

Budowanie aplikacji dla środowiska ADOBE® AIR®

Informacje prawne

Informacje prawne można znaleźć na stronie http://help.adobe.com/pl_PL/legalnotices/index.html.

Spis treści

Rozdział 1: Informacje o środowisku Adobe AIR

Rozdział 2: Instalowanie środowiska Adobe AIR

Instalowanie środowiska Adobe AIR	3
Usuwanie środowiska Adobe AIR	5
Instalowanie i uruchamianie przykładowych aplikacji dla środowiska AIR	6
Aktualizacje środowiska Adobe AIR	6

Rozdział 3: Praca z interfejsem API środowiska AIR

Klasy języka ActionScript 3.0 charakterystyczne dla środowiska AIR	8
Klasy dostępne w programie Flash Player z funkcjonalnością charakterystyczną dla środowiska AIR	13
Składniki Flex charakterystyczne dla środowiska AIR	16

Rozdział 4: Narzędzia platformy Adobe Flash przeznaczone do tworzenia w AIR

Instalowanie AIR SDK	18
Konfigurowanie Flex SDK	20
Konfigurowanie zewnętrznych zestawów SDK	21

Rozdział 5: Tworzenie pierwszej aplikacji

Tworzenie pierwszej aplikacji AIR środowiska Flex w programie Flash Builder	22
Tworzenie pierwszej aplikacji AIR na komputery stacjonarne za pomocą programu Flash Professional	25
Tworzenie pierwszej aplikacji AIR dla systemu Android w programie Flash Professional	27
Tworzenie pierwszej aplikacji AIR dla systemu iOS	28
Tworzenie pierwszej aplikacji AIR opartej na kodzie HTML w programie Dreamweaver	32
Tworzenie pierwszej aplikacji AIR w formacie HTML za pomocą programu AIR SDK	35
Tworzenie pierwszej aplikacji AIR na komputery stacjonarne za pomocą zestawu SDK środowiska Flex	39
Tworzenie pierwszej aplikacji AIR dla systemu Android za pomocą zestawu SDK środowiska Flex	43

Rozdział 6: Programowanie aplikacji AIR na komputery stacjonarne

Obieg pracy dotyczący programowania aplikacji AIR na komputery stacjonarne	48
Ustawianie właściwości aplikacji na komputery stacjonarne	49
Debugowanie aplikacji AIR na komputery stacjonarne	55
Pakowanie pliku instalacyjnego AIR na komputery stacjonarne	56
Pakowanie instalatora natywnego dla komputerów	59
Tworzenie pakietu z dołączonym środowiskiem wykonawczym dla komputerów	63
Rozpowszechnianie pakietów AIR na komputery stacjonarne	66

Rozdział 7: Programowanie aplikacji AIR dla urządzeń przenośnych

Konfigurowanie środowiska programistycznego	69
Uwagi dotyczące projektowania aplikacji na urządzenia przenośne	70
Obieg pracy tworzenia aplikacji AIR dla urządzeń przenośnych	74
Konfigurowanie właściwości aplikacji dla urządzeń przenośnych	75
Pakowanie aplikacji AIR dla urządzeń przenośnych	99
Debugowanie aplikacji AIR dla urządzeń przenośnych	106

Spis treści

Instalowanie środowiska AIR i aplikacji AIR na urządzeniach przenośnych	115
Aktualizowanie aplikacji AIR dla urządzeń przenośnych	118
Używanie powiadomień w trybie push	119
Rozdział 8: Programowanie aplikacji AIR dla urządzeń telewizyjnych	
Funkcje środowiska AIR dotyczące obsługi telewizorów	128
Uwagi dotyczące projektowania aplikacji AIR dla urządzeń telewizyjnych	131
Etapy programowania aplikacji AIR dla urządzeń telewizyjnych	147
Właściwości deskryptora aplikacji AIR dla urządzeń telewizyjnych	149
Pakowanie pliku AIR aplikacji dla urządzeń telewizyjnych	153
Debugowanie aplikacji AIR dla urządzeń telewizyjnych	154
Rozdział 9: Korzystanie z rozszerzeń natywnych dla środowiska Adobe AIR	
Pliki rozszerzeń natywnych środowiska AIR (ANE)	159
Rozszerzenia natywne a klasa NativeProcess języka ActionScript	160
Rozszerzenia natywne a biblioteki klas ActionScript (pliki SWC)	160
Obsługiwane urządzenia	160
Obsługiwane profile urządzeń	161
Lista zadań dotycząca korzystania z rozszerzenia natywnego	161
Deklarowanie rozszerzenia w pliku deskryptora aplikacji	161
Uwzględnianie pliku ANE w ścieżce biblioteki aplikacji	162
Pakowanie aplikacji korzystającej z rozszerzeń natywnych	163
Rozdział 10: Kompilatory języka ActionScript	
Informacje o narzędziach wiersza poleceń AIR w pakiecie Flex SDK	165
Konfiguracja kompilatora	165
Kompilowanie plików źródłowych MXML i ActionScript dla środowiska AIR	166
Kompilowanie składnika AIR oraz biblioteki kodu AIR (Flex)	168
Rozdział 11: AIR Debug Launcher (ADL)	
Korzystanie z programu ADL	169
Przykłady ADL	172
Kody wyjścia i kody błędów w programie ADL	173
Rozdział 12: Narzędzie ADT	
Polecenia narzędzia ADT	175
Zestawy opcji narzędzia ADT	190
Komunikaty o błędzie programu ADT	195
Zmienne środowiskowe narzędzia ADT	199
Rozdział 13: Podpisywanie aplikacji AIR	
Cyfrowe podpisywanie pliku AIR	201
Tworzenie niepodpisanego pośredniego pliku AIR za pomocą narzędzia ADT	210
Podpisywanie pliku pośredniego AIR za pomocą narzędzia ADT	211
Podpisywanie zaktualizowanej wersji aplikacji AIR	211
Tworzenie certyfikatu samopodpisanego za pomocą narzędzia ADT	215

Rozdział 14: Pliki deskryptora aplikacji AIR

Zmiany deskryptora aplikacji	218
Struktura pliku deskryptora aplikacji	220
Elementy deskryptora aplikacji AIR	221

Rozdział 15: Profile urządzeń

Ograniczanie profili docelowych w pliku deskryptora aplikacji	260
Możliwości różnych profili	260

Rozdział 16: Interfejs API środowiska w przeglądarce — AIR.SWF

Dostosowywanie pliku badge.swf instalacji bezproblemowej	264
Korzystanie z pliku badge.swf w celu zainstalowania aplikacji	264
Ładowanie pliku air.swf	268
Sprawdzanie, czy środowisko wykonawcze jest zainstalowane	268
Sprawdzanie ze strony internetowej, czy aplikacja AIR została zainstalowana	269
Instalowanie aplikacji AIR z przeglądarki	270
Uruchamianie zainstalowanej aplikacji AIR z przeglądarki	271

Rozdział 17: Aktualizowanie aplikacji AIR

Informacje o aktualizowaniu aplikacji	274
Prezentowanie niestandardowego interfejsu aktualizacji aplikacji	276
Pobieranie pliku AIR na komputer użytkownika	276
Sprawdzanie, czy aplikacja została uruchomiona po raz pierwszy	278
Korzystanie z architektury aktualizacji	280

Rozdział 18: Wyświetlanie kodu źródłowego

Ładowanie, konfigurowanie i otwieranie przeglądarki źródła	294
Interfejs użytkownika przeglądarki źródła	297

