La méthode `onInit()` (dans la classe AIRBadge) est appelée lorsque le chargement du fichier air.swf s'est correctement déroulé :

```
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;
    }
}
```

Le code définit la variable `_air` globale sur la classe principale du fichier air.swf chargé. Cette classe comprend les méthodes publiques suivantes auxquelles le fichier badge.swf accède pour appeler la fonctionnalité d'installation transparente :

Méthode	**Description**
`getStatus()`	Détermine si le moteur d'exécution est installé (ou s'il peut être installé) sur l'ordinateur. Pour plus d'informations, voir « Vérification de la présence du moteur d'exécution » à la page 268. • `runtimeVersion` : chaîne précisant la version du moteur d'exécution (`« 1.0.M6 »`, par exemple) qui est requise par l'application à installer.
`installApplication()`	Installe l'application spécifiée sur l'ordinateur de l'utilisateur. Pour plus d'informations, voir « Installation d'une application AIR à partir du navigateur » à la page 270. • `url` : chaîne définissant l'URL. Vous devez utiliser un chemin d'URL absolu, et non relatif. • `runtimeVersion` : chaîne précisant la version du moteur d'exécution (`« 2.5. »`, par exemple) qui est requise par l'application à installer. • `arguments` : arguments à transmettre à l'application si elle est lancée après son installation. L'application est lancée à l'installation si l'élément `allowBrowserInvocation` est défini sur `true` dans le fichier descripteur de l'application. (Pour plus d'informations sur le fichier descripteur d'application, voir « Fichiers descripteurs d'applications AIR » à la page 217.) Si l'application est lancée lors d'une installation transparente depuis le navigateur (l'utilisateur ayant choisi le lancement à l'installation), l'objet NativeApplication de l'application ne distribue d'objet BrowserInvokeEvent que si des arguments sont transmis. Tenez compte des conséquences au niveau de la sécurité qu'une transmission de données à l'application peut entraîner. Pour plus d'informations, voir « Lancement d'une application AIR installée à partir du navigateur » à la page 271.

Les paramètres pour `url` et `runtimeVersion` sont transmis dans le fichier SWF via les paramètres FlashVars dans la page HTML servant de conteneur.

Si l'application démarre automatiquement à l'installation, vous pouvez utiliser la communication LocalConnection pour que l'application installée contacte le fichier badge.swf au moment de l'appel. Pour plus d'informations, voir Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs ActionScript) ou Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs HTML).

Vous pouvez également appeler la méthode `getApplicationVersion()` du fichier air.swf pour vérifier qu'une application est déjà installée. Vous avez le choix d'appeler cette méthode avant que la procédure d'installation de l'application débute, ou après le démarrage de l'installation. Pour plus d'informations, voir « Vérification à partir d'une page Web de la présence d'une application AIR installée » à la page 269.

Chargement du fichier air.swf

Vous pouvez créer votre propre fichier SWF, il sert à utiliser les API dans le fichier air.swf pour interagir avec le moteur d'exécution et les applications AIR depuis une page Web affichée dans un navigateur. Le fichier air.swf est hébergé à l'adresse suivante http://airdownload.adobe.com/air/browserapi/air.swf. Pour référencer les API du fichier air.swf à partir de votre fichier SWF, chargez le fichier air.swf dans le même domaine d'application que celui de votre fichier SWF. Le code suivant illustre un exemple de chargement du fichier air.swf dans le domaine d'application du fichier SWF chargeant.

```
 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;
}
```

Une fois le fichier air.swf chargé (lorsque l'objet `contentLoaderInfo` de l'objet chargeur Loader distribue l'événement `init`), vous pouvez appeler n'importe quelle API du fichier air.swf, comme indiqué ci-après.

Remarque : *le fichier badge.swf, qui est intégré aux kits SDK AIR et Flex, charge automatiquement le fichier air.swf. Voir « Utilisation du fichier badge.swf pour installer une application AIR » à la page 264. Les instructions détaillées dans cette section se rapportent à la création de votre propre fichier SWF qui charge le fichier air.swf.*

Vérification de la présence du moteur d'exécution

Un fichier SWF peut vérifier si le moteur d'exécution est installé en appelant la méthode `getStatus()` dans le fichier air.swf chargé à partir de http://airdownload.adobe.com/air/browserapi/air.swf. Pour plus d'informations, voir « Chargement du fichier air.swf » à la page 268.

Une fois le fichier air.swf chargé, le fichier SWF peut appeler la méthode `getStatus()` du fichier air.swf, comme suit :

```
 var status:String = airSWF.getStatus();
```

La méthode `getStatus()` renvoie une des valeurs de chaîne suivantes en fonction de l'état du moteur d'exécution sur l'ordinateur :

Valeur de chaîne	Description
`"available"`	Il est possible d'installer le moteur d'exécution sur cet ordinateur, mais il n'y est pas installé actuellement.
`"unavailable"`	Il est impossible d'installer le moteur d'exécution sur cet ordinateur.
`"installed"`	Le moteur d'exécution est installé sur cet ordinateur.

La méthode `getStatus()` renvoie une erreur si la version requise de Flash Player (version 9 mise à jour 3 ou ultérieure sous Windows et Mac OS, ou version 10 sous Linux) n'est pas installée dans le navigateur.

Vérification à partir d'une page Web de la présence d'une application AIR installée

Un fichier SWF peut vérifier si une application AIR (avec un ID d'application et un ID d'éditeur correspondants) est installée en appelant la méthode `getApplicationVersion()` dans le fichier air.swf chargé à partir de http://airdownload.adobe.com/air/browserapi/air.swf. Pour plus d'informations, voir « Chargement du fichier air.swf » à la page 268.

Une fois le fichier air.swf chargé, le fichier SWF peut appeler la méthode `getApplicationVersion()` du fichier air.swf comme suit :

```
 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.
    }
}
```

La méthode `getApplicationVersion()` possède les paramètres suivants :

Paramètres	Description
`appID`	ID d'application pour l'application. Pour plus d'informations, voir « id » à la page 239.
`pubID`	ID d'éditeur de l'application. Pour plus d'informations, voir « publisherID » à la page 248. Si l'application concernée ne possède pas d'identifiant d'éditeur, définissez le paramètre `pubID` sur une chaîne vide ("").
`callback`	Fonction de rappel pour servir comme fonction de gestionnaire. La méthode getApplicationVersion() fonctionne de façon asynchrone et c'est lorsque la présence (ou l'absence) de la version est détectée que cette méthode de rappel est invoquée. La définition de la méthode de rappel doit inclure un paramètre, une chaîne, qui est défini sur la chaîne de version de l'application installée. Si l'application n'est pas installée, une valeur « null » est transmise à la fonction, comme illustré dans l'exemple de code précédent.

La méthode `getApplicationVersion()` renvoie une erreur si la version requise de Flash Player (version 9 mise à jour 3 ou ultérieure sous Windows et Mac OS, ou version 10 sous Linux) n'est pas installée dans le navigateur.

Remarque : *depuis la version 1.5.3 d'AIR, l'utilisation de l'identifiant d'éditeur est déconseillée. Les identifiants d'éditeur ne sont plus automatiquement affectés aux applications. Les applications peuvent néanmoins continuer à les stipuler à des fins de compatibilité ascendante.*

Installation d'une application AIR à partir du navigateur

Un fichier SWF peut installer une application AIR en appelant la méthode `installApplication()` dans le fichier air.swf chargé à partir de `http://airdownload.adobe.com/air/browserapi/air.swf`. Pour plus d'informations, voir « Chargement du fichier air.swf » à la page 268.

Une fois le fichier air.swf chargé, le fichier SWF peut appeler la méthode `installApplication()` du fichier air.swf comme suit :

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

La méthode `installApplication()` installe l'application spécifiée sur l'ordinateur de l'utilisateur. Cette méthode est dotée des paramètres suivants :

Paramètre	Description
`url`	Chaîne définissant l'URL du fichier AIR à installer. Vous devez utiliser un chemin d'URL absolu, et non relatif.
`runtimeVersion`	Chaîne précisant la version du moteur d'exécution (par exemple « 1.0 ») qui est requise par l'application à installer.
`arguments`	Tableau d'arguments à transmettre à l'application si elle est lancée au moment de l'installation. Seuls les caractères alphanumériques sont reconnus dans les arguments. Pour transmettre d'autres valeurs, pensez à utiliser un schéma de codage. L'application est lancée à l'installation si l'élément `allowBrowserInvocation` est défini sur `true` dans le fichier descripteur de l'application. (Pour plus d'informations sur le fichier descripteur d'application, voir « Fichiers descripteurs d'applications AIR » à la page 217.) Si l'application est lancée lors d'une installation transparente depuis le navigateur (l'utilisateur ayant choisi le lancement à l'installation), l'objet NativeApplication de l'application ne distribue d'objet BrowserInvokeEvent que si des arguments ont été transmis. Pour plus d'informations, voir « Lancement d'une application AIR installée à partir du navigateur » à la page 271.

La méthode `installApplication()` ne peut fonctionner que lorsqu'elle est appelée dans le gestionnaire d'événement pour un événement utilisateur, tel qu'un clic de souris.

La méthode `installApplication()` renvoie une erreur si la version requise de Flash Player (version 9 mise à jour 3 ou ultérieure sous Windows et Mac OS, ou version 10 sous Linux) n'est pas installée dans le navigateur.

Sous Mac OS, l'utilisateur doit détenir les droits système appropriés lui permettant d'installer une version de mise à jour d'une application dans le répertoire de cette application (et de privilèges d'administration si l'application met à jour le moteur d'exécution). S'il utilise Windows, l'utilisateur doit détenir des privilèges d'administration.

Vous pouvez également appeler la méthode `getApplicationVersion()` du fichier air.swf pour vérifier si une application est déjà installée. Vous avez le choix d'appeler cette méthode avant que la procédure d'installation de l'application débute, ou après le démarrage de l'installation. Pour plus d'informations, voir « Vérification à partir d'une page Web de la présence d'une application AIR installée » à la page 269. Dès qu'elle est exécutée, l'application peut communiquer avec le contenu SWF dans le navigateur en utilisant la classe LocalConnection. Pour plus d'informations, voir Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs ActionScript) ou Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs HTML).

Lancement d'une application AIR installée à partir du navigateur

Pour utiliser la fonctionnalité d'appel du navigateur (lui permettant d'être lancé à partir du navigateur), le fichier descripteur d'application de l'application cible doit comporter le paramètre suivant :

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Pour plus d'informations sur le fichier descripteur d'application, voir « Fichiers descripteurs d'applications AIR » à la page 217.

Un fichier SWF présent dans le navigateur peut lancer une application AIR en appelant la méthode `launchApplication()` dans le fichier air.swf chargé à partir de http://airdownload.adobe.com/air/browserapi/air.swf. Pour plus d'informations, voir « Chargement du fichier air.swf » à la page 268.

Une fois le fichier air.swf chargé, le fichier SWF peut appeler la méthode `launchApplication()` du fichier air.swf comme suit :

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

La méthode `launchApplication()` est définie au niveau supérieur du fichier air.swf (qui est chargé dans le domaine d'application du fichier SWF de l'interface utilisateur). Le fait d'appeler cette méthode déclenche le lancement de l'application spécifiée par AIR (si cette application est installée et si l'appel du navigateur est autorisé, par le biais du paramètre `allowBrowserInvocation` dans le fichier descripteur de l'application). Cette méthode prend en charge les paramètres suivants :

Paramètre	Description
`appID`	ID d'application pour l'application à lancer. Pour plus d'informations, voir « id » à la page 239.
`pubID`	ID d'éditeur de l'application à lancer. Pour plus d'informations, voir « publisherID » à la page 248. Si l'application concernée ne possède pas d'identifiant d'éditeur, définissez le paramètre `pubID` sur une chaîne vide ("").
`arguments`	Tableau d'arguments à transmettre à l'application. L'objet NativeApplication de l'application distribue un événement BrowserInvokeEvent dont une propriété arguments est définie sur ce tableau. Seuls les caractères alphanumériques sont reconnus dans les arguments. Pour transmettre d'autres valeurs, pensez à utiliser un schéma de codage.

La méthode `launchApplication()` ne peut fonctionner que lorsqu'elle est appelée dans le gestionnaire d'événement pour un événement utilisateur, tel qu'un clic de souris.

La méthode `launchApplication()` renvoie une erreur si la version requise de Flash Player (version 9 mise à jour 3 ou ultérieure sous Windows et Mac OS, ou version 10 sous Linux) n'est pas installée dans le navigateur.

Si l'élément `allowBrowserInvocation` est défini sur `false` dans le fichier descripteur de l'application, l'appel de la méthode `launchApplication()` n'a aucune incidence.

Avant de présenter l'interface utilisateur pour lancer l'application, il est peut-être préférable d'appeler la méthode `getApplicationVersion()` dans le fichier air.swf. Pour plus d'informations, voir « Vérification à partir d'une page Web de la présence d'une application AIR installée » à la page 269.

Lorsque l'application est invoquée par le biais de la fonctionnalité d'appel du navigateur, l'objet NativeApplication de l'application distribue un objet BrowserInvokeEvent. Pour plus d'informations, voir Appel d'une application AIR à partir du navigateur (développeurs ActionScript) ou Appel d'une application AIR à partir du navigateur (développeurs HTML).

Si vous utilisez la fonction d'appel du navigateur, tenez compte des implications au niveau de la sécurité. Ces implications sont décrites dans Appel d'une application AIR à partir du navigateur (développeurs ActionScript) et Appel d'une application AIR à partir du navigateur (développeurs HTML).

Dès qu'elle est exécutée, l'application peut communiquer avec le contenu SWF dans le navigateur en utilisant la classe LocalConnection. Pour plus d'informations, voir Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs ActionScript) ou Communications avec d'autres occurrences de Flash Player et d'AIR (développeurs HTML).

Remarque : *depuis la version 1.5.3 d'AIR, l'utilisation de l'identifiant d'éditeur est déconseillée. Les identifiants d'éditeur ne sont plus automatiquement affectés aux applications. Les applications peuvent néanmoins continuer à les stipuler à des fins de compatibilité ascendante.*

Chapitre 17 : Mise à jour des applications AIR

L'utilisateur peut installer ou mettre à jour une application AIR en double-cliquant sur un fichier AIR sur l'ordinateur ou à partir du navigateur (à l'aide de la fonction d'installation transparente). Le programme d'installation d'Adobe® AIR™ gère l'installation et avertit l'utilisateur s'il met à jour une application existante.

Par ailleurs, la classe Updater permet à une application installée de se mettre automatiquement à niveau vers une nouvelle version. (Il est possible qu'une application installée détecte qu'une nouvelle version peut être téléchargée et installée.) La classe Updater inclut une méthode `update()`, qui permet de pointer vers un fichier AIR résidant sur l'ordinateur de l'utilisateur et d'effectuer la mise à jour vers cette version. Vous devez mettre en package l'application sous forme de fichier AIR pour pouvoir utiliser la classe Updater. Les applications mises en package sous forme de package ou fichier exécutable natif devraient utiliser les fonctionnalités de mise à jour fournies par la plate-forme native.

L'ID d'application et l'ID d'éditeur d'un fichier AIR de mise à jour doivent correspondre à l'application à mettre à jour. L'ID d'éditeur est issu du certificat de signature, ce qui signifie que la mise à jour et l'application à mettre à jour doivent être signées avec le même certificat.

Dans AIR 1.5.3 ou ultérieur, le fichier descripteur d'application comprend un élément `<publisherID>`. Vous devez utiliser cet élément si vous avez développé des versions de l'application à l'aide d'AIR 1.5.2 ou antérieur. Pour plus d'informations, voir « publisherID » à la page 248.

Depuis AIR 1.1, vous pouvez effectuer la migration d'une application pour utiliser un nouveau certificat développeur. La migration d'une application pour utiliser une nouvelle signature implique la signature du fichier AIR de mise à jour avec l'ancien et le nouveau certificat. La migration de certificats est un processus unidirectionnel. Après la migration, seuls les fichiers AIR signés avec le nouveau certificat (ou avec les deux certificats) sont reconnus comme étant des mises à jour d'une installation existante.

La gestion des mises à jour des applications peut se révéler complexe. AIR 1.5 comprend la nouvelle *structure de mise à jour des applications AdobeAIR*. Cette structure fournit des API qui aident les développeurs à doter les applications AIR de bonnes capacités de mise à jour.

Vous pouvez utiliser la migration de certificats pour passer d'un certificat auto-signé à un certificat de signature de code commercial, ou d'un certificat auto-signé ou commercial à un autre. Si vous n'effectuer pas la migration du certificat, les utilisateurs existants doivent supprimer la version actuelle de votre application avant d'installer la nouvelle. Pour plus d'informations, voir « Changement de certificats » à la page 205.

Il est recommandé d'inclure un mécanisme de mise à jour dans votre application. Si vous créez une nouvelle version de celle-ci, le mécanisme de mise à jour peut inviter l'utilisateur à installer la nouvelle version.

Le programme d'installation d'une application AIR crée des fichiers journaux à l'installation, la mise à jour ou la suppression de l'application. Vous pouvez consulter ces journaux pour déterminer la cause de tout problème d'installation. Voir Installation logs.