Rozdział 19: Debugowanie za pomocą introspektora HTML w środowisku AIR

Informacje o introspektorze AIR	298
Ładowanie kodu introspektora AIR	298
Kontrola obiektu na karcie Konsola	299
Konfigurowanie introspektora AIR	301
Interfejs introspektora AIR	301
Korzystanie z introspektora AIR z treścią w nieaplikacyjnym obszarze izolowanym	307

Rozdział 20: Lokalizowanie aplikacji AIR

Lokalizowanie nazwy i opisu aplikacji w instalatorze aplikacji AIR	309
Lokalizowanie treści HTML w strukturze lokalizowania AIR HTML	310

Rozdział 21: Zmienne środowiskowe ścieżek

Ustawianie zmiennej PATH w systemach Linux i Mac OS za pomocą powłoki Bash	320
Ustawianie zmiennej PATH w systemie Windows	321

Rozdział 1: Informacje o środowisku Adobe AIR

Oprogramowanie Adobe® AIR® jest środowiskiem wykonawczym dla różnych systemów operacyjnych i typów ekranów, które umożliwia wykorzystanie umiejętności projektowania stron internetowych w celu tworzenia i projektowania rozbudowanych aplikacji internetowych (RIA, Rich Internet Application) dla komputerów i urządzeń przenośnych. Aplikacje AIR dla komputerów stacjonarnych oraz urządzeń telewizyjnych i przenośnych można opracowywać za pomocą języka ActionScript 3.0, korzystając z technologii Adobe® Flex i Adobe® Flash® (opartej na formacie SWF). Do opracowywania aplikacji AIR dla komputerów stacjonarnych można też używać języków HTML i JavaScript® oraz technologii AJAX (opartej na języku HTML).

Więcej informacji na temat rozpoczynania pracy ze środowiskiem Adobe AIR można znaleźć w serwisie Adobe AIR Developer Connection (<http://www.adobe.com/devnet/air/>).

Tworząc aplikacje dla środowiska AIR, programista może posługiwać się tymi narzędziami, które zna najlepiej, i stosować techniki, które uważa za najdogodniejsze w danej sytuacji. Dzięki obsłudze technologii Flash, Flex, HTML, JavaScript i Ajax możliwe jest stworzenie aplikacji w pełni dostosowanej do konkretnych potrzeb.

Aplikacje można na przykład tworzyć przy użyciu następujących kombinacji technologii:

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

Użytkownicy korzystają z aplikacji AIR w taki sam sposób jak z aplikacji macierzystych. Środowisko wykonawcze jest instalowane jednokrotnie na komputerze lub urządzeniu użytkownika. Następnie możliwe jest instalowanie i uruchamianie aplikacji AIR w taki sam sposób jak w przypadku innych aplikacji dla komputerów stacjonarnych. W systemie iOS nie jest instalowane osobne środowisko wykonawcze AIR. Każda aplikacja AIR w tym systemie jest autonomiczna.

Środowisko wykonawcze udostępnia spójną platformę i architekturę niezależną od systemu operacyjnego. Można w nim instalować aplikacje, eliminując konieczność testowania ich w różnych przeglądarkach w celu zapewnienia spójnego działania i interakcji w różnych środowiskach lokalnych. Zamiast opracowywać aplikację z myślą o konkretnym systemie operacyjnym, programista tworzy aplikację dla środowiska wykonawczego. Taka strategia ma szereg zalet:

- Aplikacje opracowane dla środowiska AIR działają w wielu różnych systemach operacyjnych, bez dodatkowego nakładu pracy ze strony programisty. Środowisko wykonawcze zapewnia spójną i przewidywalną formę prezentacji oraz interakcji we wszystkich obsługiwanych systemach operacyjnych.
- Aplikacje powstają szybciej, ponieważ ich twórcy mogą korzystać z powszechnie stosowanych technologii internetowych i wzorców projektowych. Aplikacje internetowe można obsługiwać również na komputerach bez konieczności poznawania tradycyjnych technik programowania dla tych komputerów lub złożoności kodu natywnego.
- Tworzenie aplikacji jest prostsze niż w językach niższego poziomu, takich jak C i C++. Jeśli nie jest konieczne używanie złożonych, niskopoziomowych wywołań API specyficznych dla systemu operacyjnego.

Podczas opracowywania aplikacji dla środowiska AIR można korzystać z bogatej gamy architektur i interfejsów API:

- interfejsów API specyficznych dla środowiska AIR, udostępnianych przez to środowisko i jego architekturę;
- interfejsów API języka ActionScript, używanych w plikach SWF oraz architekturze Flex (oraz innych bibliotekach i architekturach opartych na języku ActionScript);

- HTML, CSS i JavaScript
- większości architektur Ajax.
- Rozszerzenia natywne dla środowiska Adobe AIR udostępniają interfejsy API języka ActionScript, które umożliwiają programowy dostęp do funkcji specyficznych dla platformy przy użyciu kodu natywnego. Rozszerzenia natywne mogą też zapewnić dostęp do starszego kodu natywnego i do kodu natywnego oferującego większą wydajność.

Środowisko AIR radykalnie zmienia sposób tworzenia, wdrażania i użytkowania aplikacji. Użytkownik zyskuje większą kontrolę nad procesem twórczym. Aplikacje oparte na technologiach Flash, Flex, HTML i AJAX mogą działać na komputerach stacjonarnych oraz na urządzeniach przenośnych i telewizyjnych.

Informacje o zawartości wszystkich nowych aktualizacji środowiska AIR zawiera strona uwag na temat wersji środowiska Adobe AIR (http://www.adobe.com/go/learn_air_relnotes_pl).

Rozdział 2: Instalowanie środowiska Adobe AIR

Środowisko wykonawcze Adobe® AIR® pozwala uruchamiać aplikacje AIR. Środowisko wykonawcze można zainstalować na różne sposoby:

- Instalując je niezależnie od aplikacji AIR.
- Przeprowadzając pierwszą instalację aplikacji AIR za pośrednictwem paska instalacyjnego strony internetowej. (Wówczas zostanie wyświetlony monit o zainstalowanie także środowiska wykonawczego).
- Tworząc własny instalator, który będzie instalować zarówno aplikację, jak i środowisko wykonawcze. Rozpowszechnianie środowiska wykonawczego AIR w ten sposób wymaga zatwierdzenia przez firmę Adobe. Wniosek o zatwierdzenie można złożyć na stronie [Licencjonowanie środowisk wykonawczych firmy Adobe](#). Firma Adobe nie udostępnia narzędzi do utworzenia takiego instalatora. Jest jednak dostępnych wiele zestawów narzędzi innych firm.
- Instalując aplikację AIR z dołączonym środowiskiem wykonawczym AIR. Takie środowisko wykonawcze jest używane tylko przez dołączoną do niego aplikację. Nie służy ono do uruchamiania innych aplikacji AIR. Dołączanie środowiska wykonawczego jest możliwe w systemach Mac i Windows. W systemie iOS wszystkie aplikacje zawierają dołączone środowisko wykonawcze. Od wersji 3.7 środowiska AIR wszystkie aplikacje dla systemu Android zawierają domyślnie dołączone środowisko wykonawcze (choć istnieje opcja używania zewnętrznego środowiska wykonawczego).
- Konfiguruje środowisko programistyczne AIR, takie jak zestaw SDK środowiska AIR, program Adobe® Flash® Builder™ lub zestaw SDK programu Adobe Flex® (który zawiera narzędzia programistyczne AIR wywoływane z wiersza poleceń). Środowisko wykonawcze wchodzące w skład pakietu SDK służy tylko do debugowania aplikacji, a nie do uruchamiania zainstalowanych aplikacji AIR.