Remarque : *les nouvelles versions du moteur d'exécution Adobe AIR comprennent parfois des versions mises à jour de WebKit. Celles-ci sont susceptibles d'apporter des modifications inattendues au contenu HTML dans une application AIR déployée, modifications exigeant éventuellement la mise à jour de l'application. Un mécanisme de mise à jour peut informer l'utilisateur qu'une nouvelle version de l'application est disponible. Pour plus d'informations, voir A propos de l'environnement HTML (développeurs ActionScript) ou A propos de l'environnement HTML (développeurs HTML).*

A propos de la mise à jour des applications

La classe Updater (intégrée au package flash.desktop) contient une méthode unique, `update()`, qui permet de mettre à jour l'application en cours d'exécution. Par exemple, si l'utilisateur dispose d'une version du fichier AIR (« Sample_App_v2.air ») située sur le bureau, le code suivant met à jour l'application :

Exemple 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);
```

Exemple JavaScript :

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Pour que l'application puisse utiliser la classe Updater, l'utilisateur ou l'application doit télécharger la version mise à jour du fichier AIR sur l'ordinateur. Pour plus d'informations, voir « Téléchargement d'un fichier AIR sur l'ordinateur de l'utilisateur » à la page 276.

Résultats de l'appel à la méthode Updater.update()

Lorsqu'une application du moteur d'exécution appelle la méthode `update()`, le moteur d'exécution ferme l'application, puis tente d'installer la nouvelle version à partir du fichier AIR. Le moteur d'exécution vérifie que l'ID d'application et l'ID d'éditeur spécifiés dans le fichier AIR correspondent à ceux de l'application qui appellent la méthode `update()`. (Pour plus d'informations sur l'ID d'application et l'ID d'éditeur, voir « Fichiers descripteurs d'applications AIR » à la page 217.) Il vérifie également que la chaîne de version correspond à la chaîne `version` transmise à la méthode `update()`. Si l'installation réussit, le moteur d'exécution ouvre la nouvelle version de l'application. Dans le cas contraire (si l'installation échoue), il ouvre à nouveau la version existante (pré-installée) de l'application.

Sous Mac OS, pour installer une version mise à jour d'une application, l'utilisateur doit disposer de privilèges système appropriés pour accéder au répertoire de l'application. S'il utilise Windows ou Linux, l'utilisateur doit disposer de privilèges d'administrateur.

Si la version mise à jour de l'application requiert une mise à jour du moteur d'exécution, la nouvelle version du moteur est installée. Pour mettre à jour le moteur d'exécution, un utilisateur doit disposer des privilèges administrateur pour l'ordinateur.

Lors du test d'une application avec l'application de débogage du lanceur AIR (ADL), l'appel de la méthode `update()` provoque une exception d'exécution.

A propos de la chaîne de version

La chaîne spécifiée comme paramètre `version` de la méthode `update()` doit correspondre à la chaîne de l'élément `version` ou `versionNumber` du fichier descripteur d'application associé au fichier AIR à installer. Par mesure de sécurité, la définition du paramètre `version` est obligatoire. En imposant à l'application de vérifier le numéro de version du fichier AIR, celle-ci évite d'installer une version antérieure par inadvertance. (Une version antérieure de l'application risque d'être affectée par des problèmes de sécurité résolus dans l'application actuellement installée.) L'application doit également vérifier la chaîne de version du fichier AIR avec la chaîne de version de l'application installée afin d'éviter les tentatives d'attaque pour rétrograder l'application.

Avant AIR 2.5, utilisez n'importe quel format pour la chaîne de version (« 2.01 » ou « version 2 », par exemple). A partir d'AIR 2.5 ou ultérieur, la chaîne de version doit correspondre à une séquence de nombres d'un, deux ou trois chiffres, séparés par un point. Exemple : « .0 », « 1.0 » et « 67.89.999 » sont tous des nombres de version valides. Validez la chaîne de version avant de mettre à jour l'application.

Si une application Adobe AIR télécharge un fichier AIR via Internet, il est conseillé de disposer d'un mécanisme permettant au service Web d'informer l'application Adobe AIR de la version en cours de téléchargement. L'application peut ainsi utiliser cette chaîne comme paramètre `version` de la méthode `update()`. Si le fichier AIR est obtenu par un autre moyen, dans lequel la version du fichier AIR est inconnue, l'application AIR peut examiner le fichier AIR afin de déterminer les informations de version. (Un fichier AIR est une archive compressée et le fichier descripteur d'application correspond au deuxième enregistrement dans l'archive.)

Pour plus d'informations sur le fichier descripteur d'application, voir « Fichiers descripteurs d'applications AIR » à la page 217.

Flux de travail de signature associé aux mises à jour d'application

La publication ad hoc de mises à jour complique les tâches de gestion de versions d'application multiples, ainsi que le processus de suivi des dates d'expiration des certificats. Un certificat risque d'arriver à expiration avant que vous ne puissiez publier une mise à jour.

Le moteur d'exécution d'Adobe AIR traite une mise à jour d'application publiée sans signature de migration comme une nouvelle application. Les utilisateurs doivent désinstaller l'application AIR en cours pour pouvoir installer la mise à jour.

Pour résoudre le problème, téléchargez chaque application mise à jour avec le certificat le plus récent sur une URL de déploiement distincte. Incluez un mécanisme destiné à vous rappeler d'appliquer une signature de migration lorsque le délai de 180 jours du certificat est entamé, mais pas dépassé. Pour plus d'informations, voir « Signature d'une version mise à jour d'une application AIR » à la page 211.

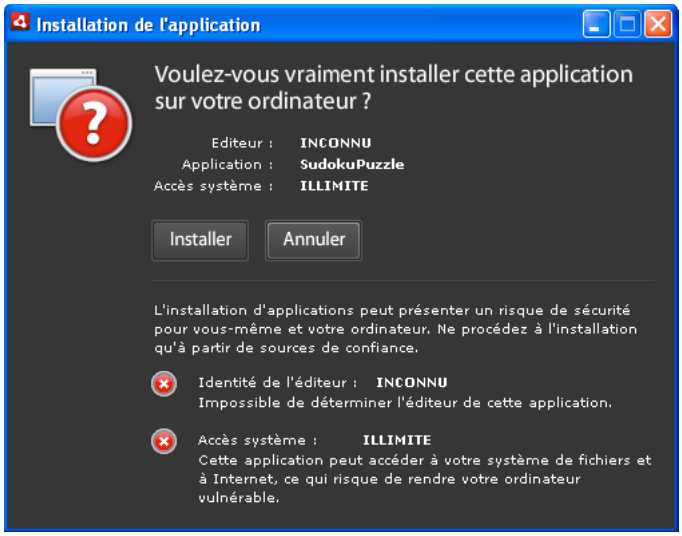
Pour apprendre à appliquer une signature, voir « Commandes de l'outil ADT » à la page 174.

Exécutez les tâches suivantes pour rationaliser le processus d'application des signatures de migration :

- Téléchargez chaque application mise à jour sur une URL de déploiement distincte.
- Téléchargez le fichier XML du descripteur de mise à jour et le certificat associé à la mise à jour le plus récent sur la même URL.
- Signez l'application mise à jour à l'aide du certificat le plus récent.
- Appliquez une signature de migration à l'application mise à jour à l'aide du certificat utilisé pour signer la version précédente (qui réside sur une autre URL).

Présentation d'une interface utilisateur personnalisée pour la mise à jour d'applications

AIR inclut une interface de mise à jour par défaut :



Cette interface est toujours utilisée la première fois qu'un utilisateur installe une version d'une application sur une machine. Vous pouvez cependant définir votre propre interface en vue de l'utiliser dans les occurrences suivantes. Si l'application définit une interface de mise à jour personnalisée, spécifiez un élément `customUpdateUI` dans le fichier descripteur de l'application actuellement installée :

```
<customUpdateUI>true</customUpdateUI>
```

Après avoir installé l'application et une fois que l'utilisateur a ouvert un fichier AIR avec un ID d'application et un ID d'éditeur correspondant à l'application installée, c'est le moteur d'exécution qui ouvre l'application et non le programme d'installation par défaut de l'application AIR. Pour plus d'informations, voir « customUpdateUI » à la page 229.

Lors de son exécution (c'est-à-dire lorsque l'objet `NativeApplication.nativeApplication` distribue un événement `load`), l'application peut décider de mettre ou non l'application à jour (à l'aide de la classe Updater). Si elle décide d'effectuer la mise à jour, elle peut présenter à l'utilisateur sa propre interface d'installation (qui diffère de l'interface d'exécution standard).

Téléchargement d'un fichier AIR sur l'ordinateur de l'utilisateur

Pour utiliser la classe Updater, l'utilisateur ou l'application doit tout d'abord enregistrer un fichier AIR localement sur l'ordinateur de l'utilisateur.

Remarque : *AIR 1.5 comprend une structure de mise à jour qui aide les développeurs à doter les applications AIR de bonnes capacités de mise à jour. L'utilisation de cette structure peut être bien plus simple que l'utilisation directe de la méthode `update()` de la classe Update. Pour plus d'informations, voir « Utilisation de la structure de mise à jour » à la page 280.*

Le code suivant lit un fichier AIR à partir d'une URL (http://example.com/air/updates/Sample_App_v2.air) et l'enregistre dans le répertoire de stockage de l'application.

Exemple 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.");
}
```

Exemple 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.");
}
```

Pour plus d'informations, voir :

- Flux de travail pour la lecture et l'écriture de fichiers (Développeurs ActionScript)
- Flux de travail pour la lecture et l'écriture de fichiers (Développeurs HTML)

Vérifications permettant de savoir si l'application est exécutée pour la première fois

Après avoir mis à jour une application, vous pouvez afficher un message de bienvenue ou de mise en route destiné à l'utilisateur. Au démarrage, l'application vérifie si elle est exécutée pour la première fois afin d'afficher ou non ce message.

Remarque : *AIR 1.5 comprend une structure de mise à jour qui aide les développeurs à doter les applications AIR de bonnes capacités de mise à jour. Cette structure fournit des méthodes simples qui permettent de vérifier si la version de l'application est exécutée pour la première fois. Pour plus d'informations, voir «* *Utilisation de la structure de mise à jour* *à la page 280.*

Pour cela, vous devez enregistrer un fichier dans le répertoire de stockage de l'application lors de l'initialisation de l'application. Chaque fois que l'application démarre, elle vérifie l'existence de ce fichier. Si le fichier n'existe pas, l'application est exécutée pour la première fois pour l'utilisateur actuel. Si le fichier existe, l'application a déjà été exécutée au moins une fois. Si le fichier existe et contient un numéro de version antérieur au numéro de version actuel, vous savez que l'utilisateur exécute la nouvelle version pour la première fois.

L'exemple Flex suivant illustre ce concept :

```
<?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>
```

L'exemple suivant illustre le concept en JavaScript :

```
<html>
    <head>
        <script src="AIRAliases.js" />
        <script>
            var file;
            var currentVersion = "1.2";
            function system extension() {
                file = air.File.appStorageDirectory.resolvePath("Preferences/version.txt");
                air.trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            function checkVersion() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.READ);
                var reversion = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    window.document.getElementById("log").innerHTML
                            = "You have updated to version " + currentVersion + ".\n";
                } else {
                    saveFile();
                }
                window.document.getElementById("log").innerHTML
                                += "Welcome to the application.";
```

```
            }
            function firstRun() {
                window.document.getElementById("log").innerHTML
                            = "Thank you for installing the application. \n"
                            + "This is the first time you have run it.";
                saveFile();
            }
            function saveFile() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        </script>
    </head>
    <body onLoad="system extension()">
        <textarea ID="log" rows="100%" cols="100%" />
    </body>
</html>
```

Si votre application enregistre les données localement (par exemple, dans le répertoire de stockage de l'application), vous pouvez vérifier toutes les données préalablement enregistrées (de versions antérieures) lors de la première exécution.

Utilisation de la structure de mise à jour

La gestion des mises à jour d'applications s'avère parfois laborieuse. La *structure de mise à jour des applications AdobeAIR* contient des API qui permettent aux développeurs d'intégrer des fonctionnalités de mise à jour performantes aux applications AIR. La structure de mise à jour d'AIR exécute les tâches suivantes à l'intention des développeurs :

- Recherche de mises à jour à la fréquence définie ou sur demande de l'utilisateur
- Téléchargement de fichiers AIR (mises à jour) à partir d'une source Web
- Alerte de l'utilisateur lors de la première exécution de la version nouvellement installée
- Confirmation de la part de l'utilisateur que celui-ci souhaite vérifier la présence de mises à jour
- Affichage d'informations sur la version de la nouvelle mise à jour à l'intention de l'utilisateur
- Affichage de la progression du téléchargement et d'informations d'erreurs à l'intention de l'utilisateur

La structure de mise à jour d'AIR propose un exemple d'interface utilisateur dont dispose l'application. Cette interface présente à l'utilisateur des informations de base et les options de configuration associées aux mises à jour de l'application. L'application peut également définir une interface utilisateur personnalisée à utiliser avec la structure de mise à jour.

La structure de mise à jour AIR permet de stocker les informations relatives à la version mise à jour d'une application AIR dans de simples fichiers de configuration XML. Pour la plupart des applications, l'inclusion dans ces fichiers de configuration du code de base fournit à l'utilisateur final des fonctionnalités de mise à jour satisfaisantes.

Même sans faire appel à la structure de mise à jour, Adobe AIR comprend une classe Updater dont disposent les applications AIR pour effectuer la mise à jour vers de nouvelles versions. Cette classe permet à l'application de procéder à une mise à jour vers la version contenue dans un fichier AIR situé sur l'ordinateur de l'utilisateur. Toutefois, la gestion des mises à jour peut se révéler plus complexe que la simple mise à jour d'une application à partir d'un fichier AIR stocké localement.

Fichiers de structure de mise à jour AIR

La structure de mise à jour AIR figure dans le répertoire frameworks/libs/air du kit de développement SDK AIR 2. Elle comprend les fichiers suivants :

- applicationupdater.swc : définit les fonctionnalités de base de la bibliothèque de mise à jour et s'utilise avec ActionScript. Cette version ne comporte pas d'interface utilisateur.
- applicationupdater.swf : définit les fonctionnalités de base de la bibliothèque de mise à jour et s'utilise avec JavaScript. Cette version ne comporte pas d'interface utilisateur.
- applicationupdater_ui.swc : définit une version Flex 4 des fonctionnalités de base de la bibliothèque de mise à jour, et comprend une interface utilisateur dont votre application peut se servir pour afficher les options de mise à jour.
- applicationupdater_ui.swf : définit une version JavaScript des fonctionnalités de base de la bibliothèque de mise à jour, et comprend une interface utilisateur dont votre application peut se servir pour afficher les options de mise à jour.

Pour plus d'informations, voir :

- « Configuration de l'environnement de développement Flex » à la page 281
- « Intégration des fichiers de la structure dans une application AIR de type HTML » à la page 282
- « Exemple de base : utilisation de la version ApplicationUpdaterUI » à la page 282

Configuration de l'environnement de développement Flex

Les fichiers SWC qui figurent dans le répertoire frameworks/libs/air du kit SDK AIR 2 définissent les classes que vous pouvez utiliser lors du développement dans Flash et Flex.

Pour utiliser la structure de mise à jour lors d'une compilation avec le kit SDK Flex, intégrez le fichier ApplicationUpdater.swc ou ApplicationUpdater_UI.swc dans l'appel au compilateur amxmlc. Dans l'exemple suivant, le compilateur charge le fichier ApplicationUpdater.swc dans le sous-répertoire lib du répertoire SDK Flex :

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

Dans l'exemple suivant, le compilateur charge le fichier ApplicationUpdater_UI.swc dans le sous-répertoire lib du répertoire SDK Flex :

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Lors d'un développement avec Flash Builder, ajoutez le fichier SWC dans l'onglet Chemin de la bibliothèque des paramètres du chemin de création Flex de la boîte de dialogue Propriétés.

Assurez-vous de copier les fichiers SWC vers le répertoire que vous référencerez dans le compilateur amxmlc (avec le kit SDK Flex) ou Flash Builder.

Intégration des fichiers de la structure dans une application AIR de type HTML

Le répertoire frameworks/html de la structure de mise à jour contient les fichiers SWF suivants :

- applicationupdater.swf : définit les fonctionnalités de base de la bibliothèque de mise à jour, sans interface utilisateur.
- applicationupdater_ui.swf : définit les fonctionnalités de base de la bibliothèque de mise à jour, notamment une interface utilisateur qui permet à l'application d'afficher les options de mise à jour.

Le code JavaScript des applications AIR peut utiliser les classes définies dans les fichiers SWF.

Pour utiliser la structure de mise à jour, intégrez le fichier applicationupdater.swf ou applicationupdater_ui.swf dans le répertoire (ou un sous-répertoire) de l'application. Ensuite, dans le fichier HTML qui utilisera la structure (dans le code JavaScript), insérez une balise `script` chargeant le fichier :

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

Vous pouvez également charger le fichier applicationupdater_ui.swf à l'aide de la balise `script` :

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

L'API définie dans ces deux fichiers est décrite dans la suite de ce document.

Exemple de base : utilisation de la version ApplicationUpdaterUI

La version ApplicationUpdaterUI de la structure de mise à jour propose une interface de base que vous pouvez facilement utiliser dans l'application. Voici un exemple de base.

D'abord, créez une application AIR qui appelle la structure de mise à jour :

1 S'il s'agit d'une application AIR de type HTML, chargez le fichier applicationupdaterui.swf :

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 Dans la logique du programme de l'application AIR, instanciez un objet ApplicationUpdaterUI.

Dans ActionScript, utilisez le code suivant :

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Dans JavaScript, utilisez le code suivant :

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Vous pouvez ajouter ce code dans une fonction d'initialisation qui s'exécute après le chargement de l'application.

3 Créez un fichier texte nommé updateConfig.xml et ajoutez-lui les éléments suivants :