Wymagania systemowe dla instalacji środowiska AIR i uruchamiania aplikacji AIR zostały szczegółowo opisane na stronie: [Adobe AIR: wymagania systemowe](http://www.adobe.com/pl/products/air/systemreqs/)(<http://www.adobe.com/pl/products/air/systemreqs/>).

Instalator środowiska wykonawczego oraz instalator aplikacji AIR tworzą pliki dzienników w trakcie instalowania, aktualizowania i usuwania aplikacji AIR lub samego środowiska wykonawczego AIR. Dzięki tym dziennikom można określić przyczyny problemów z instalowaniem. Więcej informacji zawiera artykuł [Dzienniki instalacji](#).

Instalowanie środowiska Adobe AIR

Aby zainstalować lub zaktualizować środowisko wykonawcze, użytkownik musi mieć uprawnienia administracyjne do komputera.

Instalowanie środowiska wykonawczego na komputerze z systemem Windows

- 1 Pobierz plik instalacyjny środowiska wykonawczego ze strony <http://get.adobe.com/air>.
- 2 Kliknij dwukrotnie plik instalacyjny środowiska wykonawczego.
- 3 Postępuj zgodnie z monitami wyświetlanymi w oknie instalacji, aby przeprowadzić instalację.

Instalowanie środowiska wykonawczego na komputerze Mac

- 1 Pobierz plik instalacyjny środowiska wykonawczego ze strony <http://get.adobe.com/air>.

- 2 Kliknij dwukrotnie plik instalacyjny środowiska wykonawczego.
- 3 Postępuj zgodnie z monitami wyświetlanymi w oknie instalacji, aby przeprowadzić instalację.
- 4 Jeśli instalator wyświetli okno dialogowe uwierzytelniania, wprowadź swoją nazwę użytkownika systemu Mac OS oraz hasło.

Instalowanie środowiska wykonawczego na komputerze z systemem Linux

Uwaga: Środowisko AIR 2.7 i nowsze wersje nie są obecnie obsługiwane w systemie Linux. Aplikacje AIR wdrożone w systemie Linux powinny nadal korzystać z zestawu SDK środowiska AIR 2.6.

Korzystanie z instalatora w formacie binarnym:

- 1 Znajdź binarny plik instalacyjny na stronie http://kb2.adobe.com/cps/853/cpsid_85304.html i pobierz go.
- 2 Ustaw uprawnienia do pliku, tak aby można było uruchomić aplikację instalatora. W wierszu poleceń uprawnienia do plików można ustawić za pomocą następującego polecenia:

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

Niektóre wersje systemu Linux umożliwiają ustawianie uprawnień do pliku w oknie dialogowym właściwości otwieranym za pomocą menu kontekstowego.

- 3 Uruchom instalator z wiersza poleceń lub kliknij dwukrotnie plik instalacyjny środowiska wykonawczego.
- 4 Postępuj zgodnie z monitami wyświetlanymi w oknie instalacji, aby przeprowadzić instalację.

Środowisko Adobe AIR jest instalowane jako pakiet rodzimy. Innymi słowy w formacie rpm lub jako dystrybucja oparta na rpm oraz deb w dystrybucji Debian. Obecnie środowisko AIR nie obsługuje żadnych innych formatów pakietów.

Korzystanie z instalatorów pakietów:

- 1 Znajdź plik pakietu AIR na stronie http://kb2.adobe.com/cps/853/cpsid_85304.html. Pobierz pakiet rpm lub Debian — w zależności od tego, który format jest obsługiwany w danym systemie.
- 2 W razie potrzeby kliknij dwukrotnie plik pakietu AIR, aby zainstalować pakiet.

Instalację można także przeprowadzić z wiersza poleceń:

- a W systemie Debian:

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

- b W systemie obsługującym pakiety rpm:

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

W przypadku aktualizacji już zainstalowanej wersji (AIR 1.5.3 lub nowsza):

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

Aby móc instalować aplikacje środowiska AIR 2 i AIR, użytkownik musi mieć uprawnienia administratora w systemie.

Środowisko Adobe AIR jest instalowane w następującym miejscu: /opt/Adobe AIR/Versions/1.0

Środowisko AIR rejestruje typ MIME „application/vnd.adobe.air-application-installer-package+zip”, co oznacza, że pliki z rozszerzeniem .air należą do tego typu MIME i są rejestrowane razem ze środowiskiem wykonawczym AIR.

Instalowanie środowiska wykonawczego na urządzeniu z systemem Android

Najnowszą wersję środowiska wykonawczego AIR można zainstalować za pomocą witryny Android Market.

Aby zainstalować wersję środowiska wykonawczego AIR dla programistów, można użyć łącza na stronie internetowej lub polecenia `-installRuntime` w narzędziu ADT. W danej chwili może być zainstalowana tylko jedna wersja środowiska wykonawczego AIR. Nie można zainstalować zarówno wersji standardowej, jak i wersji dla programistów.

Więcej informacji zawiera sekcja „[Polecenie installRuntime narzędzia ADT](#)” na stronie 188.

Instalowanie środowiska wykonawczego na urządzeniu z systemem iOS

Niezbędny kod środowiska wykonawczego AIR jest dołączany do każdej aplikacji opracowanej dla telefonu iPhone, odtwarzacza iPod Touch oraz tabletu iPad. Nie trzeba instalować oddzielnego składnika ze środowiskiem wykonawczym.

Więcej tematów Pomocy

„[Środowisko AIR for iOS](#)” na stronie 75

Usuwanie środowiska Adobe AIR

Zainstalowane środowisko wykonawcze można usunąć, postępując według poniższych procedur.

Usuwanie środowiska wykonawczego z komputera z systemem Windows

- 1 W menu Start systemu Windows wybierz kolejno polecenia Ustawienia > Panel sterowania.
- 2 W Panelu sterowania otwórz aplet Programy, Programy i funkcje albo Dodaj lub usuń programy (zależnie od używanej wersji systemu Windows).
- 3 Wybierz pozycję „Adobe AIR”, aby usunąć środowisko wykonawcze.
- 4 Kliknij przycisk Zmień/usuń.

Usuwanie środowiska wykonawczego z komputera z systemem Mac

- Kliknij dwukrotnie program „Adobe AIR Uninstaller”, który znajduje się w folderze /Applications/Utilities.

Usuwanie środowiska wykonawczego z komputera z systemem Linux

Wykonaj jedną z następujących czynności:

- Wybierz polecenie „Adobe AIR Uninstaller” z menu Applications.
- Uruchom plik binarny instalatora środowiska AIR z opcją `-uninstall`
- Usuń pakiety środowiska AIR (`adobeair` i `adobecerts`) za pomocą menedżera pakietów.

Usuwanie środowiska wykonawczego z urządzenia z systemem Android

- 1 Otwórz aplikację Settings (Ustawienia) na urządzeniu.
- 2 Dotknij pozycji Adobe AIR w obszarze Applications > Manage Applications (Aplikacje > Zarządzaj aplikacjami).
- 3 Dotknij przycisku Uninstall (Odinstaluj).

Można też użyć polecenia `-uninstallRuntime` w narzędziu ADT. Więcej informacji zawiera sekcja „[Polecenie uninstallRuntime narzędzia ADT](#)” na stronie 189.

Usuwanie dołączonego środowiska wykonawczego

Aby usunąć dołączone środowisko wykonawcze, należy usunąć instalowaną razem z nim aplikację. Dołączone środowiska wykonawcze są używane tylko do uruchamiania instalowanych razem z nimi aplikacji.

Instalowanie i uruchamianie przykładowych aplikacji dla środowiska AIR

Aby móc zainstalować lub zaktualizować aplikację AIR, użytkownik musi mieć uprawnienia administracyjne do komputera.