```
<?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>
```

Modifiez l'élément `URL` du fichier updateConfig.xml pour qu'il corresponde à l'emplacement final du fichier descripteur de la mise à jour sur votre serveur Web (voir la procédure suivante).

L'élément `delay` définit le nombre de jours devant s'écouler avant que l'application ne vérifie la présence de nouvelles mises à jour.

4 Ajoutez le fichier updateConfig.xml dans le répertoire de projet de l'application AIR.

5 Faites en sorte que l'objet updater fasse référence au fichier updateConfig.xml, et appelez la méthode `initialize()` de l'objet.

Dans ActionScript, utilisez le code suivant :

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

Dans JavaScript, utilisez le code suivant :

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 Créez une seconde version de l'application AIR dont la version diffère de la première application. (La version est spécifiée dans le fichier descripteur de l'application, dans l'élément `version`.)

Ajoutez ensuite la mise à jour de l'application AIR sur le serveur Web :

1 Placez la version mise à jour du fichier AIR sur le serveur Web.

2 Créez un fichier texte, updateDescriptor.2.5.xml, et ajoutez-lui le contenu suivant :

```
<?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>
```

Modifiez les éléments `versionNumber`, `URL` et `description` du fichier updateDescriptor.xml en fonction du fichier AIR de mise à jour. Ce format de fichier descripteur de mise à jour est utilisé par les applications qui font appel à la structure de mise à jour intégrée au kit SDK d'AIR 2.5 (et ultérieur).

3 Créez un fichier texte, updateDescriptor.1.0.xml, et ajoutez-lui le contenu suivant :

```
<?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>
```

Modifiez les éléments `version`, `URL` et `description` du fichier updateDescriptor.xml en fonction du fichier AIR de mise à jour. Ce format de fichier descripteur de mise à jour est utilisé par les applications qui font appel à la structure de mise à jour intégrée au kit SDK d'AIR 2 (et ultérieur).

Remarque : *la création de ce second fichier descripteur de mise à jour ne s'impose que si vous prenez en charge les mises à jour d'applications créées avant AIR 2.5.*

4 Placez les fichiers updateDescriptor.2.5.xml et updateDescriptor.1.0.xml dans le répertoire du serveur Web qui contient le fichier AIR de mise à jour.

Il s'agit là d'un exemple de base, mais il fournit des fonctionnalités de mise à jour suffisantes pour la plupart des applications. La suite de ce document décrit l'utilisation de la structure de mise à jour en fonction de vos besoins.

Pour accéder à un autre exemple d'utilisation de la structure de mise à jour, voir l'exemple d'application suivant dans le Centre des développeurs d'Adobe AIR :

- Update Framework in a Flash-based Application
  (http://www.adobe.com/go/learn_air_qs_update_framework_flash_fr)

Mise à jour vers AIR 2.5

En raison de la modification des règles d'affectation de numéro de version aux applications dans AIR 2.5, la structure de mise à jour d'AIR 2 ne peut pas analyser les informations de version dans un fichier descripteur d'application AIR 2.5. Cette incompatibilité signifie que vous devez mettre à jour l'application de sorte à utiliser la nouvelle structure de mise à jour AVANT de mettre à jour l'application de sorte à utiliser le kit SDK d'AIR 2.5. Par conséquent, mettre à jour l'application vers AIR 2.5 ou ultérieur à partir de n'importe quelle version d'AIR antérieure à 2.5 nécessite DEUX mises à jour. La première mise à jour doit utiliser l'espace de noms d'AIR 2 et inclure la bibliothèque de la structure de mise à jour d'AIR 2.5 (vous pouvez continuer à mettre en package l'application à l'aide du kit SDK d'AIR 2.5). La seconde mise à jour peut faire appel à l'espace de noms d'AIR 2.5 et inclure les nouvelles fonctionnalités de l'application.

Vous pouvez également restreindre la mise à jour intermédiaire à la mise à jour de l'application AIR 2.5 directement par le biais de la classe Updater d'AIR.

L'exemple suivant illustre la procédure de mise à jour d'une application de la version 1.0 à la version 2.0. La version 1.0 utilise l'ancien espace de noms 2.0. La version 2.0 fait appel à l'espace de noms 2.5 et possède les nouvelles fonctionnalités mises en œuvre par le biais des API d'AIR 2.5.

1. Créez une version intermédiaire de l'application, version 1.0.1, basée sur la version 1.0 de l'application.
   a. Créez l'application dans la structure de mise à jour de l'application d'AIR 2.5.

      Remarque : *utilisez le fichier* `applicationupdater.swc` *ou* `applicationupdater_ui.swc` *pour les applications AIR basées sur la technologie Flash, le fichier* `applicationupdater.swf` *ou* `applicationupdater_ui.swf` *pour les applications AIR de type HTML.*

   b. Créez un fichier descripteur de mise à jour associé à la version 1.0.1 en utilisant l'ancien espace de noms et la version, comme illustré ci-après :

```
<?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. Créez la version 2.0 de l'application, qui utilise les API et l'espace de noms d'AIR 2.5.
3. Créez un fichier descripteur de mise à jour pour mettre à jour l'application de la version 1.0.1 à la version 2.0.

```
<?xml version="1.0" encoding="utf-8"?>
    <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
        <version>2.0</version>
        <url>http://example.com/updates/sample_2.0.air</url>
        <description>This is the intermediate version.</description>
    </update>
```

Définition des fichiers descripteurs de mise à jour et ajout du fichier AIR dans le serveur Web

Lorsque vous utilisez la structure de mise à jour d'AIR, vous définissez les informations de base relatives à la mise à jour disponible dans des fichiers descripteurs de mise à jour, stockés sur le serveur Web. Un fichier descripteur de mise à jour est un simple fichier XML. La structure de mise à jour intégrée à l'application examine ce fichier pour savoir si une nouvelle version a été téléchargée.

Le format du fichier descripteur de mise à jour a été modifié dans AIR 2.5 et utilise à présent un autre espace de noms. L'espace de noms d'origine correspond à « http://ns.adobe.com/air/framework/update/description/1.0 ». L'espace de noms d'AIR 2.5 correspond à « http://ns.adobe.com/air/framework/update/description/2.5 ».

Les applications AIR créées avant AIR 2.5 ne lisent que la version 1.0 du descripteur de mise à jour. Les applications AIR créées avec la structure de mise à jour intégrée à AIR 2.5 ou ultérieur ne lisent que la version 2.5 du descripteur de mise à jour. En raison de cette incompatibilité des versions, il s'avère souvent nécessaire de créer deux fichiers descripteurs de mise à jour. La logique de mise à jour des versions AIR 2.5 de l'application doit télécharger un descripteur de mise à jour qui utilise le nouveau format. Les versions antérieures de l'application AIR doivent continuer à utiliser le format d'origine. Les deux fichiers doivent être modifiés pour chaque mise à jour (jusqu'à ce que vous arrêtiez la prise en charge des versions créées avant AIR 2.5).

Le fichier descripteur de mise à jour contient les données suivantes :

- `versionNumber` : nouvelle version de l'application AIR. Utilisez l'élément `versionNumber` dans les descripteurs de mise à jour associés à la mise à jour des applications AIR 2.5. La valeur doit correspondre à la chaîne que contient l'élément `versionNumber` du fichier descripteur de la nouvelle application AIR. Si le numéro de version indiqué dans le fichier descripteur de mise à jour ne correspond pas à celui du fichier AIR de mise à jour, la structure de mise à jour renvoie une exception.
- `version` : nouvelle version de l'application AIR. Utilisez l'élément `version` dans les fichiers descripteurs de mise à jour associés à la mise à jour des applications créées avant AIR 2.5. La valeur doit correspondre à la chaîne que contient l'élément `version` du fichier descripteur de la nouvelle application AIR. Si la version indiquée dans le fichier descripteur de mise à jour ne correspond pas à celle du fichier AIR de mise à jour, la structure de mise à jour renvoie une exception.
- `versionLabel` : chaîne de version intelligible destinée à être présentée aux utilisateurs. L'élément `versionLabel` est facultatif, mais est réservé à la version 2.5 des fichiers descripteurs de mise à jour. Utilisez-le si le fichier descripteur d'application contient un élément `versionLabel` défini sur la même valeur.
- `url` : emplacement du fichier AIR de mise à jour. Ce fichier contient la version mise à jour de l'application AIR.
- `description` : détails de la nouvelle version. Ces informations peuvent s'afficher pour l'utilisateur pendant le processus de mise à jour.

Les éléments `version` et `url` sont obligatoires. L'élément `description` est facultatif.

Voici un exemple de fichier descripteur de mise à jour (version 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>
```

Et voici un exemple de fichier descripteur de mise à jour (version 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>
```

Pour définir la balise `description` avec plusieurs langues, utilisez plusieurs éléments `text` définissant un attribut `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>
```

Placez le fichier descripteur de mise à jour et le fichier AIR de mise à jour sur le serveur Web.

Le répertoire templates inclus avec le descripteur de mise à jour comprend des exemples de fichiers descripteur de mise à jour. Ces derniers comprennent des versions de langage unique et des versions multilingues.

Instanciation d'un objet updater

Au terme du chargement de la structure de mise à jour d'AIR dans le code (voir « Configuration de l'environnement de développement Flex » à la page 281 et « Intégration des fichiers de la structure dans une application AIR de type HTML » à la page 282), vous devez instancier un objet updater, comme illustré par l'exemple suivant.

Exemple ActionScript :

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

Exemple JavaScript :

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

Le code précédent utilise la classe ApplicationUpdater (qui ne fournit pas d'interface utilisateur). Pour faire appel à la classe ApplicationUpdaterUI (qui fournit une interface utilisateur), utilisez le code suivant.

Exemple ActionScript :

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Exemple JavaScript :

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Les autres exemples de code de ce document supposent que vous avez instancié un objet updater nommé `appUpdater`.

Configuration des paramètres de la mise à jour

Les classes ApplicationUpdater et ApplicationUpdaterUI peuvent être configurées par l'intermédiaire d'un fichier de configuration fourni avec l'application ou par le code ActionScript ou JavaScript de l'application.

Définition des paramètres de la mise à jour dans un fichier de configuration XML

Ce fichier de configuration de la mise à jour est un fichier XML. Il peut contenir les éléments suivants :

- `updateURL` : chaîne représentant l'emplacement du descripteur de la mise à jour sur le serveur distant. Tout emplacement URLRequest valide est autorisé. Vous devez définir la propriété `updateURL`, par l'intermédiaire du fichier de configuration ou du script (voir « Définition des fichiers descripteurs de mise à jour et ajout du fichier AIR dans le serveur Web » à la page 284). Vous devez définir cette propriété avant d'utiliser l'objet updater (avant d'appeler la méthode `initialize()` de l'objet updater, décrite à la section « Initialisation de la structure de mise à jour » à la page 289).

- `delay` : nombre représentant un intervalle de jours donné (des valeurs comme `0.25` sont autorisées) correspondant à la fréquence de vérification de la présence de mises à jour. Une valeur 0 (définie par défaut) spécifie que l'objet updater ne vérifie pas automatiquement la présence de mise à jour.

Le fichier de configuration d'ApplicationUpdaterUI peut contenir l'élément suivant en plus des éléments `updateURL` et `delay` :

- `defaultUI` : liste des éléments `dialog`. Chaque élément `dialog` possède un attribut `name` correspondant à une boîte de dialogue de l'interface utilisateur. Chaque élément `dialog` possède un attribut `visible` qui spécifie si la boîte de dialogue est visible. La valeur par défaut est `true`. Les valeurs possibles de l'attribut `name` sont les suivantes :
  - `"checkForUpdate"` : correspondant aux boîtes de dialogue Rechercher une mise à jour, Aucune mise à jour et Erreur de mise à jour.
  - `"downloadUpdate"` : correspondant à la boîte de dialogue Télécharger la mise à jour.
  - `"downloadProgress"` : correspondant aux boîtes de dialogue Progression du téléchargement et Erreur de téléchargement.
  - `"installUpdate"` : correspondant à la boîte de dialogue Installer la mise à jour.
  - `"fileUpdate"` : correspondant aux boîtes de dialogue Mise à jour des fichiers, Aucune mise à jour de fichiers et Erreur de fichier.
- `"unexpectedError"` : correspondant à la boîte de dialogue Erreur imprévue.

  Lorsque l'attribut est défini sur `false`, la boîte de dialogue correspondante ne s'affiche pas dans le cadre de la procédure de mise à jour.

Voici un exemple de fichier de configuration pour la structure 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>
```

Voici un exemple de fichier de configuration pour la structure ApplicationUpdaterUI, comprenant une définition de l'élément `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>
```

Pointez la propriété `configurationFile` vers l'emplacement du fichier :

Exemple ActionScript :

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

Exemple JavaScript :

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

Le répertoire templates de la structure de mise à jour comprend un exemple de fichier de configuration, config-template.xml.

Définition des paramètres de mise à jour dans le code ActionScript ou JavaScript

Ces paramètres de configuration peuvent également être définis dans le code de l'application, comme dans le code suivant :

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

Les propriétés de l'objet updater sont `updateURL` et `delay`. Ces propriétés définissent les mêmes paramètres que les éléments `updateURL` et `delay` dans le fichier de configuration : l'URL du fichier descripteur de mise à jour et l'intervalle de vérification des mises à jour. Si vous spécifiez un fichier de configuration *et* des paramètres dans le code, toutes les propriétés définies à l'aide du code sont prioritaires sur les paramètres correspondants dans le fichier de configuration.

Vous devez définir la propriété `updateURL`, par l'intermédiaire du fichier de configuration ou du script (voir « Définition des fichiers descripteurs de mise à jour et ajout du fichier AIR dans le serveur Web » à la page 284), avant d'utiliser l'objet updater (avant d'appeler la méthode `initialize()` de l'objet updater, décrite à la section « Initialisation de la structure de mise à jour » à la page 289).

La structure ApplicationUpdaterUI définit les propriétés supplémentaires suivantes de l'objet updater :

- `isCheckForUpdateVisible` : correspondant aux boîtes de dialogue Rechercher une mise à jour, Aucune mise à jour et Erreur de mise à jour.
- `isDownloadUpdateVisible` : correspondant à la boîte de dialogue Télécharger une mise à jour.
- `isDownloadProgressVisible` : correspondant aux boîtes de dialogue Progression du téléchargement et Erreur de téléchargement.
- `isInstallUpdateVisible` : correspondant à la boîte de dialogue Installer la mise à jour.
- `isFileUpdateVisible` : correspondant aux boîtes de dialogue Mise à jour des fichiers, Aucune mise à jour de fichiers et Erreur de fichier.
- `isUnexpectedErrorVisible` : correspondant à la boîte de dialogue Erreur imprévue.

Chaque propriété correspond à une ou plusieurs boîtes de dialogue de l'interface utilisateur ApplicationUpdaterUI. Chaque propriété est une valeur booléenne, dont la valeur par défaut est `true`. Lorsqu'elle est définie sur `false`, les boîtes de dialogue correspondantes ne s'affichent pas lors de la procédure de mise à jour.

Les propriétés de ces boîtes de dialogue remplacent les paramètres du fichier de configuration de mise à jour.

Processus de mise à jour

La structure de mise à jour AIR complète le processus de mise à jour selon la procédure suivante :

1. L'initialisation du programme de mise à jour contrôle si une vérification de mise à jour a été effectuée au cours de l'intervalle défini (voir « Configuration des paramètres de la mise à jour » à la page 286). Si une vérification de mise à jour doit être effectuée, le processus se poursuit.
2. Le programme de mise à jour télécharge et interprète le fichier descripteur de mise à jour.
3. Le programme de mise à jour télécharge le fichier AIR de mise à jour.
4. Le programme de mise à jour installe la version mise à jour de l'application.

L'objet updater distribue des événements à la fin de chacune de ces étapes. Dans la version ApplicationUpdater, vous pouvez annuler les événements qui indiquent le succès d'une étape du processus. Si vous annulez l'un de ces événements, l'étape suivante du processus est annulée. Dans la version ApplicationUpdaterUI, le programme de mise à jour affiche une boîte de dialogue qui permet à l'utilisateur d'annuler ou de continuer à chaque étape du processus.

Si vous annulez l'événement, vous pouvez appeler les méthodes de l'objet updater pour reprendre le processus.

Au fur et à mesure que la version ApplicationUpdater du programme de mise à jour poursuit le processus, l'état en cours est enregistré dans une propriété `currentState`. Cette propriété est définie sur une chaîne qui peut prendre les valeurs suivantes :

- `"UNINITIALIZED"` : le programme de mise à jour n'a pas été initialisé.
- `"INITIALIZING"` : le programme de mise à jour est en cours d'initialisation.
- `"READY"` : le programme de mise à jour a été initialisé.
- `"BEFORE_CHECKING"` : le programme de mise à jour n'a pas encore vérifié la présence du fichier descripteur de mise à jour.
- `"CHECKING"` : le programme de mise à jour recherche un fichier descripteur de mise à jour.
- `"AVAILABLE"` : le fichier descripteur de mise à jour est disponible.
- `"DOWNLOADING"` : le programme de mise à jour télécharge le fichier AIR.
- `"DOWNLOADED"` : le programme de mise à jour a téléchargé le fichier AIR.
- `"INSTALLING"` : le programme de mise à jour installe le fichier AIR.
- `"PENDING_INSTALLING"` : le programme de mise à jour a été initialisé et des mises à jour sont en attente.

Certaines méthodes de l'objet updater ne s'exécutent que si le programme de mise à jour est dans un certain état.

Initialisation de la structure de mise à jour

Après la définition des propriétés de configuration (voir « Exemple de base : utilisation de la version ApplicationUpdaterUI » à la page 282), appelez la méthode `initialize()` pour initialiser la mise à jour :

```
appUpdater.initialize();
```

Cette méthode effectue les opérations suivantes :

- Elle initialise la structure de mise à jour, en installant silencieusement et de façon synchrone les mises à jour en attente. Il est obligatoire d'appeler cette méthode au démarrage de l'application car elle peut redémarrer l'application lorsqu'elle est appelée.
- Elle vérifie si une mise à jour a été reportée et, le cas échéant, procède à son installation.
- Si une erreur se produit pendant le processus de mise à jour, elle efface le fichier de mise à jour et les informations de version dans l'emplacement de stockage de l'application.
- Si le délai est arrivé à expiration, elle démarre le processus de mise à jour. Sinon, elle réinitialise le minuteur.

L'appel à cette méthode peut entraîner la distribution des événements suivants par l'objet updater :

- `UpdateEvent.INITIALIZED` : distribué lorsque l'initialisation est terminée.
- `ErrorEvent.ERROR` : distribué lorsqu'une erreur se produit pendant l'initialisation.

Lors de la distribution de l'événement `UpdateEvent.INITIALIZED`, le processus de mise à jour est terminé.

Lorsque vous appelez la méthode `initialize()`, le programme de mise à jour démarre le processus et effectue toutes les étapes, en fonction du paramètre de délai du minuteur. Toutefois, vous pouvez également démarrer le processus de mise à jour à tout moment en appelant la méthode `checkNow()` de l'objet updater :

```
appUpdater.checkNow();
```

Cette méthode n'a aucune incidence si le processus de mise à jour est déjà en cours d'exécution. Sinon, elle démarre le processus de mise à jour.

L'objet updater peut distribuer l'événement suivant après un appel à la méthode `checkNow()` :

- L'événement `UpdateEvent.CHECK_FOR_UPDATE` juste avant la tentative de téléchargement du fichier descripteur de mise à jour.

Si vous annulez l'événement `checkForUpdate`, vous pouvez appeler la méthode `checkForUpdate()` de l'objet updater. (Voir la section suivante.) Si vous n'annulez pas l'événement, le processus de mise jour recherche le fichier descripteur de mise à jour.

Gestion du processus de mise à jour dans la version ApplicationUpdaterUI

Dans la version ApplicationUpdaterUI, l'utilisateur peut annuler le processus via les boutons Annuler des boîtes de dialogue de l'interface utilisateur. De même, vous pouvez annuler par programmation le processus de mise à jour en appelant la méthode `cancelUpdate()` de l'objet ApplicationUpdaterUI.

Vous pouvez définir les propriétés de l'objet ApplicationUpdaterUI ou définir les éléments du fichier de configuration de mise à jour pour spécifier les confirmations de boîtes de dialogue devant être affichées par le programme de mise à jour. Pour plus d'informations, voir « Configuration des paramètres de la mise à jour » à la page 286.

Gestion du processus de mise à jour dans la version ApplicationUpdater

Vous pouvez appeler la méthode `preventDefault()` des objets d'événements distribués par l'objet ApplicationUpdater pour annuler des étapes du processus de mise à jour (voir « Processus de mise à jour » à la page 288). L'annulation du comportement par défaut permet à votre application d'afficher un message à l'utilisateur pour lui demander s'il souhaite continuer.

Les sections suivantes montrent comment poursuivre le processus de mise à jour lorsqu'une étape du processus a été annulée.

Téléchargement et interprétation du fichier descripteur de mise à jour

L'objet ApplicationUpdater distribue l'événement `checkForUpdate` avant le début du processus de mise à jour, juste avant que le programme de mise à jour ne tente de télécharger le fichier descripteur de mise à jour. Si vous annulez le comportement par défaut de l'événement `checkForUpdate`, le programme de mise à jour ne télécharge pas le fichier descripteur de mise à jour. Pour reprendre le processus de mise à jour, vous pouvez appeler la méthode `checkForUpdate()` :

```
appUpdater.checkForUpdate();
```

L'appel à la méthode `checkForUpdate()` oblige le programme de mise à jour à effectuer un téléchargement asynchrone et à interpréter le fichier descripteur de mise à jour. À la suite d'un appel à la méthode `checkForUpdate()`, l'objet updater peut distribuer les événements suivants :

- `StatusUpdateEvent.UPDATE_STATUS` : le programme de mise à jour à bien téléchargé et interprété le fichier descripteur de mise à jour. Les propriétés de cet événement sont les suivantes :
  - `available` : valeur booléenne. Définie sur `true` si une version différente de l'application en cours est disponible, sur `false` dans le cas contraire (la version est identique).
  - `version` : chaîne. La version indiquée dans le fichier descripteur d'application du fichier de mise à jour.
  - `details` : tableau. S'il n'y a pas de versions localisées de la description, ce tableau renvoie une chaîne vide (`""`) en tant que premier élément et la description en tant que second élément.

S'il existe plusieurs versions de la description (dans le fichier descripteur de mise à jour), le tableau contient plusieurs sous-tableaux. Chaque tableau comprend deux éléments : le premier étant le code de langue (par exemple `"en"`), et le second la description correspondante (une chaîne) dans cette langue. Voir « Définition des fichiers descripteurs de mise à jour et ajout du fichier AIR dans le serveur Web » à la page 284.

- `StatusUpdateErrorEvent.UPDATE_ERROR` : une erreur s'est produite et le programme de mise à jour n'a pas pu télécharger ou interpréter le fichier descripteur de mise à jour.

Téléchargement du fichier AIR de mise à jour

L'objet ApplicationUpdater distribue l'événement `updateStatus` dès que le programme de mise à jour à bien téléchargé et interprété le fichier descripteur de mise à jour. Le comportement par défaut consiste à démarrer le téléchargement du fichier de mise à jour lorsque ce dernier est disponible. Si vous annulez le comportement par défaut, vous pouvez reprendre le processus en appelant la méthode `downloadUpdate()` :

```
appUpdater.downloadUpdate();
```

L'appel de cette méthode oblige le programme de mise à jour à effectuer un téléchargement asynchrone de la mise à jour du fichier AIR.

La méthode `downloadUpdate()` peut distribuer les événements suivants :

- `UpdateEvent.DOWNLOAD_START` : la connexion au serveur a été établie. Lorsque la bibliothèque ApplicationUpdaterUI est utilisée, cet événement affiche une boîte de dialogue dans laquelle une barre de progression permet de suivre l'avancée du téléchargement.
- `ProgressEvent.PROGRESS` : distribué régulièrement au fur et à mesure de la progression du téléchargement du fichier.
- `DownloadErrorEvent.DOWNLOAD_ERROR` : distribué lorsqu'une erreur se produit pendant la connexion ou le téléchargement du fichier de mise à jour. Cet événement est également distribué pour les états HTTP non valides (par exemple « 404 - Fichier introuvable »). Cet événement a une propriété `errorID`, un nombre entier qui définit d'autres informations d'erreur. Une autre propriété `subErrorID` peut contenir d'autres informations d'erreur.
- `UpdateEvent.DOWNLOAD_COMPLETE` : le programme de mise à jour à bien téléchargé et interprété le fichier descripteur de mise à jour. Si vous n'annulez pas cet événement, la version ApplicationUpdater procède à l'installation de la mise à jour. Dans la version ApplicationUpdaterUI, une boîte de dialogue est présentée à l'utilisateur pour lui proposer de continuer.

Mise à jour de l'application

L'objet ApplicationUpdater déclenche l'événement `downloadComplete` lorsque le téléchargement du fichier de mise à jour est terminé. Si vous annulez le comportement par défaut, vous pouvez reprendre le processus en appelant la méthode `installUpdate()` :

```
appUpdater.installUpdate(file);
```

Un appel à cette méthode oblige le programme de mise à jour à installer la mise à jour du fichier AIR. La méthode comprend un paramètre, `file`, correspondant à un objet File qui fait référence au fichier AIR à utiliser en tant que mise à jour.

L'objet ApplicationUpdater peut distribuer l'événement `beforeInstall` à la suite d'un appel à la méthode `installUpdate()` :

- `UpdateEvent.BEFORE_INSTALL` : distribué juste avant l'installation de la mise à jour. Il est parfois utile d'empêcher l'installation de la mise à jour à ce stade de sorte que l'utilisateur puisse terminer son travail en cours avant de lancer l'installation. L'appel à la méthode `preventDefault()` de l'objet Event reporte l'installation jusqu'au prochain démarrage et empêche le lancement de tout processus de mise à jour supplémentaire. (Cela comprend les mises à jour résultant d'un appel à la méthode `checkNow()` ou liées à une vérification périodique.)

Installation à partir d'un fichier AIR arbitraire

Vous pouvez appeler la méthode `installFromAIRFile()` pour installer la mise à jour à partir d'un fichier AIR situé sur l'ordinateur de l'utilisateur :

```
appUpdater.installFromAIRFile();
```

Cette méthode oblige le programme de mise à jour à installer la mise à jour de l'application à partir du fichier AIR.

La méthode `installFromAIRFile()` peut distribuer les événements suivants :

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS` : distribué lorsque ApplicationUpdater a bien validé le fichier envoyé par la méthode `installFromAIRFile()`. Les propriétés de cet événement sont les suivantes :
  - `available` : définie sur `true` si une version différente de l'application actuelle est disponible ; `false`, dans le cas contraire (les versions sont identiques).
  - `version` : chaîne représentant la nouvelle version disponible.
  - `path` : représente le chemin natif du fichier de mise à jour.

  Vous pouvez annuler cet événement si la propriété available de l'objet StatusFileUpdateEvent est définie sur `true`. L'annulation de l'événement annule également le processus de mise à jour. Pour poursuivre le processus de mise à jour, appelez la méthode `installUpdate()`.
- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR` : une erreur s'est produite et le programme de mise à jour n'a pas pu installer l'application AIR.

Annulation du processus de mise à jour

Pour annuler le processus de mise à jour, vous pouvez appeler la méthode `cancelUpdate()` :

```
appUpdater.cancelUpdate();
```

Cette méthode annule tous les téléchargements en attente, en supprimant tous les fichiers partiellement téléchargés, et réinitialise le minuteur de vérification périodique.

Cette méthode n'a aucune incidence si le programme de mise à jour est initialisé.

Localisation de l'interface ApplicationUpdaterUI

La classe ApplicationUpdaterUI fournit une interface utilisateur par défaut pour le processus de mise à jour. Celle-ci comprend des boîtes de dialogue qui permettent à l'utilisateur de démarrer le processus, de l'annuler et d'effectuer d'autres actions associées.

L'élément `description` du fichier descripteur de mise à jour vous permet de définir la description de l'application en plusieurs langues. Utilisez plusieurs éléments `text` définissant des attributs `lang`, comme dans l'exemple suivant :

```
<?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>
```

La structure de mise à jour utilise la description correspondant le mieux au chaînage de localisation de l'utilisateur final. Pour plus d'informations, voir Définition du fichier descripteur de mise à jour et ajout du fichier AIR dans votre serveur Web.

Les développeurs Flex peuvent ajouter directement une nouvelle langue dans le regroupement `"ApplicationUpdaterDialogs"`.

Les développeurs JavaScript peuvent appeler la méthode `addResources()` de l'objet updater. Cette méthode ajoute dynamiquement un nouveau regroupement de ressources pour une langue. Le regroupement de ressources définit les chaînes localisées d'une langue. Ces chaînes sont utilisées dans les champs de texte des différentes boîtes de dialogue.

Les développeurs JavaScript peuvent utiliser la propriété `localeChain` de la classe ApplicationUpdaterUI pour définir la chaîne de paramètres régionaux utilisée par l'interface utilisateur. En général, seuls les développeurs JavaScript (HTML) utilisent cette propriété. Les développeurs Flex peuvent utiliser ResourceManager pour gérer la chaîne de paramètres régionaux.

Par exemple, le code JavaScript suivant définit des regroupements de ressources pour le roumain et le hongrois :

```
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;
```

Pour plus d'informations, voir la description de la méthode `addResources()` de la classe ApplicationUpdaterUI dans le guide de référence du langage.

Chapitre 18 : Affichage du code source

De même qu'il est possible d'afficher le code source d'une page HTML dans un navigateur Web, les utilisateurs peuvent afficher le code source d'une application AIR HTML. Le kit de développement d'Adobe® AIR® comprend le fichier JavaScript AIRSourceViewer.js, que vous pouvez intégrer à l'application pour permettre aux utilisateurs finaux de consulter facilement le code source.

Chargement, configuration et ouverture de Source Viewer

Le code de Source Viewer figure dans un fichier JavaScript, AIRSourceViewer.js, qui réside dans le répertoire frameworks du kit SDK AIR. Pour intégrer Source Viewer à l'application, copiez le fichier AIRSourceViewer.js dans le répertoire de projet de l'application et chargez-le par le biais d'une balise script dans le fichier HTML principal de l'application :

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

Le fichier AIRSourceViewer.js définit la classe SourceViewer, à laquelle vous pouvez accéder à partir du code JavaScript en appelant `air.SourceViewer`.

La classe SourceViewer définit trois méthodes : `getDefault()`, `setup()` et `viewSource()`.

Méthode	Description
`getDefault()`	Méthode statique. Renvoie une occurrence de SourceViewer, qui vous permet d'appeler les autres méthodes.
`setup()`	Applique des paramètres de configuration à Source Viewer. Pour plus d'informations, voir « Configuration de Source Viewer » à la page 294.
`viewSource()`	Ouvre une nouvelle fenêtre dans laquelle l'utilisateur peut rechercher et ouvrir les fichiers sources de l'application hôte.

Remarque : *le code utilisant Source Viewer doit se trouver dans le sandbox de sécurité de l'application (dans un fichier résidant dans le répertoire de l'application).*

Le code JavaScript suivant, par exemple, instancie un objet SourceViewer et ouvre la fenêtre Source Viewer, qui répertorie tous les fichiers sources :

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Configuration de Source Viewer

La méthode `config()` applique certains paramètres à Source Viewer. Elle gère un seul paramètre : `configObject`. L'objet `configObject` possède les propriétés suivantes qui définissent les paramètres de configuration de Source Viewer : `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` et `typesToAdd`.

default

Chaîne spécifiant le chemin relatif du fichier initial à afficher dans Source Viewer.

Le code JavaScript suivant, par exemple ouvre la fenêtre Source Viewer et y affiche initialement le fichier index.html :

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

Tableau de chaînes spécifiant les fichiers et répertoires à exclure de la liste de Source Viewer. Les chemins sont relatifs au répertoire de l'application. Les caractères génériques ne sont pas pris en charge.

Le code JavaScript suivant, par exemple, ouvre la fenêtre Source Viewer, qui répertorie tous les fichiers sources, à l'exception du fichier AIRSourceViewer.js et des fichiers résidant dans les sous-répertoires Images et Sounds :

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

Tableau comprenant deux nombres, qui spécifient les coordonnées x et y initiales de la fenêtre Source Viewer.

Le code JavaScript suivant, par exemple, ouvre cette fenêtre aux coordonnées d'écran [40, 60] (X = 40, Y = 60) :

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

Valeur booléenne spécifiant si la fenêtre Source Viewer doit être modale (true) ou non (false). Cette fenêtre est modale par défaut.

Le code JavaScript suivant, par exemple, ouvre la fenêtre Source Viewer de telle sorte que l'utilisateur puisse interagir avec elle et avec les fenêtres de toute application :

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

Tableau de chaînes spécifiant les types de fichier à ajouter à la liste de Source Viewer, en plus des types inclus par défaut.

Ces types de fichier correspondent à :

- Fichiers texte : TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Fichiers image : JPG, JPEG, PNG, GIF

  Si vous n'entrez pas de valeur, tous les types par défaut sont inclus (à l'exception de ceux spécifiés par la propriété `typesToExclude`).

  Le code JavaScript suivant, par exemple, ouvre la fenêtre Source Viewer et inclut les fichiers VCF et VCARD :

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToAdd = ["text.vcf", "text.vcard"];
  viewer.viewSource(configObj);
  ```

Pour chaque type de fichier inclus, vous devez spécifier « text » (pour les types de fichier texte) ou « image » (pour les types de fichier image).

typesToExclude
Tableau de chaînes spécifiant les types de fichier à exclure de Source Viewer.

Ces types de fichier correspondent à :

- Fichiers texte : TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Fichiers image : JPG, JPEG, PNG, GIF

Le code JavaScript suivant, par exemple, ouvre la fenêtre Source Viewer sans répertorier les fichiers GIF ou XML :

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

Pour chaque type de fichier répertorié, vous devez spécifier « text » (pour les types de fichier `texte`) ou « `image` » (pour les types de fichier image).