Dostępne są aplikacje przykładowe, które ilustrują możliwości środowiska AIR. Aplikacje te można pobrać i zainstalować, postępując według poniższych instrukcji:

- 1 Pobierz i uruchom [aplikacje przykładowe dla środowiska AIR](#). Dostępne są zarówno aplikacje skompilowane, jak i ich kod źródłowy.
- 2 Aby pobrać i uruchomić aplikację przykładową, kliknij jej przycisk Install Now (Zainstaluj teraz). Zostanie wyświetlone pytanie o zainstalowanie i uruchomienie aplikacji.
- 3 Jeśli chcesz pobrać aplikacje przykładowe i uruchomić je później, wybierz łącza pobierania. Aplikacje AIR można uruchamiać w dowolnym momencie w następujący sposób:
 - W systemie Windows kliknij dwukrotnie ikonę aplikacji na pulpicie lub wybierz aplikację z menu Start systemu Windows.
 - W systemie Mac OS kliknij dwukrotnie ikonę aplikacji zainstalowaną domyślnie w folderze Applications katalogu użytkownika (Na przykład: Macintosh HD/Users/JoeUser/Applications/).

Uwaga: Aktualizacje tych instrukcji znajdują się w uwagach do wydania środowiska AIR, dostępnych pod adresem: http://www.adobe.com/go/learn_air_relnotes_pl.

Aktualizacje środowiska Adobe AIR

Co pewien czas firma Adobe aktualizuje środowisko Adobe AIR, udostępniając nowe funkcje i poprawki przeznaczone do usuwania mniejszych problemów. Funkcja automatycznego powiadamiania i aktualizowania umożliwia Adobe automatyczne powiadamianie użytkowników o dostępności zaktualizowanej wersji Adobe AIR.

Aktualizacje środowiska Adobe AIR zapewniają jego poprawne działanie i często zawierają istotne zmiany w zakresie zabezpieczeń. Adobe zaleca aktualizowanie do najnowszej wersji Adobe AIR zawsze, gdy nowa wersja jest dostępna, a szczególnie w sytuacjach, gdy aktualizacja obejmuje zabezpieczenia.

Domyślnie podczas uruchamiania aplikacji AIR środowisko wykonawcze sprawdza, czy dostępna jest aktualizacja. Aplikacja sprawdza dostępność aktualizacji po upływie dwóch tygodni od ostatniego sprawdzania. Jeśli aktualizacja jest dostępna, wówczas środowisko AIR pobiera ją w tle.

Użytkownicy mogą wyłączyć funkcję automatycznego aktualizowania, korzystając z aplikacji AIR SettingsManager. Aplikację AIR SettingsManager można pobrać ze strony <http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air>.

Normalny proces instalacji w środowisku Adobe AIR obejmuje nawiązanie połączenia z witryną <http://airinstall.adobe.com> w celu przesłania podstawowych informacji nt. środowiska instalacji, takich jak wersja i język systemu operacyjnego. Te informacje są przesyłane tylko raz podczas instalacji i dzięki nim Adobe może potwierdzić, czy instalacja się powiodła. Nie są gromadzone ani przesyłane żadne informacje, które mogłyby umożliwić identyfikację konkretnych osób.

Aktualizowanie dołączonych środowisk wykonawczych

Jeśli aplikacja jest rozpowszechniana w pakiecie z dołączonym środowiskiem wykonawczym, nie jest ono aktualizowane automatycznie. Dla bezpieczeństwa użytkowników należy monitorować aktualizacje publikowane przez firmę Adobe i aktualizować aplikację przy użyciu nowej wersji środowiska wykonawczego, gdy zostanie opublikowana odpowiednia zmiana dotycząca zabezpieczeń.

Rozdział 3: Praca z interfejsem API środowiska AIR

Środowisko Adobe® AIR® udostępnia funkcje, które nie są dostępne dla treści SWF działających w programie Adobe® Flash® Player.

Dla twórców aplikacji w języku ActionScript 3.0

Interfejs API środowiska AIR jest udokumentowany w następujących dwóch publikacjach:

- [ActionScript 3.0 — Podręcznik programistów](#)
- [Skorowidz języka ActionScript 3.0 dla platformy Adobe Flash](#)

Dla twórców aplikacji opartych na kodzie HTML

Klasy JavaScript przeznaczone do tworzenia aplikacji AIR opartych na języku HTML i dostępne za pośrednictwem pliku AIRAliases.js (zobacz [Dostęp do klas API środowiska AIR z poziomu kodu JavaScript](#)) są udokumentowane w następujących dwóch publikacjach:

- [Podręcznik dla programistów HTML w środowisku Adobe AIR](#)
- [Skorowidz interfejsu API środowiska Adobe AIR dla programistów HTML](#)

Klasy języka ActionScript 3.0 charakterystyczne dla środowiska AIR

Poniższa tabela zawiera klasy środowiska wykonawczego charakterystyczne dla środowiska Adobe AIR. Nie są one dostępne w treści SWF działającej w odtwarzaczu Adobe® Flash® Player w przeglądarce.

Dla programistów HTML

Klasy JavaScript przeznaczone do tworzenia aplikacji AIR opartych na języku HTML i dostępne za pośrednictwem pliku AIRAliases.js są wymienione w [Skorowidzu interfejsu API środowiska Adobe AIR dla programistów HTML](#).

Klasa	Pakiet języka ActionScript 3.0	Dodane w następującej wersji środowiska 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

Klasa	Pakiet języka ActionScript 3.0	Dodane w następującej wersji środowiska AIR
CameraPosition	flash.media	3.0
CameraRoll	flash.media	2.0
CameraRollBrowseOptions	flash.media	3.0
CameraUI	flash.media	2.5
CertificateStatus	flash.security	2.0
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 i starsze wersje; wycofano w wersji 3.7
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 i starsze wersje; wycofano w wersji 3.7
GameInputHand	flash.ui	3.6 i starsze wersje; wycofano w wersji 3.7
Geolocation	flash.sensors	2.0

Klasa	Pakiet języka ActionScript 3.0	Dodane w następującej wersji środowiska AIR
GeolocationEvent	flash.events	2.0
HTMLHistoryItem	flash.html	1.0
HTMLHost	flash.html	1.0
HTMLLoader	flash.html	1.0
HTMLPDFCapability	flash.html	1.0
HTMLSWFCapability	flash.html	2.0
HTMLUncaughtScriptExceptionEvent	flash.events	1.0
HTMLWindowCreateOptions	flash.html	1.0
Icon	flash.desktop	1.0
IFilePromise	flash.desktop	2.0
ImageDecodingPolicy	flash.system	2.6
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

Klasa	Pakiet języka ActionScript 3.0	Dodane w następującej wersji środowiska AIR
NativeWindowDisplayState	flash.display	1.0
NativeWindowDisplayStateEvent	flash.events	1.0
NativeWindowInitOptions	flash.display	1.0
NativeWindowRenderMode	flash.display	3.0
NativeWindowResize	flash.display	1.0
NativeWindowSystemChrome	flash.display	1.0
NativeWindowType	flash.display	1.0
NetworkInfo	flash.net	2.0
NetworkInterface	flash.net	2.0
NotificationType	flash.desktop	1.0
OutputProgressEvent	flash.events	1.0
PaperSize	flash.printing	2.0
PrintMethod	flash.printing	2.0
PrintUIOptions	flash.printing	2.0
PTRRecord	flash.net.dns	2.0
ReferencesValidationSetting	flash.security	1.0
ResourceRecord	flash.net.dns	2.0
RevocationCheckSettings	flash.security	1.0
Screen	flash.display	1.0
ScreenMouseEvent	flash.events	1.0
SecureSocket	flash.net	2.0
SecureSocketMonitor	air.net	2.0
ServerSocket	flash.net	2.0
ServerSocketConnectEvent	flash.events	2.0
ServiceMonitor	air.net	1.0
SignatureStatus	flash.security	1.0
SignerTrustSettings	flash.security	1.0
SocketMonitor	air.net	1.0
SoftKeyboardType	flash.text	3.0
SQLCollationType	flash.data	1.0
SQLColumnNameStyle	flash.data	1.0
SQLColumnSchema	flash.data	1.0
SQLConnection	flash.data	1.0
SQLException	flash.errors	1.0