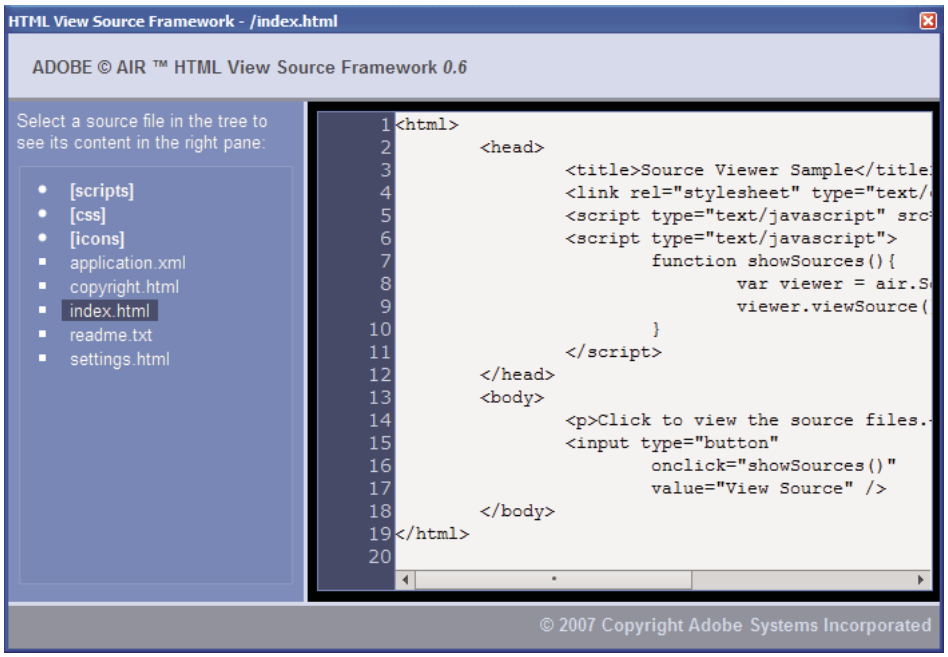
Ouverture de Source Viewer

Il est souhaitable d'inclure un élément d'interface utilisateur (lien, bouton, option de menu, par exemple) qui, lorsque l'utilisateur le sélectionne, appelle Source Viewer. L'application simple ci-dessous, par exemple, ouvre Source Viewer lorsque l'utilisateur clique sur un lien :

```
<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 utilisateur de Source Viewer

Lorsque l'application appelle la méthode `viewSource()` d'un objet SourceViewer, l'application AIR ouvre une fenêtre Source Viewer. Cette fenêtre comprend une liste de fichiers et répertoires sources (à gauche) et une zone d'affichage contenant le code source du fichier sélectionné (à droite) :



Les répertoires sont placés entre crochets. L'utilisateur peut cliquer sur un répertoire pour développer ou masquer son contenu.

Source Viewer peut afficher la source des fichiers texte portant une extension reconnue (HTML, JS, TXT, XML, etc.), ainsi que des fichiers image dotés d'une extension graphique reconnue (JPG, JPEG, PNG et GIF). Si l'utilisateur sélectionne un fichier dont l'extension est inconnue, un message d'erreur indiquant qu'il est impossible d'extraire le contenu d'un fichier de ce type s'affiche.

Les fichiers sources exclus par le biais de la méthode `setup()` ne sont pas répertoriés (voir « Chargement, configuration et ouverture de Source Viewer » à la page 294).

Chapitre 19 : Débogage à l'aide de l'outil AIR HTML Introspector

Le kit de développement d'Adobe® AIR® comprend le fichier JavaScript AIRIntrospector.js, que vous pouvez intégrer à l'application HTML pour permettre son débogage.

Présentation de l'outil AIR HTML Introspector

Adobe AIR HTML/JavaScript Application Introspector (appelé AIR HTML Introspector) propose des fonctions qui facilitent le développement et le débogage des applications HTML :

- Cette application contient un outil d'introspection qui permet de pointer sur un élément de l'interface utilisateur de l'application pour afficher ses propriétés de marquage et DOM.
- Elle intègre une console à laquelle vous envoyez des références d'objet à des fins d'introspection, ce qui vous permet d'ajuster la valeur des propriétés et d'exécuter le code JavaScript. Vous pouvez également sérialiser les objets vers la console, auquel cas il vous est impossible de modifier les données. Vous avez aussi la possibilité de copier et d'enregistrer du texte de la console.
- Elle comprend une arborescence des propriétés et fonctions DOM.
- Elle permet de modifier les attributs et les nœuds de texte des éléments DOM.
- Elle répertorie les liens, styles CSS, images et fichiers JavaScript chargés dans l'application.
- Elle permet d'afficher la source HTML initiale et la source de marquage en cours de l'interface utilisateur.
- Elle permet d'accéder aux fichiers du répertoire de l'application. (Cette fonction est uniquement disponible pour la console AIR HTML Introspector ouverte pour le sandbox de l'application. Elle ne l'est pas dans les consoles ouvertes pour du contenu provenant d'un sandbox hors application.)
- Elle comprend un visualisateur pour l'affichage des objets XMLHttpRequest et de leurs propriétés, y compris les propriétés `responseText` et `responseXML` (le cas échéant).
- Vous pouvez rechercher du texte dans le code et les fichiers source.

Chargement du code de l'outil AIR HTML Introspector

Le code de l'outil AIR HTML Introspector figure dans un fichier JavaScript, AIRIntrospector.js, qui réside dans le répertoire frameworks du kit SDK AIR. Pour intégrer cet outil à l'application, copiez le fichier AIRIntrospector.js dans le répertoire de projet de l'application et chargez-le par le biais d'une balise script dans le fichier HTML principal de l'application :

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

Intégrez également le fichier dans les fichiers HTML qui correspondent aux différentes fenêtres natives de l'application.

Important : *incluez le fichier AIRIntrospector.js lors du développement et du débogage de l'application seulement. Supprimez-le de l'application AIR mise en package que vous distribuez.*

Le fichier AIRIntrospector.js définit la classe Console, à laquelle vous pouvez accéder à partir du code JavaScript en appelant `air.Introspector.Console`.

Remarque : *le code utilisant l'outil AIR HTML Introspector doit se trouver dans le sandbox de sécurité de l'application (dans un fichier résidant dans le répertoire de l'application).*

Inspection d'un objet dans l'onglet Console

La classe Console définit cinq méthodes : `log()`, `warn()`, `info()`, `error()` et `dump()`.

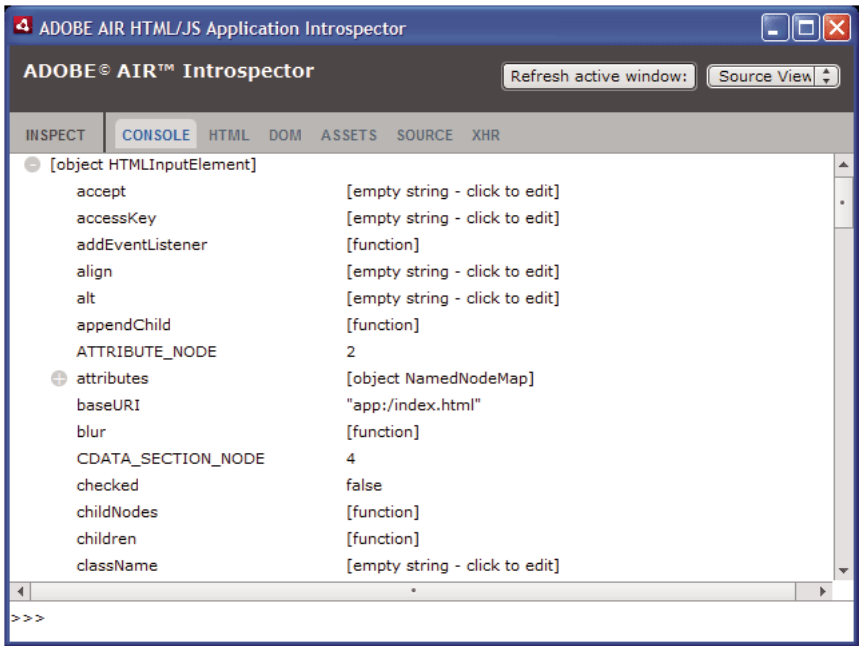
Les méthodes `log()`, `warn()`, `info()` et `error()` permettent d'envoyer un objet à l'onglet Console. La méthode `log()` est la plus élémentaire d'entre elles. Le code suivant envoie un objet simple, représenté par la variable `test`, à l'onglet Console :

```
var test = "hello";
air.Introspector.Console.log(test);
```

Il est cependant plus judicieux d'envoyer un objet complexe à l'onglet Console. La page HTML suivante, par exemple, contient un bouton (`btn1`) qui appelle une fonction qui envoie l'objet lui-même à l'onglet 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>
```

Lorsque vous cliquez sur le bouton, l'onglet Console affiche l'objet btn1 et, et vous pouvez développer l'arborescence de celui-ci pour inspecter ses propriétés :



Vous pouvez modifier une propriété de l'objet en cliquant sur le listing à droite du nom de la propriété et en modifiant le texte correspondant.

Les méthodes `info()`, `error()` et `warn()` sont similaires à la méthode `log()`. Toutefois, lorsque vous les appelez, une icône s'affiche au début de la ligne sur la console :

Méthode	**Icône**
`info()`	
`error()`	
`warn()`	

Les méthodes `log()`, `warn()`, `info()` et `error()` se contentent d'envoyer une référence à un objet ; par conséquent, seules les propriétés présentes au moment de l'affichage sont disponibles. Pour sérialiser l'objet lui-même, utilisez la méthode `dump()`. Cette méthode gère deux paramètres :

Paramètre	**Description**
`dumpObject`	Objet à sérialiser.
`levels`	Nombre maximal de niveaux à examiner dans l'arborescence de l'objet (outre le niveau racine). La valeur par défaut est 1 (autrement dit, un niveau sous le niveau racine de l'arborescence est affiché). Ce paramètre est facultatif.

L'appel de la méthode `dump()` sérialise un objet avant de l'envoyer à l'onglet Console ; vous ne pouvez donc pas modifier ses propriétés. Considérons par exemple le code qui suit :

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

Lorsque vous exécutez ce code, l'objet `testObject` et ses propriétés s'affichent sur la console, mais il vous est impossible de modifier la valeur des propriétés.

Configuration de l'outil AIR HTML Introspector

Vous pouvez configurer la console en définissant les propriétés de la variable globale`AIRIntrospectorConfig`. Le code JavaScript suivant, par exemple, configure l'outil AIR HTML Introspector de sorte à renvoyer automatiquement à la ligne le texte des colonnes à partir de 100 caractères :

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

Veillez à définir les propriétés de la variable `AIRIntrospectorConfig` avant de charger le fichier AIRIntrospector.js (via une balise `script`).

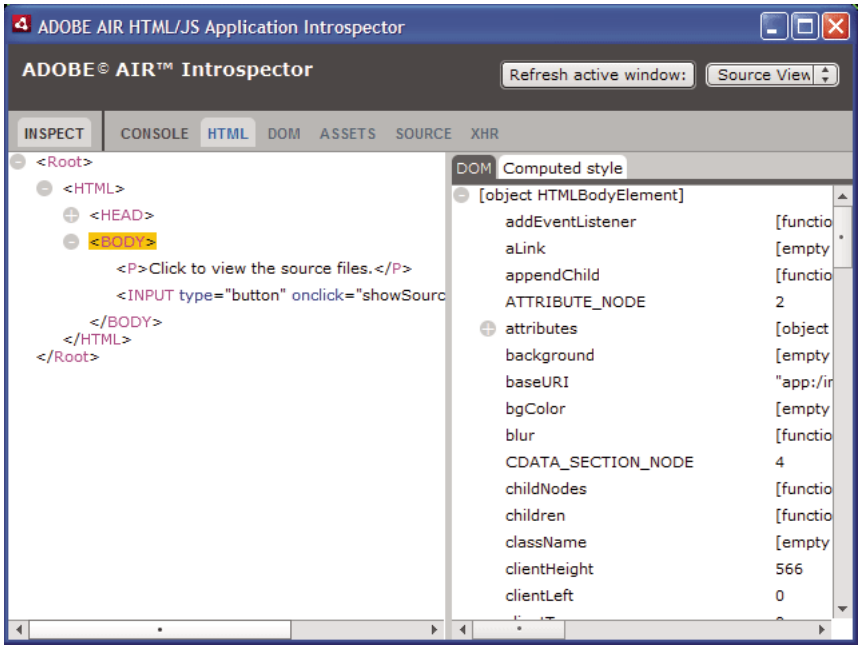
Les huit propriétés de la variable `AIRIntrospectorConfig` sont les suivantes :

Propriété	Valeur par défaut	Description
`closeIntrospectorOnExit`	`true`	Configure la fenêtre de l'outil AIR HTML Introspector de sorte qu'elle se ferme à la fermeture de toutes les autres fenêtres de l'application.
`debuggerKey`	123 (touche F12)	Code de touche du raccourci clavier permettant d'afficher et de masquer la fenêtre de l'outil AIR HTML Introspector.
`debugRuntimeObjects`	`true`	Configure l'outil AIR HTML Introspector de sorte à développer les objets d'exécution en plus des outils définis dans JavaScript.
`flashTabLabels`	`true`	Configure les onglets Console et XMLHttpRequest de sorte qu'ils clignotent en cas de modification (lorsque du texte y est enregistré par exemple).
`introspectorKey`	122 (touche F11)	Code de touche du raccourci clavier permettant d'ouvrir le panneau Inspect.
`showTimestamp`	`true`	Configure l'onglet Console de sorte à afficher les informations d'horodatage au début de chaque ligne.
`showSender`	`true`	Configure l'onglet Console de sorte à afficher des informations sur l'objet qui envoie le message au début de chaque ligne.
`wrapColumns`	2000	Nombre de colonnes au-delà duquel se produit un retour à la ligne dans les fichiers sources.

Interface de l'outil AIR HTML Introspector

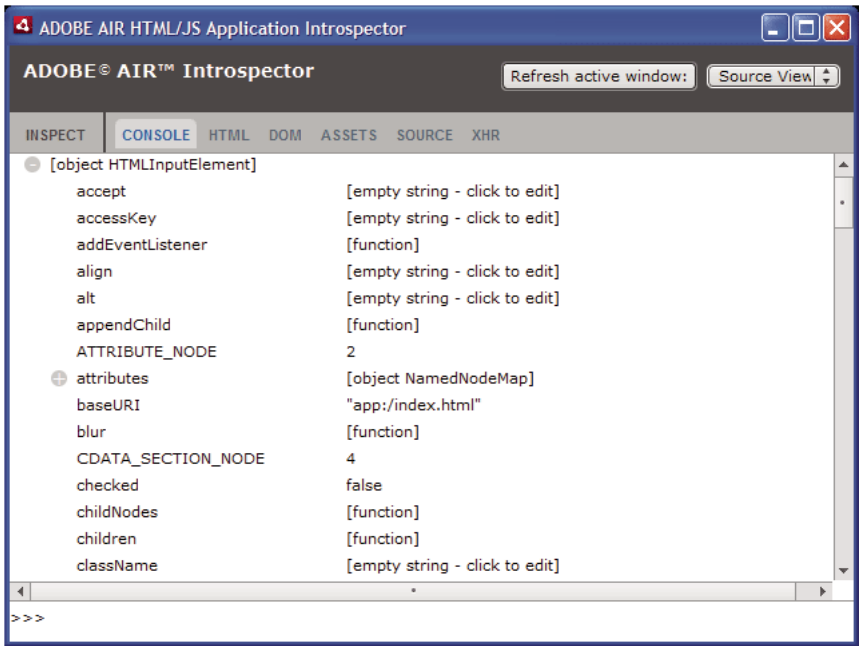
Pour ouvrir la fenêtre de l'outil AIR HTML Introspector lors du débogage de l'application, appuyez sur la touche F12 ou appelez une des méthodes de la classe Console (voir « Inspection d'un objet dans l'onglet Console » à la page 299). Vous pouvez remplacer par une autre touche la touche d'accès rapide F12 (voir « Configuration de l'outil AIR HTML Introspector » à la page 301).

La fenêtre de l'outil AIR HTML Introspector contient six onglets, Console, HTML, DOM, Assets, Source et XHR illustrés ci-dessous :



Onglet Console

L'onglet Console affiche les valeurs des propriétés transmises en tant que paramètres à l'une des méthodes de la classe air.Introspector.Console. Pour plus d'informations, voir « Inspection d'un objet dans l'onglet Console » à la page 299.