Klasa	Pakiet języka ActionScript 3.0	Dodane w następującej wersji środowiska AIR
SQLExceptionEvent	flash.events	1.0
SQLExceptionOperation	flash.errors	1.0
SQLEvent	flash.events	1.0
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

Klasa	Pakiet języka ActionScript 3.0	Dodane w następującej wersji środowiska AIR
URLMonitor	air.net	1.0
URLRequestDefaults	flash.net	1.0
XMLSignatureValidator	flash.security	1.0

Klasy dostępne w programie Flash Player z funkcjonalnością charakterystyczną dla środowiska AIR

Następujące klasy są dostępne dla zawartości SWF uruchomionej w przeglądarce, AIR udostępnia jednak dodatkowe właściwości i metody:

Pakiet	Klasa	Właściwość, metoda lub zdarzenie	Dodane w następującej wersji środowiska 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
		Zdarzenie orientationChange	2.0
		Zdarzenie 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

Pakiet	Klasa	Właściwość, metoda lub zdarzenie	Dodane w następującej wersji środowiska AIR
flash.events	Event	CLOSING	1.0
		DISPLAYING	1.0
		PREPARING	2.6
		EXITING	1.0
		HTML_BOUNDS_CHANGE	1.0
		HTML_DOM_INITIALIZE	1.0
		HTML_RENDER	1.0
		LOCATION_CHANGE	1.0
		NETWORK_CHANGE	1.0
		STANDARD_ERROR_CLOSE	2.0
		STANDARD_INPUT_CLOSE	2.0
		STANDARD_OUTPUT_CLOSE	2.0
		USER_IDLE	1.0
		USER_PRESENT	1.0
		HTTPStatusEvent	HTTP_RESPONSE_STATUS
	responseHeaders		1.0
	responseURL		1.0
	KeyboardEvent	commandKey	1.0
		controlKey	1.0

Pakiet	Klasa	Właściwość, metoda lub zdarzenie	Dodane w następującej wersji środowiska AIR
flash.net	FileReference	extension	1.0
		Zdarzenie httpResponseStatus	1.0
		uploadUnencoded()	1.0
	NetStream	Zdarzenie drmAuthenticate	1.0
		Zdarzenie 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	httpResponseStatus event	1.0	

Pakiet	Klasa	Właściwość, metoda lub zdarzenie	Dodane w następującej wersji środowiska 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

Większość nowych właściwości lub metod jest dostępna wyłącznie dla treści w obszarze izolowanym w aplikacji AIR. Jednak nowe elementy w klasie URLRequest są również dostępne dla treści działającej w innych obszarach izolowanych.

Metody `ByteArray.compress()` i `ByteArray.uncompress()` zawierają nowy parametr `algorithm`, który umożliwia wybieranie kompresji deflate i zlib. Ten parametr jest dostępny tylko dla treści działającej w środowisku AIR.

Składniki Flex charakterystyczne dla środowiska AIR

Następujące komponenty Adobe® Flex™ MX są dostępne podczas tworzenia treści w środowisku Adobe AIR:

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

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

Ponadto Flex 4 zawiera następujące komponenty AIR spark:

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

Więcej informacji na temat komponentów AIR Flex zawiera strona [Korzystanie z komponentów AIR Flex](#).

Rozdział 4: Narzędzia platformy Adobe Flash przeznaczone do tworzenia w AIR

Następujące narzędzia programistyczne platformy służą do tworzenia aplikacji AIR.

Dla programistów ActionScript 3.0 (Flash i Flex):

- Adobe Flash Professional (zobacz [Publikowanie dla środowiska AIR](#))
- Zestawy SDK Adobe Flex 3.x i 4.x (zobacz „[Konfigurowanie Flex SDK](#)” na stronie 20 i „[Narzędzie ADT](#)” na stronie 175).
- Adobe Flash Builder (zobacz [Tworzenie aplikacji AIR za pomocą programu Flex Builder](#))

Dla programistów HTML i Ajax:

- Adobe AIR SDK (zobacz „[Instalowanie AIR SDK](#)” na stronie 18 i „[Narzędzie ADT](#)” na stronie 175).
- Adobe Dreamweaver CS3, CS4, CS5 (informacje: [Rozszerzenie środowiska AIR dla programu Dreamweaver](#))

Instalowanie AIR SDK

Pakiet Adobe AIR SDK zawiera następujące narzędzia wiersza poleceń, za pomocą których można uruchamiać i pakować aplikacje:

AIR Debug Launcher (ADL) Umożliwia uruchamianie aplikacji AIR bez konieczności ich instalowania. Więcej informacji zawiera sekcja „[AIR Debug Launcher \(ADL\)](#)” na stronie 169.

AIR Development Tool (ADT) Pakuje aplikacje AIR do postaci pakietów instalacyjnych przeznaczonych do dystrybucji. Więcej informacji zawiera sekcja „[Narzędzie ADT](#)” na stronie 175.

W celu korzystania z narzędzi wiersza poleceń AIR należy zainstalować na komputerze środowisko Java. Używna maszyna wirtualna Java może pochodzić z pakietu JRE lub JDK (wersja 1.5 lub nowsza). Pakiety Java JRE i Java JDK są dostępne na stronie <http://java.sun.com/>.

W celu uruchomienia narzędzia ADT komputer musi udostępniać co najmniej 2 GB pamięci.

***Uwaga:** Środowisko Java nie jest wymagane do tego, aby użytkownicy końcowi mogli uruchamiać aplikacje AIR.*

Skrócony przegląd procesu tworzenia aplikacji AIR przy użyciu pakietu AIR SDK zawiera sekcja „[Tworzenie pierwszej aplikacji AIR w formacie HTML za pomocą programu AIR SDK](#)” na stronie 35.

Pobieranie i instalowanie AIR SDK

W celu pobrania i zainstalowania AIR SDK należy wykonać poniższe czynności:

Instalowanie pakietu AIR SDK w systemie Windows

- Pobierz plik instalacyjny AIR SDK.
- Pakiet AIR SDK jest dystrybuowany jako standardowe archiwum plików. W celu zainstalowania środowiska AIR należy wyodrębnić zawartość pakietu SDK do folderu na komputerze (na przykład: C:\Program Files\Adobe\AIRSDK lub C:\AIRSDK).

- Narzędzia ADL i ADT są zawarte w folderze bin w pakiecie AIR SDK; ścieżkę do tego folderu należy dodać do zmiennej środowiskowej PATH.

Instalowanie pakietu AIR SDK w systemie Mac OS X

- Pobierz plik instalacyjny AIR SDK.
- Pakiet AIR SDK jest dystrybuowany jako standardowe archiwum plików. W celu zainstalowania środowiska AIR wyodrębnij zawartość pakietu SDK do folderu na komputerze (na przykład: /Users/<nazwa_użytkownika>/Applications/AIRSDK).
- Narzędzia ADL i ADT są zawarte w folderze bin w zestawie SDK środowiska AIR. Ścieżkę do tego folderu należy dodać do zmiennej środowiskowej PATH.

Instalowanie pakietu AIR SDK w systemie Linux

- Pakiet SDK jest dostępny w formacie tbz2.
- W celu zainstalowania pakietu SDK utwórz folder, w którym pakiet SDK zostanie rozpakowany, a następnie użyj następującego polecenia: tar -jxvf <ścieżka do pliku AIR-SDK.tbz2>

Informacje na temat pierwszych kroków w korzystaniu z narzędzi pakietu AIR SDK zawiera sekcja Tworzenie aplikacji AIR za pomocą narzędzi wiersza poleceń.

Co zawiera pakiet AIR SDK

W poniższej tabeli przedstawiono przeznaczenie plików zawartych w pakiecie AIR SDK:

Folder pakietu SDK	Opis plików/narzędzi
bin	Program AIR Debug Launcher (ADL) umożliwia uruchamianie aplikacji AIR bez konieczności tworzenia pakietów i instalowania. Informacje na temat korzystania z tego narzędzia zawiera sekcja „AIR Debug Launcher (ADL)” na stronie 169. Program AIR Developer Tool (ADT) tworzy pakiet z aplikacją jako plik AIR przeznaczony do dystrybucji. Informacje na temat korzystania z tego narzędzia zawiera sekcja „Narzędzie ADT” na stronie 175.
frameworks	Katalog libs zawiera biblioteki kodu przeznaczone do używania w aplikacjach AIR. Katalog projects zawiera kod dla skompilowanych bibliotek SWF i SWC.
include	Katalog include zawiera plik nagłówka w języku C służący do pisania rozszerzeń natywnych.
install	Katalog install zawiera sterowniki USB systemu Windows dla urządzeń z systemem Android. Są to sterowniki dostarczane przez firmę Google w zestawie SDK systemu Android.
lib	Zawiera pomocniczy kod dla narzędzi zestawu SDK środowiska AIR.

Folder pakietu SDK	Opis plików/narzędzi
runtimes	Środowiska wykonawcze AIR dla komputerów stacjonarnych i urządzeń przenośnych. Środowisko wykonawcze dla komputerów stacjonarnych jest używane przez program ADL w celu uruchamiania aplikacji AIR, zanim zostaną umieszczone w pakiecie lub zainstalowane. Środowiska wykonawcze AIR dla systemu Android (pakiety APK) można zainstalować na urządzeniach z systemem Android lub w emulatorach w celu programowania i testowania. Na potrzeby urządzeń i emulatorów są używane oddzielne pakiety APK. Publiczne środowisko wykonawcze AIR dla systemu Android jest dostępne w witrynie Android Market.
samples	Ten folder zawiera przykładowy plik deskryptora aplikacji, przykład użycia funkcji łatwej instalacji (plik badge.swf) oraz domyślne ikony aplikacji AIR.
templates	descriptor-template.xml — Szablon pliku deskryptora aplikacji, który jest wymagany dla każdej aplikacji AIR. Szczegółowy opis pliku deskryptora aplikacji zawiera rozdział „Pliki deskryptora aplikacji AIR” na stronie 217. Ten folder zawiera też pliki schematów dotyczące struktury XML deskryptora aplikacji dla każdej standardowej wersji środowiska AIR.

Konfigurowanie Flex SDK

W przypadku tworzenia aplikacji Adobe® AIR® w środowisku Adobe® Flex™ istnieją następujące możliwości:

- Można pobrać i zainstalować program Adobe® Flash® Builder™, który udostępni zintegrowane narzędzia przeznaczone do tworzenia projektów Adobe AIR, a także do testowania, debugowania i pakowania aplikacji AIR. Zobacz „[Tworzenie pierwszej aplikacji AIR środowiska Flex w programie Flash Builder](#)” na stronie 22.
- Można pobrać pakiet Adobe® Flex™ SDK i tworzyć aplikacje Flex AIR w ulubionym edytorze tekstu oraz za pomocą narzędzi wiersza poleceń.

Skrócony przegląd procesu tworzenia aplikacji AIR przy użyciu zestawu SDK środowiska Flex zawiera sekcja „[Tworzenie pierwszej aplikacji AIR na komputery stacjonarne za pomocą zestawu SDK środowiska Flex](#)” na stronie 39.

Instalowanie Flex SDK

W celu tworzenia aplikacji AIR za pomocą narzędzi wiersza poleceń należy zainstalować na komputerze środowisko Java. Maszyna wirtualna Java może być używana z pakietu JRE lub JDK (wersja 1.5 lub nowsza). Pakiety Java JRE i JDK są dostępne na stronie <http://java.sun.com/>.

Uwaga: Środowisko Java nie jest wymagane do tego, aby użytkownicy końcowi mogli uruchamiać aplikacje AIR.

Pakiet Flex SDK udostępnia interfejs API AIR oraz narzędzia wiersza poleceń używane do pakowania, kompilowania i debugowania aplikacji AIR.

- 1 W razie potrzeby pakiet Flex SDK można pobrać pod adresem <http://opensource.adobe.com/wiki/display/flexsdk/Downloads>.
- 2 Umieść zawartość pakietu SDK w folderze (na przykład: Flex SDK).
- 3 Skopiuj zawartość zestawu SDK środowiska AIR do lokalizacji plików zestawu SDK środowiska Flex.

***Uwaga:** W przypadku komputera Mac upewnij się, że są kopiowane lub zastępowane poszczególne pliki w folderach zestawu SDK, a nie całe katalogi. Kopiowanie katalogu na komputerze Mac do katalogu o tej samej nazwie powoduje domyślnie usunięcie istniejących plików w katalogu docelowym, a nie scalenie zawartości tych dwóch katalogów. Można użyć polecenia `ditto` w oknie terminala, aby scalić zestaw SDK środowiska AIR z zestawem SDK środowiska Flex: `ditto air_sdk_folder flex_sdk_folder`*

4 Programy narzędziowe wiersza polecenia dla środowiska AIR znajdują się w folderze bin.

Konfigurowanie zewnętrznych zestawów SDK

W celu opracowywania aplikacji dla systemów Android i iOS należy pobrać pliki obsługi, zestawy SDK lub inne narzędzia programistyczne udostępniane przez twórców platformy.

Informacje o pobieraniu i instalowaniu zestawu SDK systemu Android zawiera artykuł [Programiści systemu Android: Instalowanie zestawu SDK](#). Począwszy od wersji 2.6 środowiska AIR, nie trzeba pobierać zestawu SDK systemu Android. Zestaw SDK środowiska AIR obecnie zawiera podstawowe składniki potrzebne do instalowania i uruchamiania pakietów APK. Mimo to zestaw SDK systemu Android może być przydatny w przypadku różnych zadań programistycznych, takich jak opracowywanie i uruchamianie emulatorów programowych oraz tworzenie zrzutów ekranu urządzeń.

Zewnętrzny zestaw SDK nie jest potrzebny do opracowywania oprogramowania dla systemu iOS. Są jednak wymagane specjalne certyfikaty i profile obsługi. Więcej informacji zawiera artykuł [Uzyskiwanie plików dla programistów od firmy Apple](#).

Rozdział 5: Tworzenie pierwszej aplikacji

Tworzenie pierwszej aplikacji AIR środowiska Flex w programie Flash Builder

W celu uzyskania szybkich i niezawodnych instrukcji na temat działania środowiska Adobe® AIR® można wykorzystać poniższe instrukcje, aby utworzyć i spakować prostą aplikację AIR „Hello World” opartą na pliku SWF za pomocą programu Adobe® Flash® Builder.