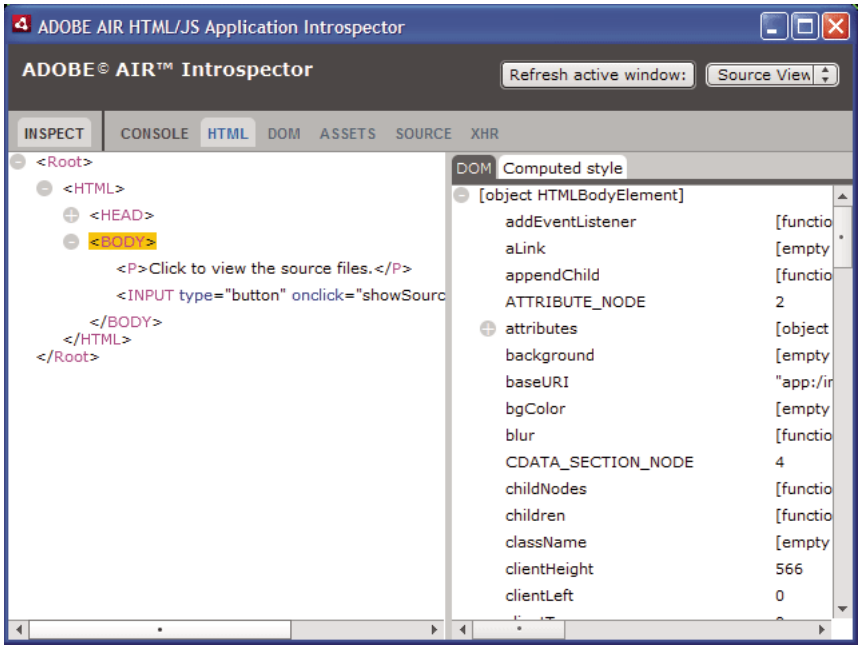


- Pour effacer le contenu de la console, cliquez avec le bouton droit de la souris sur le texte et sélectionnez Clear Console.

- Pour enregistrer le texte de l'onglet Console dans un fichier, cliquez avec le bouton droit de la souris sur l'onglet et sélectionnez Save Console To File.
- Pour enregistrer le texte de l'onglet Console dans le Presse-papiers, cliquez avec le bouton droit de la souris sur l'onglet et sélectionnez Save Console To Clipboard. Pour copier le texte sélectionné uniquement dans le Presse-papiers, cliquez dessus avec le bouton droit de la souris et sélectionnez Copy.
- Pour enregistrer le texte de la classe Console dans un fichier, cliquez avec le bouton droit de la souris sur l'onglet et sélectionnez Save Console To File.
- Pour rechercher du texte dans l'onglet, appuyez sur Ctrl+F (Windows) ou Commande+F (Mac OS). (La recherche ne porte pas sur les nœuds de l'arborescence qui ne sont pas visibles.)

Onglet HTML

L'onglet HTML permet d'afficher l'intégralité du modèle HTML DOM dans une structure arborescente. Cliquez sur un élément pour afficher ses propriétés dans la partie droite de l'onglet. Cliquez sur les icônes + et - pour développer ou réduire un nœud dans l'arborescence.



Vous pouvez modifier tout attribut ou élément de texte dans l'onglet HTML. La valeur modifiée est répercutée dans l'application.

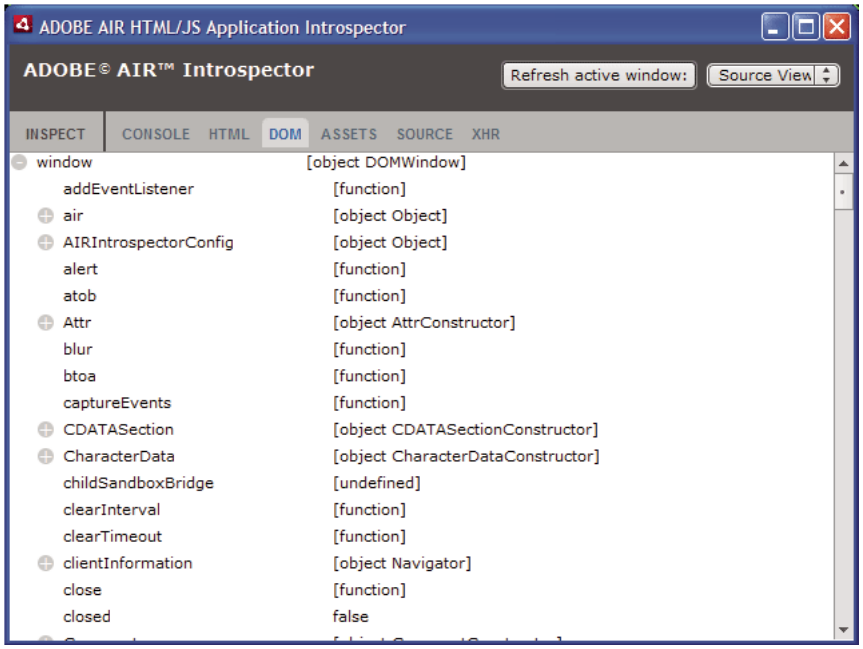
Cliquez sur le bouton Inspect (à gauche de la liste d'onglets de la fenêtre de l'outil AIR HTML Introspector). Vous pouvez cliquer sur tout élément dans la page HTML de la fenêtre principale. L'objet DOM associé est affiché dans l'onglet HTML. Lorsque la fenêtre principale possède le focus, vous pouvez aussi appuyer sur le raccourci clavier pour activer ou désactiver le bouton Inspect. Le raccourci clavier par défaut est F11. Vous pouvez le remplacer par une autre touche (voir « Configuration de l'outil AIR HTML Introspector » à la page 301).

Cliquez sur le bouton Refresh active window (dans la partie supérieure de la fenêtre de l'outil AIR HTML Introspector) pour actualiser les données affichées dans l'onglet HTML.

Appuyez sur Ctrl+F (Windows) ou Commande+F (Mac OS) pour rechercher du texte dans l'onglet. (La recherche ne porte pas sur les nœuds de l'arborescence qui ne sont pas visibles.)

Onglet DOM

L'onglet DOM affiche l'objet Window dans une structure arborescente. Vous pouvez modifier toute propriété de type chaîne ou numérique. La valeur modifiée est répercutée dans l'application.

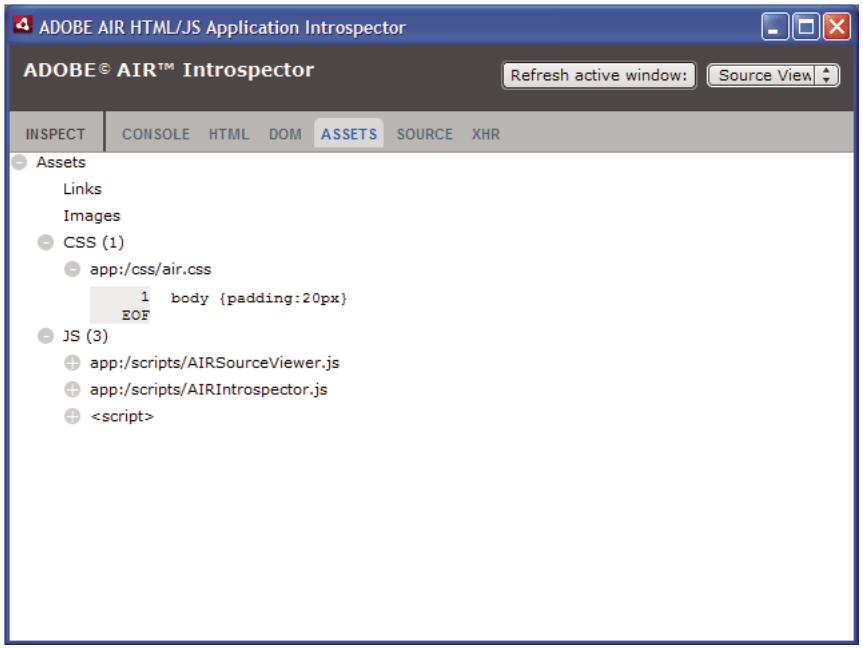


Cliquez sur le bouton Refresh active window (dans la partie supérieure de la fenêtre de l'outil AIR HTML Introspector) pour actualiser les données affichées dans l'onglet DOM.

Appuyez sur Ctrl+F (Windows) ou Commande+F (Mac OS) pour rechercher du texte dans l'onglet. (La recherche ne porte pas sur les nœuds de l'arborescence qui ne sont pas visibles.)

Onglet Assets

L'onglet Assets permet de vérifier les liens, les images et les fichiers CSS et JavaScript chargés dans la fenêtre native. Lorsque vous développez l'un de ces nœuds, le contenu du fichier ou l'image utilisé s'affiche.



Cliquez sur le bouton Refresh active window (dans la partie supérieure de la fenêtre de l'outil AIR HTML Introspector) pour actualiser les données affichées dans l'onglet Assets.

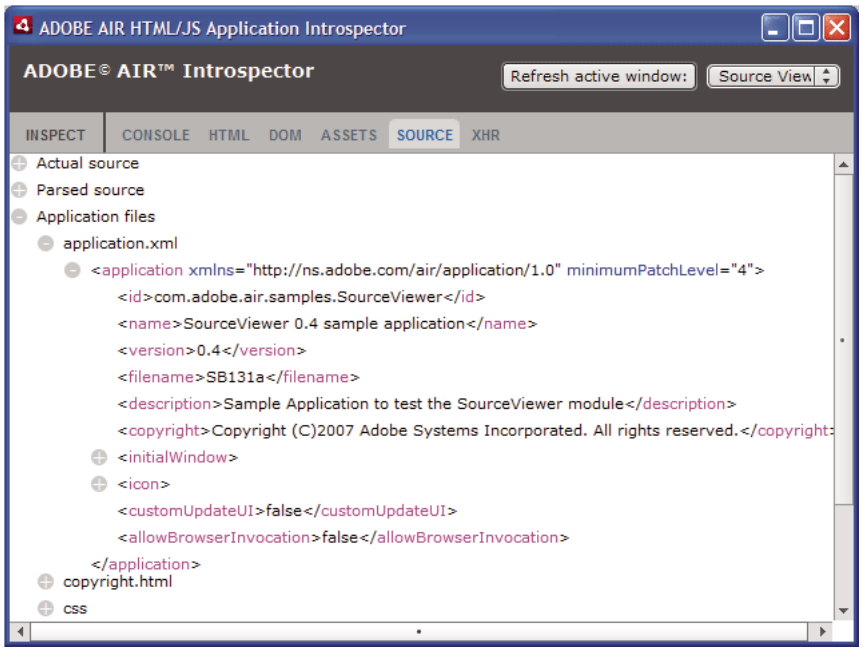
Appuyez sur Ctrl+F (Windows) ou Commande+F (Mac OS) pour rechercher du texte dans l'onglet. (La recherche ne porte pas sur les nœuds de l'arborescence qui ne sont pas visibles.)

Onglet Source

L'onglet Source comporte trois sections :

- Actual source : affiche le code source HTML de la page chargée en tant que contenu racine au démarrage de l'application.
- Parsed source : affiche le code de marquage en cours constituant l'interface utilisateur de l'application, qui peut différer du code source en tant que tel, puisque l'application génère le code de marquage à la volée à l'aide de techniques Ajax.

- Application files : répertorie les fichiers qui figurent dans le répertoire de l'application. Cette liste est uniquement disponible lorsque l'outil AIR HTML Introspector est lancé à partir du sandbox de sécurité de l'application. Dans cette section, vous pouvez afficher le contenu des fichiers de texte ou des images.

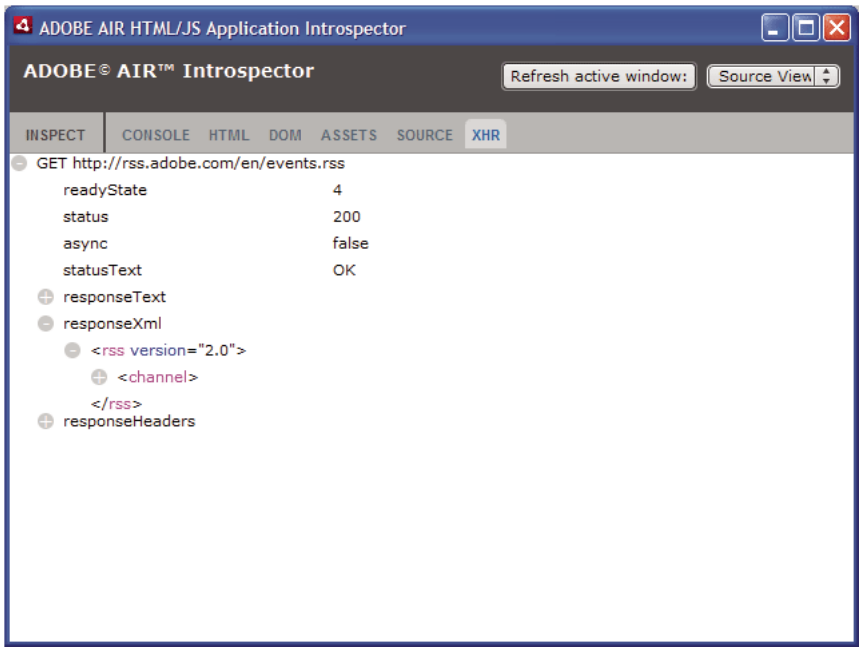


Cliquez sur le bouton Refresh active window (dans la partie supérieure de la fenêtre de l'outil AIR HTML Introspector) pour actualiser les données affichées dans l'onglet Source.

Appuyez sur Ctrl+F (Windows) ou Commande+F (Mac OS) pour rechercher du texte dans l'onglet. (La recherche ne porte pas sur les nœuds de l'arborescence qui ne sont pas visibles.)

Onglet XHR

L'onglet XHR intercepte toutes les requêtes XMLHttpRequest au sein de l'application et enregistre ces informations. Vous pouvez ainsi afficher les propriétés XMLHttpRequest, y compris `responseText` et `responseXML` (le cas échéant) dans une structure arborescente.



Appuyez sur Ctrl+F (Windows) ou Commande+F (Mac OS) pour rechercher du texte dans l'onglet. (La recherche ne porte pas sur les nœuds de l'arborescence qui ne sont pas visibles.)

Utilisation de l'outil AIR HTML Introspector avec du contenu d'un sandbox hors application

Vous pouvez charger du contenu à partir du répertoire de l'application dans un iframe ou une image mappé sur un sandbox hors application (Voir Sécurité HTML dans Adobe AIR (développeurs ActionScript) ou Sécurité HTML dans Adobe AIR (développeurs HTML)). Vous pouvez utiliser l'outil AIR HTML Introspector avec un tel contenu, à condition de respecter les règles suivantes :

- Le fichier AIRIntrospector.js file doit être inclus dans le contenu du sandbox de l'application et du sandbox hors application (iframe).
- Ne remplacez pas la propriété `parentSandboxBridge`. Elle est utilisée par le code de l'outil AIR HTML Introspector. Ajoutez les propriétés lorsqu'elles sont nécessaires. N'utilisez pas la syntaxe suivante, par exemple :

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)}} ;
  ```

  Utilisez plutôt la syntaxe ci-dessous :

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- Il est impossible d'ouvrir la fenêtre de l'outil AIR HTML Introspector à partir du contenu d'un sandbox hors application, en appuyant sur la touche F12 ou en appelant une des méthodes de la classe air.Introspector.Console. La seule façon d'ouvrir cette fenêtre consiste à cliquer sur le bouton Open Introspector, qui est ajouté par défaut dans l'angle supérieur droit de l'iframe ou de l'image. (En raison des restrictions de sécurité appliquées au contenu du sandbox hors application, seule une action de l'utilisateur, telle que cliquer sur un bouton, permet d'ouvrir une nouvelle fenêtre).
- Vous pouvez ouvrir une fenêtre de l'outil AIR HTML Introspector pour le sandbox de l'application et une autre pour le sandbox hors application. Le titre de ces fenêtres permet de les différencier.
- L'onglet Source n'affiche pas les fichiers d'application lorsque l'outil AIR HTML Introspector s'exécute à partir d'un sandbox hors application.
- L'outil AIR HTML Introspector peut uniquement vérifier le code du sandbox à partir duquel il a été ouvert.

Chapitre 20 : Localisation d'applications AIR

Adobe AIR 1.1 et ultérieur

Adobe® AIR® assure une prise en charge multilingue.

Pour une présentation de la procédure de localisation de contenu dans ActionScript 3.0 et la structure Flex, voir « Localisation d'applications » dans le Guide du développeur ActionScript 3.0.

Langues prises en charge dans AIR

La prise en charge de la localisation des applications AIR dans les langues suivantes a été introduite dans AIR 1.1 :

- Chinois simplifié
- Chinois traditionnel
- Français
- Allemand
- Italien
- japonais
- coréen
- Portugais (Brésil)
- Russe
- Espagnol

Les langues suivantes ont été ajoutées dans AIR 1.5 :

- Tchèque
- Néerlandais
- Polonais
- Suédois
- Turc

Voir aussi

Building multilingual Flex applications on Adobe AIR (Création d'applications Flex multilingues dans Adobe AIR)

Building a multilingual HTML-based application (disponible en anglais uniquement)

Localisation du nom et de la description de l'application dans le programme d'installation de l'application d'AIR

Adobe AIR 1.1 et ultérieur

Vous pouvez spécifier plusieurs langues pour les éléments `name` et `description` du fichier descripteur d'application. Par exemple, le code suivant définit le nom de l'application en trois langues (anglais, français et allemand) :

```
<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>
```

L'attribut `xml:lang` de chaque élément text définit un code de langue, comme indiqué dans la norme RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

L'élément name définit le nom de l'application qu'affiche le programme d'installation de l'application AIR. Ce programme utilise la valeur localisée qui correspond le mieux aux langues d'interface utilisateur définies par les paramètres du système d'exploitation.

De même, vous pouvez spécifier des versions en plusieurs langues de l'élément `description` dans le fichier descripteur d'application. Cet élément définit le texte descriptif qu'affiche le programme d'installation de l'application AIR.

Ces paramètres s'appliquent uniquement aux langues disponibles dans le programme d'installation de l'application AIR. Ils ne définissent pas les jeux de paramètres régionaux disponibles pour l'application installée lorsqu'elle s'exécute. L'interface utilisateur des applications AIR peut prendre en charge plusieurs langues, y compris celles gérées par le programme d'installation de l'application AIR.

Pour plus d'informations, voir « Éléments du fichier descripteur d'application AIR » à la page 221.

Voir aussi

Building multilingual Flex applications on Adobe AIR (Création d'applications Flex multilingues dans Adobe AIR)

Building a multilingual HTML-based application (disponible en anglais uniquement)

Localisation de contenu HTML à l'aide de la structure de localisation HTML d'AIR

Adobe AIR 1.1 et ultérieur

Le SDK d'AIR 1.1 comprend une structure de localisation HTML, que définit le fichier JavaScript AIRLocalizer.js. Ce fichier réside dans le répertoire frameworks du SDK d'AIR. Il contient une classe air.Localizer, qui propose des fonctionnalités facilitant la création d'applications prenant en charge plusieurs versions localisées.

Chargement du code de la structure de localisation HTML d'AIR

Pour utiliser la structure de localisation, copiez le fichier AIRLocalizer.js dans votre projet. Incluez-le ensuite dans le fichier HTML principal de l'application, à l'aide d'une balise script :

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

Vous pouvez ensuite appeler l'objet `air.Localizer.localizer` par le biais de code JavaScript :

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

L'objet `air.Localizer.localizer` est un objet singleton qui définit des méthodes et des propriétés relatives à l'utilisation et à la gestion des ressources localisées. La classe Localizer comprend les méthodes suivantes :

Méthode	Description
`getFile()`	Extrait le texte d'un paquet de ressources spécifié pour un jeu de paramètres régionaux spécifique. Voir « Extraction de ressources d'un jeu de paramètres régionaux spécifique » à la page 317.
`getLocaleChain()`	Renvoie les langues qui figurent dans le chaînage de jeux de paramètres régionaux. Voir « Définition du chaînage de jeux de paramètres régionaux » à la page 316
`getResourceBundle()`	Renvoie les clés du regroupement et les valeurs correspondantes sous forme d'objet. Voir « Extraction de ressources d'un jeu de paramètres régionaux spécifique » à la page 317.
`getString()`	Extrait la chaîne définie pour une ressource. Voir « Extraction de ressources d'un jeu de paramètres régionaux spécifique » à la page 317.
`setBundlesDirectory()`	Définit l'emplacement du répertoire de regroupements. Voir « Personnalisation des paramètres de la structure de localisation HTML d'AIR » à la page 315.
`setLocalAttributePrefix()`	Définit le préfixe utilisé par les attributs de localisation utilisés dans des éléments DOM HTML. Voir « Personnalisation des paramètres de la structure de localisation HTML d'AIR » à la page 315
`setLocaleChain()`	Définit l'ordre des langues dans le chaînage de jeux de paramètres régionaux. Voir « Définition du chaînage de jeux de paramètres régionaux » à la page 316
`sortLanguagesByPreference()`	Trie les jeux de paramètres régionaux du chaînage de jeux de paramètres régionaux en fonction des paramètres du système d'exploitation. Voir « Définition du chaînage de jeux de paramètres régionaux » à la page 316
`update()`	Met à jour le DOM HTML (ou un élément DOM) avec les chaînes localisées du chaînage de jeux de paramètres régionaux en cours. Pour plus d'informations sur les chaînages de jeux de paramètres régionaux, voir « Gestion des chaînages de jeux de paramètres régionaux » à la page 313. Pour plus d'informations sur la méthode `update()`, voir « Mise à jour d'éléments DOM afin qu'ils utilisent le jeu de paramètres régionaux en cours » à la page 314.

La classe Localizer comprend les propriétés statiques suivantes :

Propriété	Description
`localizer`	Renvoie une référence à l'objet singleton Localizer de l'application.
`ultimateFallbackLocale`	Jeu de paramètres régionaux utilisé lorsque l'application ne prend en charge aucune préférence utilisateur. Voir « Définition du chaînage de jeux de paramètres régionaux » à la page 316

Spécification des langues prises en charge

Utilisez l'élément `<supportedLanguages>` dans le fichier descripteur de l'application pour identifier les langues prises en charge par l'application. Cet élément est utilisé uniquement par iOS, le moteur d'exécution captif de Mac et les applications Android ; il est ignoré par tous les autres types d'applications.

Si vous ne spécifiez pas l'élément `<supportedLanguages>`, le programme de mise en package exécute par défaut les actions suivantes en fonction du type d'application :

- iOS : les langues prises en charge par le moteur d'exécution d'AIR sont répertoriées dans l'App Store d'iOS comme langues prises en charges dans l'application.
- Moteur d'exécution captif de Mac : toute application mise en package avec un paquet captif ne possède aucune information de localisation.
- Android : le paquet d'application dispose de ressources pour toutes les langues prises en charge par le moteur d'exécution d'AIR.

Pour plus d'informations, voir « supportedLanguages » à la page 252.

Définition de regroupements de ressources

La structure de localisation HTML lit les versions localisées des chaînes dans des fichiers de *localisation*. Un fichier de localisation est un ensemble de valeurs basées sur des clés, sérialisées dans un fichier de texte. Ce type de fichier est parfois appelé un *regroupement*.

Créez dans le répertoire de projet de votre application un sous-répertoire que vous nommerez locale. (Vous pouvez également utiliser un autre nom, voir « Personnalisation des paramètres de la structure de localisation HTML d'AIR » à la page 315.) Ce répertoire contient les fichiers de localisation et est appelé le *répertoire de regroupements*.

Créez un sous-répertoire dans le répertoire de regroupements pour chaque jeu de paramètres régionaux pris en charge par votre application. Attribuez à chaque sous-répertoire un nom correspondant au code du jeu de paramètres régionaux. Par exemple, nommez le répertoire français « fr » et le répertoire anglais « en ». Vous pouvez utiliser un trait de soulignement (_) pour définir un jeu de paramètres régionaux constitué d'une langue et d'un code de pays. Par exemple, nommez le répertoire anglais américain « en_us ». Vous pouvez éventuellement remplacer le trait de soulignement par un tiret (« en-us », par exemple). La structure de localisation HTML prend ces deux formats en charge.

Libre à vous d'ajouter tout nombre de fichiers de ressources à un sous-répertoire de jeu de paramètres régionaux. En règle générale, vous créez un fichier de localisation par langue (et le placez dans le répertoire correspondant à celle-ci). La structure de localisation HTML comprend une méthode `getFile()` qui vous permet de lire le contenu d'un fichier (voir « Extraction de ressources d'un jeu de paramètres régionaux spécifique » à la page 317.

Les fichiers dotés de l'extension .properties constituent des fichiers de propriétés de localisation. Ils permettent de définir les paires clé-valeur d'un jeu de paramètres régionaux. Un fichier de propriétés définit une valeur chaîne sur chaque ligne. Ainsi, l'exemple suivant attribue la valeur chaîne « `Hello in English.` » à une clé nommée `greeting` :

```
greeting=Hello in English.
```

Un fichier de propriétés contenant le texte suivant définit six paires clé-valeur :

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Cet exemple illustre une version anglaise du fichier de propriétés, à stocker dans le répertoire en.

Une version française de ce fichier de propriétés est placée dans le répertoire fr :

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Vous pouvez définir plusieurs fichiers de ressources pour différents types d'information. Imaginons un fichier nommé legal.properties, qui contient un modèle de texte juridique (tel un avis de copyright). Libre à vous de réutiliser cette ressource dans diverses applications. Vous pouvez de même créer plusieurs fichiers définissant du contenu localisé pour différentes parties de l'interface utilisateur.

Utilisez le codage UTF-8 pour ces fichiers afin de prendre en charge plusieurs langues.

Gestion des chaînages de jeux de paramètres régionaux

Lorsque votre application charge le fichier AIRLocalizer.js, il examine les jeux de paramètres régionaux définis dans celle-ci. Ces jeux correspondent aux sous-répertoires du répertoire de regroupements (voir « Définition de regroupements de ressources » à la page 312). La liste des jeux de paramètres régionaux disponibles est appelée *chaînage de jeux de paramètres régionaux*. Le fichier AIRLocalizer.js trie automatiquement le chaînage de jeux de paramètres régionaux en fonction de l'ordre de préférence défini par les paramètres du système d'exploitation. (La propriété `Capabilities.languages` trie les langues d'interface utilisateur du système d'exploitation dans l'ordre de préférence.)

Par conséquent, si une application définit des ressources pour "en", "en_US" et "en_UK", la structure de localisation HTML d'AIR trie le chaînage de jeux de paramètres régionaux de manière appropriée. Lorsqu'une application démarre sur un système qui indique que le jeu de paramètres régionaux principal correspond à "en", le chaînage est trié de la façon suivante : `["en", "en_US", "en_UK"]`. Dans ce cas, l'application recherche des ressources dans le regroupement « en » en premier, puis dans le regroupement "en_US".

Cependant, si le système indique que le jeu de paramètres régionaux principal correspond à "en_US", le tri devient `["en_US", "en", "en_UK"]`. Dans ce cas, l'application recherche des ressources dans le regroupement "en_US" en premier, puis dans le regroupement "en".

Par défaut, l'application définit le premier jeu de paramètres régionaux du chaînage en tant que jeu à utiliser par défaut. Vous pouvez inviter l'utilisateur à sélectionner un jeu de paramètres régionaux lors de la première exécution de l'application. Vous pouvez ensuite choisir de stocker la sélection dans un fichier de préférences et utiliser ce jeu de paramètres régionaux lors de démarrages suivants.

Votre application peut utiliser des chaînes de ressource de tout jeu de paramètres régionaux du chaînage. Si un jeu de paramètres régionaux spécifique ne définit pas une chaîne de ressource, l'application utilise la chaîne de ressource correspondante suivante d'autres jeux de paramètres régionaux du chaînage.

Vous pouvez personnaliser le chaînage de jeux de paramètres régionaux en appelant la méthode `setLocaleChain()` de l'objet Localizer. Voir « Définition du chaînage de jeux de paramètres régionaux » à la page 316

Mise à jour d'éléments DOM avec du contenu localisé

Un élément de l'application peut référencer une valeur de clé dans un fichier de propriétés de localisation. Ainsi, l'élément `title` de l'exemple ci-dessous spécifie un attribut `local_innerHTML`. La structure de localisation utilise cet attribut pour rechercher une valeur localisée. Par défaut, la structure recherche des attributs dont le nom commence par « `local_` ». Elle met à jour les attributs portant un nom correspondant au texte suivant « `local_` ». Dans ce cas, elle définit l'attribut `innerHTML` de l'élément `title`. L'attribut `innerHTML` utilise la valeur définie pour la clé `mainWindowTitle` dans le fichier de propriétés par défaut (default.properties) :

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Si le jeu de paramètres régionaux en cours ne définit pas de valeur correspondante, la structure de localisation examine le reste du chaînage de jeux. Elle utilise le jeu suivant du chaînage pour lequel une valeur est définie.

Dans l'exemple suivant, le texte (attribut `innerHTML`) de l'élément `p` utilise la valeur de la clé `greeting` définie dans le fichier de propriétés par défaut :

```
<p local_innerHTML="default.greeting" />
```

Dans l'exemple suivant, l'attribut value (et le texte affiché) de l'élément `input` utilise la valeur de la clé `btnBlue` définie dans le fichier de propriétés par défaut :

```
<input type="button" local_value="default.btnBlue" />
```

Pour mettre à jour le DOM HTML afin qu'il utilise les chaînes définies dans le chaînage de jeu de paramètres régionaux en cours, appelez la méthode `update()` de l'objet Localizer. L'appel de la méthode `update()` force l'objet Localizer à analyser le DOM et à appliquer des manipulations lorsqu'il détecte des attributs de localisation (« `local_...` ») :

```
air.Localizer.localizer.update();
```

Vous pouvez définir des valeurs pour un attribut (tel que « innerHTML ») et pour l'attribut de localisation correspondant (tel que « local_innerHTML »). Dans ce cas, la structure de localisation ne remplace la valeur de l'attribut que si elle détecte une valeur correspondante dans le chaînage de localisation. Par exemple, l'élément suivant définit les attributs `value` et `local_value` :

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

Vous pouvez aussi vous contenter de mettre à jour un élément DOM spécifique. Pour plus d'informations, voir « Mise à jour d'éléments DOM afin qu'ils utilisent le jeu de paramètres régionaux en cours » à la page 314

Par défaut, la structure de localisation HTML d'AIR applique le préfixe « `local_` » aux attributs définissant des paramètres de localisation pour un élément. Ainsi, un attribut `local_innerHTML` définit par défaut le regroupement et le nom de ressource utilisés pour la valeur `innerHTML` d'un élément. De même, un attribut `local_value` définit par défaut le regroupement et le nom de ressource utilisés pour l'attribut `value` d'un élément. Vous pouvez configurer la structure de localisation de sorte à utiliser un préfixe autre que « `local_` ». Voir « Personnalisation des paramètres de la structure de localisation HTML d'AIR » à la page 315.

Mise à jour d'éléments DOM afin qu'ils utilisent le jeu de paramètres régionaux en cours

Lorsque l'objet Localizer met à jour le DOM HTML, les éléments marqués doivent utiliser des valeurs d'attribut basées sur des chaînes définies dans le chaînage de jeux de paramètres régionaux en cours. Pour que la structure de localisation HTML mette à jour le DOM HTML, appelez la méthode `update()` de l'objet `Localizer` :

```
air.Localizer.localizer.update();
```

Pour mettre à jour un élément DOM spécifique seulement, transmettez-le en tant que paramètre à la méthode `update()`. La méthode `update()` possède un seul paramètre, `parentNode`, qui est facultatif. Lorsqu'il est spécifié, le paramètre `parentNode` définit l'élément DOM à localiser. L'appel de la méthode `update()` en incluant le paramètre `parentNode` définit des valeurs localisées pour tous les éléments enfants qui spécifient des attributs de localisation.

Soit, par exemple, l'élément `div` ci-dessous :

```
<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>
```

Pour mettre cet élément à jour afin qu'il utilise les chaînes localisées définies dans le chaînage de jeu de paramètres régionaux en cours, utilisez le code JavaScript suivant :

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Si le chaînage de jeux de paramètres régionaux ne contient pas de valeur de clé, la structure de localisation définit la valeur de l'attribut sur la valeur de l'attribut « `local_` ». Ainsi, imaginons que dans l'exemple précédent, la structure de localisation n'a pas trouvé de valeur pour la clé `lblColors` (dans aucun des fichiers default.properties du chaînage de jeux de paramètres régionaux). Dans ce cas, elle attribue la valeur `"default.lblColors"` à l'attribut `innerHTML`. L'utilisation de cette valeur indique une absence de ressources (au développeur).

La méthode `update()` distribue un événement `resourceNotFound` lorsqu'une ressource est introuvable dans le chaînage de jeux de paramètres régionaux. La constante `air.Localizer.RESOURCE_NOT_FOUND` définit la chaîne `"resourceNotFound"`. L'événement possède trois propriétés : `bundleName`, `resourceName` et `locale`. La propriété `bundleName` représente le nom du regroupement introuvable. La propriété `resourceName` représente le nom de la ressource introuvable. La propriété `locale` représente le nom du jeu de paramètres régionaux dans lequel la ressource est introuvable.

La méthode `update()` distribue un événement `bundleNotFound` lorsque le regroupement spécifié est introuvable. La constante `air.Localizer.BUNDLE_NOT_FOUND` définit la chaîne `"bundleNotFound"`. L'événement possède deux propriétés : `bundleName` et `locale`. La propriété `bundleName` représente le nom du regroupement introuvable. La propriété `locale` représente le nom du jeu de paramètres régionaux dans lequel la ressource est introuvable.

La propriété `update()` s'exécute en mode asynchrone (et distribue des événements `resourceNotFound` et `bundleNotFound` de manière asynchrone). Le code suivant définit des écouteurs pour les événements `resourceNotFound` et `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);
}
```

Personnalisation des paramètres de la structure de localisation HTML d'AIR

La méthode `setBundlesDirectory()` de l'objet Localizer vous permet de personnaliser le chemin d'accès au répertoire de regroupements. La méthode `setLocalAttributePrefix()` de l'objet Localizer vous permet de personnaliser le chemin d'accès au répertoire de regroupements et la valeur d'attribut utilisée par l'objet.

Le répertoire de regroupements par défaut correspond au sous-répertoire de jeux de paramètres régionaux du répertoire d'application. Vous pouvez spécifier un autre répertoire en appelant la méthode `setBundlesDirectory()` de l'objet Localizer. Cette méthode gère un paramètre unique, `path`, qui représente sous forme de chaîne le chemin d'accès au répertoire de regroupements souhaité. Le paramètre `path` prend en charge les valeurs suivantes :

- Une chaîne définissant un chemin relatif au répertoire d'application, telle que `"locales"`
- Une chaîne définissant une URL valide utilisant le modèle d'URL `app`, `app-storage` ou `file`, telle que `"app://languages"` (n'utilisez *pas* le modèle d'URL `http`)
- Un objet File

Pour plus d'informations sur les URL et les chemins de répertoire, voir :

- Chemin des objets File (développeurs ActionScript)
- Chemin des objets File (développeurs HTML)

Par exemple, le code suivant définit le répertoire de regroupements sur le sous-répertoire languages du répertoire de stockage de l'application (répertoire d'application) :

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

Définissez le paramètre `path` sur un chemin valide. Sinon, la méthode renvoie une exception BundlePathNotFoundError. « `BundlePathNotFoundError` » correspond à la propriété `name` de cette erreur, dont la propriété `message` spécifie le chemin non valide.

Par défaut, la structure de localisation HTML d'AIR applique le préfixe « `local_` » aux attributs définissant des paramètres de localisation pour un élément. Par exemple, l'attribut `local_innerHTML` définit le regroupement et le nom de ressource utilisés pour la valeur `innerHTML` de l'élément `input` suivant :

```
<p local_innerHTML="default.greeting" />
```

La méthode `setLocalAttributePrefix()` de l'objet Localizer vous permet d'utiliser un préfixe d'attribut autre que « `local_` ». Cette méthode statique gère un paramètre unique, qui correspond à la chaîne à utiliser comme préfixe d'attribut. Par exemple, le code suivant force la structure de localisation à utiliser « loc_ » comme préfixe d'attribut :

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Vous pouvez personnaliser le préfixe d'attribut utilisé par la structure de localisation. Vous souhaiterez peut-être personnaliser le préfixe si la valeur par défaut (« `local_` ») crée un conflit avec le nom d'un autre attribut utilisé par le code. Lorsque vous appelez cette méthode, veillez à utiliser des caractères valides pour les attributs HTML. (La valeur ne doit pas contenir d'espaces, par exemple.)

Pour plus d'informations sur l'utilisation d'attributs de localisation dans des éléments HTML, voir « Mise à jour d'éléments DOM avec du contenu localisé » à la page 313.

Les paramètres relatifs au répertoire de regroupements et au préfixe d'attribut ne sont pas persistants d'une session d'application à une autre. Si vous utilisez des paramètres personnalisés, veillez donc à les définir à chaque initiation de l'application.

Définition du chaînage de jeux de paramètres régionaux

Lorsque vous chargez le code AIRLocalizer.js, il définit automatiquement le chaînage de jeux de paramètres régionaux par défaut. Les jeux de paramètres régionaux disponibles dans le répertoire de regroupements et les paramètres de langue du système d'exploitation définissent ce chaînage (pour plus d'informations voir « Gestion des chaînages de jeux de paramètres régionaux » à la page 313).

Vous pouvez modifier le chaînage de jeux de paramètres régionaux en appelant la méthode statique `setLocaleChain()` de l'objet Localizer. Ainsi, vous souhaiterez peut-être appeler cette méthode si l'utilisateur indique qu'il préfère utiliser une langue spécifique. La méthode `setLocaleChain()` gère un paramètre unique, `chain`, qui correspond à un tableau de jeux de paramètres régionaux, tel que `["fr_FR","fr","fr_CA"]`. L'ordre des jeux dans le tableau détermine l'ordre dans lequel la structure recherche ultérieurement des ressources. Si une ressource est absente du premier jeu du chaînage, la structure continue la recherche dans les autres jeux. Si l'argument `chain` est absent, n'est pas un tableau ou est un tableau vide, la fonction échoue et renvoie une exception IllegalArgumentsError.

La méthode statique `getLocaleChain()` de l'objet Localizer renvoie un tableau répertoriant les jeux de paramètres régionaux du chaînage en cours.

Le code suivant lit le chaînage en cours et ajoute deux jeux de paramètres régionaux français au début :

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

La méthode `setLocaleChain()` distribue un événement « `change` » lorsqu'elle met à jour le chaînage de jeux de paramètres régionaux. La constante `air.Localizer.LOCALE_CHANGE` définit la chaîne « `change` ». L'événement possède une propriété unique, `localeChain`, un tableau des codes de jeux de paramètres régionaux que comprend le nouveau chaînage. Le code suivant définit un écouteur pour cet événement :

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
localizer.addEventListener(air.Localizer.LOCALE_CHANGE, changeHandler);
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
function changeHandler(event)
{
    alert(event.localeChain);
}
```

La propriété statique `air.Localizer.ultimateFallbackLocale` représente le jeu de paramètres régionaux utilisé lorsque l'application ne prend pas en charge de préférences utilisateur. La valeur par défaut est `"en"`. Vous pouvez définir un autre jeu de paramètres régionaux, comme illustré ci-dessous :

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Extraction de ressources d'un jeu de paramètres régionaux spécifique

La méthode `getString()` de l'objet Localizer renvoie la chaîne définie pour une ressource dans un jeu de paramètres régionaux spécifique. Lors de l'appel de la méthode, il est inutile de spécifier une valeur `locale`. Dans ce cas, la méthode examine la totalité du chaînage de jeux de paramètres régionaux et renvoie la chaîne du premier jeu contenant le nom de ressource concerné. Cette méthode prend en charge les paramètres suivants :

Paramètre	Description
`bundleName`	Regroupement contenant la ressource. Il s'agit du nom du fichier de propriétés sans l'extension .properties. Par exemple, si ce paramètre est défini sur `"alerts"`, le code Localizer examine les fichiers de localisation appelés alerts.properties.
`resourceName`	Nom de la ressource.
`templateArgs`	Facultatif. Tableau de chaînes servant à remplacer les balises numérotées dans la chaîne de remplacement. Par exemple, soit un appel à la fonction dans lequel le paramètre `templateArgs` est défini sur `["Raúl", "4"]` et la chaîne de ressource correspondante est `"Hello, {0}. You have {1} new messages."`. Dans ce cas, la fonction renvoie `"Hello, Raúl. You have 4 new messages."`. Pour ne pas tenir compte de ce paramètre, transmettez la valeur `null`.
`locale`	Facultatif. Code du jeu de paramètres régionaux (tel que `"en"`, `"en_us"` ou `"fr"`) à utiliser. Si un jeu de paramètres régionaux est indiqué et qu'aucune valeur correspondante n'est trouvée, la méthode ne recherche pas de valeurs dans les autres jeux du chaînage. Si aucun code de jeu de paramètres régionaux n'est spécifié, la fonction renvoie la chaîne du premier jeu du chaînage qui contient une valeur pour le nom de ressource donné.

La structure de localisation peut mettre à jour des attributs DOM HTML marqués. Vous pouvez cependant utiliser les chaînes localisées autrement. Ainsi, vous pouvez utiliser une chaîne dans du contenu HTML généré dynamiquement ou en tant que valeur de paramètre dans un appel de fonction. Par exemple, le code suivant appelle la fonction `alert()` avec la chaîne définie dans la ressource `error114` du fichier de propriété par défaut du jeu de paramètres régionaux fr_FR :

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

La méthode `getString()` distribue un événement `resourceNotFound` si la ressource est introuvable dans le regroupement spécifié. La constante `air.Localizer.RESOURCE_NOT_FOUND` définit la chaîne `"resourceNotFound"`. L'événement possède trois propriétés : `bundleName`, `resourceName` et `locale`. La propriété `bundleName` représente le nom du regroupement introuvable. La propriété `resourceName` représente le nom de la ressource introuvable. La propriété `locale` représente le nom du jeu de paramètres régionaux dans lequel la ressource est introuvable.

La méthode `getString()` distribue un événement `bundleNotFound` lorsque le regroupement spécifié est introuvable. La constante `air.Localizer.BUNDLE_NOT_FOUND` définit la chaîne `"bundleNotFound"`. L'événement possède deux propriétés : `bundleName` et `locale`. La propriété `bundleName` représente le nom du regroupement introuvable. La propriété `locale` représente le nom du jeu de paramètres régionaux dans lequel la ressource est introuvable.

La méthode `getString()` s'exécute en mode asynchrone (et distribue les événements `resourceNotFound` et `bundleNotFound` en mode asynchrone). Le code suivant définit des écouteurs pour les événements `resourceNotFound` et `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);
}
```

La méthode `getResourceBundle()` de l'objet Localizer renvoie le regroupement spécifié pour un jeu de paramètres régionaux donné. La valeur renvoyée de la méthode est un objet dont les propriétés correspondent aux clés du regroupement. (Si l'application ne trouve pas le regroupement spécifié, la méthode renvoie `null`.)

La méthode reconnaît deux paramètres : `locale` et `bundleName`.

Paramètre	Description
`locale`	Jeu de paramètres régionaux (par exemple « `fr` »).
`bundleName`	Nom du regroupement.

Par exemple, le code suivant appelle la méthode `document.write()` pour charger le regroupement par défaut pour le jeu de paramètres régionaux fr. Il appelle ensuite la méthode `document.write()` pour écrire les valeurs des clés `str1` et `str2` dans ce regroupement :

```
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/>");
```

La méthode `getResourceBundle()` distribue un événement `bundleNotFound` lorsque le regroupement spécifié est introuvable. La constante `air.Localizer.BUNDLE_NOT_FOUND` définit la chaîne `"bundleNotFound"`. L'événement possède deux propriétés : `bundleName` et `locale`. La propriété `bundleName` représente le nom du regroupement introuvable. La propriété `locale` représente le nom du jeu de paramètres régionaux dans lequel la ressource est introuvable.

La méthode `getFile()` de l'objet Localizer renvoie le contenu d'un regroupement, sous forme de chaîne, pour un jeu de paramètres régionaux donné. Le fichier de regroupement est lu au format UTF-8. La méthode prend en charge les paramètres suivants :

Paramètre	Description
`resourceFileName`	Nom du fichier de ressource (`"about.html"`, par exemple).

`templateArgs`	Facultatif. Tableau de chaînes servant à remplacer les balises numérotées dans la chaîne de remplacement. Par exemple, soit un appel à la fonction dans lequel le paramètre `templateArgs` est défini sur `["Raúl", "4"]` et le fichier de ressource correspondant contient deux lignes :
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	Dans ce cas, la fonction renvoie une chaîne de deux lignes :
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

`locale`	Code de jeu de paramètres régionaux à utiliser, tel que `"en_GB"`. Si un jeu de paramètres régionaux est indiqué et qu'aucun fichier correspondant n'est trouvé, la méthode n'examine pas les autres jeux du chaînage. Si *aucun* code de jeu de paramètres régionaux n'est spécifié, la fonction renvoie le texte du premier jeu du chaînage comportant un fichier qui correspond à `resourceFileName`.

Par exemple, le code suivant appelle la méthode `document.write()` à l'aide du contenu du fichier about.html du jeu de paramètres régionaux fr :

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

La méthode `getFile()` distribue un événement `fileNotFound` lorsqu'une ressource est introuvable dans le chaînage de jeux de paramètres régionaux. La constante `air.Localizer.FILE_NOT_FOUND` définit la chaîne `"fileNotFound"`. La méthode `getFile()` s'exécute en mode asynchrone (et distribue l'événement `fileNotFound` de manière asynchrone). L'événement possède deux propriétés : `fileName` et `locale`. La propriété `fileName` représente le nom du fichier introuvable. La propriété `locale` représente le nom du jeu de paramètres régionaux dans lequel la ressource est introuvable. Le code suivant définit un écouteur pour cet événement :

```
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);
}
```

Voir aussi

Building a multilingual HTML-based application (disponible en anglais uniquement)

Chapitre 21 : Variables d'environnement path

Le kit de développement d'AIR contient quelques programmes à lancer à partir d'une ligne de commande ou d'un terminal. L'exécution de ces programmes s'avère souvent plus pratique si le chemin d'accès au répertoire bin du kit SDK figure dans la variable d'environnement path.

Les informations ci-après passent en revue la procédure de définition de path sous Windows, Mac et Linux et servent de guide de référence. Etant donné que les configurations d'ordinateur varient considérablement, la procédure ne fonctionne toutefois pas sur chaque système. Si tel est le cas, vous devriez trouver les informations requises dans la documentation du système d'exploitation ou sur Internet.

Définition de PATH sous Linux et Mac OS à l'aide de l'interface de commande Bash

Si vous saisissez une commande dans une fenêtre de terminal, le programme qui lit la saisie et répond en conséquence doit commencer par trouver le programme de commande dans le système de fichiers. L'interface de commande recherche les commandes dans une liste de répertoires stockée dans la variable d'environnement $PATH. Pour vérifier le contenu actuel de la variable d'environnement path, saisissez :

```
echo $PATH
```

Cette commande renvoie une liste de répertoires séparés par un point-virgule, telle que :

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

La procédure a pour objet d'ajouter le chemin d'accès au répertoire bin du kit SDK d'AIR à la liste, afin que l'interface de commande trouve l'outil ADT et l'application ADL. Supposons que le kit SDK d'AIR réside dans `/Users/fred/SDKs/AIR`. La commande suivante ajoute les répertoires requis à la variable d'environnement path :

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Remarque : *si le chemin contient des caractères d'espace blanc, convertissez-les en séquences d'échappement à l'aide d'une barre oblique inversée, comme suit :*

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

Pour vous assurer que la procédure a abouti, vous pouvez utiliser à nouveau la commande `echo` :

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDK
s/android/tools
```

Parfait. Vous devriez maintenant être en mesure de saisir les commandes suivantes et d'obtenir une réponse encourageante :

```
adt -version
```

Si vous avez modifié correctement la variable $PATH, la commande devrait indiquer la version de l'outil ADT.

Un problème demeure. Lors de l'ouverture suivante d'une fenêtre de terminal, vous noterez que les nouvelles entrées ne figurent plus dans le chemin. La commande de définition du chemin doit être exécutée à chaque fois que vous démarrez un nouveau terminal.

Pour résoudre ce problème, vous ajoutez généralement la commande à l'un des scripts de démarrage utilisés par l'interface de commande. Sous Mac OS, vous pouvez créer le fichier, .bash_profile, dans le répertoire ~/nomUtilisateur. Il est alors exécuté à chaque fois que vous ouvrez une nouvelle fenêtre de terminal. Sous Ubuntu, le script de démarrage exécuté à l'ouverture d'une nouvelle fenêtre de terminal correspond à .bashrc. D'autres versions de distribution de Linux et interfaces de commande gèrent des conventions similaires.

Pour ajouter la commande au script de démarrage de l'interface de commande :

1. Accédez au répertoire d'accueil :

   ```
   cd
   ```

2. Le cas échéant, créez le profil de configuration de l'interface de commande et redirigez le texte saisi en fin de fichier à l'aide de « `cat >>` ». Utilisez le fichier adapté au système d'exploitation et à l'interface de commande. Vous disposez par exemple de `.bash_profile` sous Mac OS et de `.bashrc` sous Ubuntu.

   ```
   cat >> .bash_profile
   ```

3. Saisissez le texte à ajouter au fichier :

   ```
   export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
   ```

4. Mettez fin à la redirection du texte en appuyant sur `CTRL-MAJ-D` au clavier.

5. Affichez le fichier pour vous assurer que tout fonctionne correctement :

   ```
   cat .bash_profile
   ```

6. Ouvrez une nouvelle fenêtre de terminal pour vérifier le chemin :

   ```
   echo $PATH
   ```

   Les nouveaux ajouts devraient être recensés.

Si vous créez ultérieurement une nouvelle version de l'un des kits SDK dans d'autres répertoires, veillez à mettre à jour la commande path dans le fichier de configuration. L'interface de commande continue sinon à utiliser l'ancienne version.

Définition de la variable d'environnement path sous Windows

Si vous ouvrez une fenêtre de commande sous Windows, elle hérite des variables d'environnement globales définies dans les propriétés du système. L'une des plus importantes variables correspond à path, c'est-à-dire la liste de répertoires dans lesquels le programme de commande effectue une recherche lorsque vous saisissez le nom du programme à exécuter. Pour vérifier le contenu de la variable d'environnement path lorsque vous utilisez une fenêtre de commande, vous pouvez saisir :

```
set path
```

Une liste de répertoires séparés par un point-virgule, similaire à celle-ci, s'affiche alors :

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

La procédure a pour objet d'ajouter le chemin d'accès au répertoire bin du kit SDK d'AIR à la liste, afin que le programme de commande trouve l'outil ADT et l'application ADL. Si vous avez placé le kit SDK d'AIR dans le répertoire `C:\SDKs\AIR`, procédez comme suit pour ajouter l'entrée appropriée :

1. Ouvrez la boîte de dialogue Propriétés système à partir du Panneau de configuration ou cliquez avec le bouton droit de la souris sur l'icône Poste de travail et, depuis le menu, sélectionnez Propriétés.
2. Sur l'onglet Avancé, cliquez sur le bouton Variables d'environnement.
3. Sélectionnez l'entrée Path dans la section Variables système de la boîte de dialogue Variables d'environnement.
4. Cliquez sur Modifier.
5. Dans la zone Valeur de la variable, faites défiler la liste jusqu'à la fin du texte.
6. Tout à la fin de la valeur actuelle, entrez le texte suivant :

   ```
   ;C:\SDKs\AIR\bin
   ```

7. Cliquez sur OK dans toutes les boîtes de dialogue pour enregistrer la variable path.

Si vous avez précédemment ouvert des fenêtres de commande, notez que leur environnement n'a pas été actualisé. Ouvrez une nouvelle fenêtre de commande et saisissez la commande suivante pour vous assurer que les chemins sont correctement définis :

```
adt -version
```

Si vous modifiez ultérieurement l'emplacement du kit SDK d'AIR ou que vous ajoutez une nouvelle version, veillez à mettre à jour la variable path.