Przed przystąpieniem do pracy należy pobrać i zainstalować program Flex Builder. Należy również pobrać i zainstalować najnowszą wersję środowiska Adobe AIR, która jest dostępna na stronie www.adobe.com/go/air_pl.

Tworzenie projektu AIR

Program Flash Builder zawiera narzędzia służące do projektowania i pakowania aplikacji AIR.

Tworzenie aplikacji AIR w programie Flash Builder lub Flex Builder rozpoczyna się tak samo, jak tworzenie innych projektów aplikacji Flex — od zdefiniowania nowego projektu.

- 1 Otwórz program Flash Builder.
- 2 Wybierz opcję Plik > Nowe > Projekt Flex.
- 3 Wprowadź nazwę projektu: AIRHelloWorld.
- 4 W środowisku Flex aplikacje AIR są traktowane jako typ aplikacji. Dostępne są dwa typy:
 - Aplikacja internetowa działająca w programie Adobe® Flash® Player
 - Aplikacja komputerowa działająca w środowisku Adobe AIR

Jako typ aplikacji wybierz: Komputerowa.

- 5 Aby utworzyć projekt, kliknij przycisk Zakończ.

Projekty AIR początkowo zawierają dwa pliki: główny plik MXML oraz plik XML aplikacji (określany jako plik deskryptora aplikacji). Drugi plik określa właściwości aplikacji.

Więcej informacji zawiera artykuł [Programowanie aplikacji AIR za pomocą programu Flash Builder](#).

Pisanie kodu aplikacji AIR

W celu napisania kodu aplikacji „Hello World” należy przeprowadzić edycję pliku MXML aplikacji (AIRHelloWorld.mxml), który jest otwarty w edytorze. (Jeśli plik nie jest otwarty, należy użyć nawigatora projektów, aby otworzyć ten plik).

Aplikacje Flex AIR na komputerze stacjonarnym są zawarte w znaczniku MXML WindowedApplication. Znacznik MXML WindowedApplication tworzy zwykle okno, które zawiera podstawowe elementy sterujące, takie jak pasek tytułu i przycisk zamykania.

- 1 Dodaj atrybut `title` do składnika WindowedApplication, a następnie przypisz do niego wartość `"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 Dodaj do aplikacji składnik Label (umieść go w znaczniku WindowedApplication). Ustaw właściwość text składnika Label na "Hello AIR" i ustaw ograniczenia układu w taki sposób, aby etykieta znajdowała się na środku:

```
<?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 Następujący blok stylu dodaj bezpośrednio po otwarciu znacznika WindowedApplication i przed znacznikiem składnika label, który właśnie został wprowadzony:

```
<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>
```

Ustawienia stylu mają zastosowanie do całej aplikacji i renderują dla tła okna częściowo przepuszczalny szary kolor.

Kod aplikacji wygląda, jak poniżej:

```
<?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>
```

Następnie wymagana jest zmiana niektórych ustawień w deskrytorze aplikacji, dzięki czemu aplikacja będzie mogła stać się przezroczysta.


- 1 W panelu nawigatora Flex znajdź plik deskryptora aplikacji w katalogu źródłowym projektu: Jeśli projekt ma nazwę AIRHelloWorld, ten plik ma nazwę AIRHelloWorld-app.xml.
- 2 Kliknij dwukrotnie plik deskryptora aplikacji przeznaczony do edycji w programie Flash Builder.
- 3 W kodzie XML odszukaj wiersze oznaczone jako komentarz dla właściwości `systemChrome` i `transparent` (właściwości `initialWindow`). Usuń znaki komentarza. (Usuń znaki „`<!--`” i „`-->`”, tj. ograniczniki komentarzy.)
- 4 Ustaw wartość tekstu właściwości `systemChrome` na `none`, jak poniżej:

```
<systemChrome>none</systemChrome>
```
- 5 Ustaw wartość tekstu właściwości `transparent` na `true`, jak poniżej:

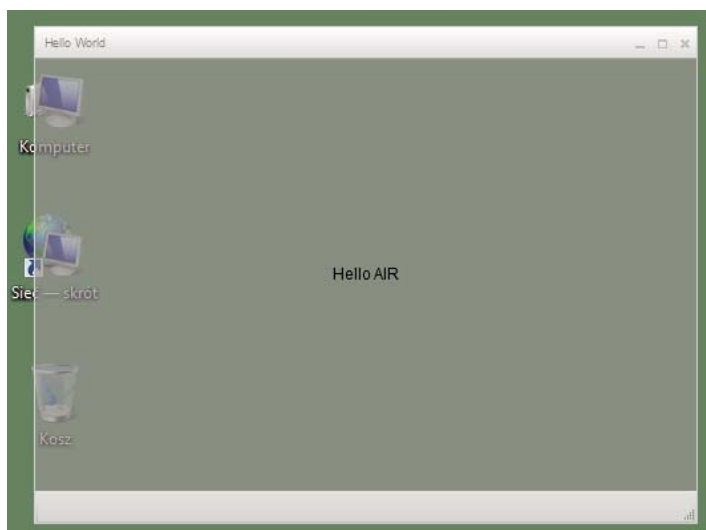
```
<transparent>true</transparent>
```
- 6 Zapisz plik.

Testowanie aplikacji AIR

Aby przetestować napisany kod aplikacji, należy uruchomić tryb debugowania.

- 1 Kliknij przycisk Debuguj  na głównym pasku narzędziowym programu. Można również wybrać polecenia Uruchom > Debuguj > AIRHelloWorld.

Powstała aplikacja AIR powinna przypominać następujący przykład.



- 2 Za pomocą właściwości `horizontalCenter` i `verticalCenter` elementu sterującego Label można umieścić tekst na środku okna. Okno można przesuwając i powiększać/zmniejszać tak samo jak okno każdej innej aplikacji.

Uwaga: Jeśli skompilowanie aplikacji nie jest możliwe, należy usunąć błędy składniowe i błędy pisowni, które przypadkowo znalazły się w kodzie. Błędy i ostrzeżenia są opisane w widoku Problemy w programie Flash Builder.

Pakowanie, podpisywanie i uruchamianie aplikacji AIR

Po wykonaniu powyższych czynności można spakować aplikację „Hello World” do pliku AIR w celu dystrybucji. Plik AIR jest plikiem archiwum, który zawiera pliki aplikacji — są to wszystkie pliki zawarte w folderze bin projektu. W tym prostym przykładzie są to pliki SWF oraz pliki XML aplikacji. Pakiet AIR jest dystrybuowany do użytkowników, którzy wykorzystają go w celu zainstalowania aplikacji. Wymaganym etapem tego procesu jest cyfrowe podpisanie pakietu.

- 1 Upewnij się, że podczas kompilacji nie pojawiają się błędy i że aplikacja działa zgodnie z oczekiwaniami.
- 2 Wybierz opcję Projekt > Eksportuj kompilację do publikacji.
- 3 Sprawdź, czy podano wartość AIRHelloWorld jako projekt i wartość AIRHelloWorld.mxml jako aplikację.
- 4 Wybierz opcję Eksportuj jako podpisany pakiet AIR. Następnie kliknij przycisk Dalej.
- 5 Jeśli istnieje certyfikat cyfrowy, kliknij przycisk Przeglądaj, aby go odszukać, a następnie wybierz plik.
- 6 Jeśli wymagane jest utworzenie nowego samopodpisanego certyfikatu cyfrowego, wybierz opcję Utwórz.
- 7 Wprowadź wymagane informacje i kliknij przycisk OK.
- 8 Kliknij przycisk Zakończ, aby wygenerować pakiet AIR, który będzie miał nazwę AIRHelloWorld.air.

Po wykonaniu tych czynności aplikację można zainstalować i uruchomić z poziomu nawigatora projektu w programie Flash Builder lub z poziomu systemu plików przez dwukrotne kliknięcie pliku AIR.

Tworzenie pierwszej aplikacji AIR na komputery stacjonarne za pomocą programu Flash Professional

W niniejszej sekcji przedstawiono krótką, interaktywną prezentację sposobu działania programu Adobe® AIR® — należy postępować wg instrukcji w niniejszym temacie w celu utworzenia i spakowania prostej aplikacji AIR „Hello World” za pomocą programu Adobe® Flash® Professional.

Przed przystąpieniem do wykonywania tej procedury należy pobrać i zainstalować środowisko Adobe AIR, które jest dostępne na stronie www.adobe.com/go/air_pl.

Tworzenie aplikacji Hello World w programie Flash

Tworzenie aplikacji Adobe AIR w programie Flash przypomina tworzenie każdego innego pliku FLA. Poniższa procedura zawiera wytyczne dotyczące procesu tworzenia prostej aplikacji Hello World w programie Flash Professional.

Aby utworzyć aplikację Hello World

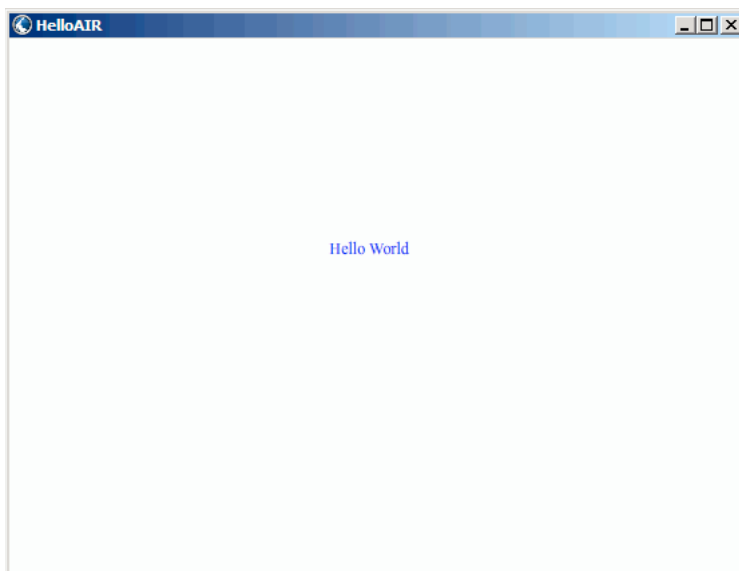
- 1 Uruchom program Flash.
- 2 Na ekranie powitalnym kliknij przycisk AIR, aby utworzyć pusty plik FLA z ustawieniami publikowania Adobe AIR.
- 3 Wybierz narzędzie Tekst w panelu Narzędzia, a następnie utwórz pole tekstu statycznego (domyślnie) w środku stołu montażowego. Rozszerz pole na tyle, aby mogło zawierać od 15 do 20 znaków.
- 4 Do pola tekstowego wprowadź tekst „Hello World”.
- 5 Zapisz plik, nadając mu nazwę (na przykład helloAIR).

Testowanie aplikacji

- 1 Naciśnij klawisze Ctrl + Enter lub wybierz opcję Sterowanie ->Testuj film -> Test, aby przetestować aplikację w środowisku Adobe AIR.
- 2 Aby użyć funkcji Debuguj film, najpierw należy dodać do aplikacji kod ActionScript. W tym celu można dodać poniższą instrukcję trace:

```
trace("Running AIR application using Debug Movie");
```
- 3 Naciśnij klawisze Ctrl + Shift + Enter lub wybierz opcję Debuguj -> Debuguj film -> Debuguj, aby uruchomić aplikację z opcją Debuguj film.

Aplikacja Hello World została przedstawiona na ilustracji:



Pakowanie aplikacji

- 1 Wybierz opcję Plik > Publikuj.
- 2 Podpisz pakiet Adobe AIR za pomocą istniejącego certyfikatu cyfrowego lub utwórz certyfikat z auto-podpisem, wykonując następujące czynności:
 - a Kliknij przycisk Nowy obok pola Certyfikat.
 - b Wprowadź wartości do pól Wydawca, Jednostka organizacji, Nazwa organizacji, E-mail, Kraj, Hasło i Potwierdź hasło.
 - c Określ typ certyfikatu. Opcja typu certyfikatu określa poziom zabezpieczenia: w przypadku certyfikatu 1024-RSA stosowany jest klucz 1024-bitowy (mniej bezpieczny), a w przypadku certyfikatu 2048-RSA stosowany jest klucz 2048-bitowy (bardziej bezpieczny).
 - d Zapisz informacje w certyfikacie poprzez określenie wartości dla opcji Zapisz jako lub kliknięcie przycisku Przeglądaj... w celu wybrania lokalizacji folderu. (Na przykład: *C:/Temp/mycert.pfx*). Po zakończeniu kliknij przycisk OK.
 - e W programie Flash ponownie zostanie wyświetlone okno dialogowe Podpis elektroniczny. Ścieżka i nazwa pliku certyfikatu samopodpisanego pojawią się w oknie dialogowym Certyfikat. Jeśli się nie pojawią, należy wprowadzić ścieżkę i nazwę pliku lub kliknąć przycisk Przeglądaj w celu zlokalizowania i wybrania pliku.

- f W polu tekstowym Hasło w oknie dialogowym Podpis cyfrowy wprowadź to samo hasło, które zostało przypisane w kroku b. Więcej informacji na temat podpisywania aplikacji Adobe AIR zawiera sekcja „[Cyfrowe podpisywanie pliku AIR](#)” na stronie 201.
- 3 W celu utworzenia aplikacji i pliku instalatora kliknij przycisk Publikuj. (W programie Flash CS4 lub CS5 kliknij przycisk OK). Przed utworzeniem pliku AIR należy wykonać operację Testuj film lub Debuguj film, aby utworzyć plik SWF i pliki application.xml.
- 4 W celu zainstalowania aplikacji kliknij dwukrotnie plik AIR (*application.air*) w folderze, w którym zapisano aplikację.
- 5 Kliknij przycisk Instal (Zainstaluj) w oknie dialogowym Application Install (Instalowanie aplikacji).
- 6 Zapoznaj się z ustawieniami Installation Preferences (Preferencje instalowania) oraz Location (Lokalizacja), a następnie upewnij się, że zaznaczone jest pole wyboru Start application after installation (Uruchom aplikację po zainstalowaniu). Następnie kliknij przycisk Continue (Kontynuuj).
- 7 Kliknij przycisk Finish (Zakończ), gdy pojawi się komunikat Installation Completed (Instalowanie zakończone).

Tworzenie pierwszej aplikacji AIR dla systemu Android w programie Flash Professional

Aby programować aplikacje AIR dla systemu Android, należy pobrać rozszerzenie Flash Professional CS5 dla systemu Android ze strony [Adobe Labs](#).

Należy również pobrać i zainstalować zestaw SDK systemu Android z witryny internetowej systemu Android, zgodnie z opisem w temacie [Programiści systemu Android: Instalowanie zestawu SDK](#).

Tworzenie projektu

- 1 Otwórz program Flash Professional CS5.
- 2 Utwórz nowy projekt AIR dla systemu Android.
Na ekranie głównym programu Flash Professional znajduje się łącze umożliwiające tworzenie aplikacji AIR dla systemu Android. Można również wybrać polecenie Plik > Nowy, a następnie wybrać szablon AIR for Android.
- 3 Zapisz dokument w postaci pliku HelloWorld fla.

Pisanie kodu

Ten samouczek tak naprawdę nie dotyczy pisania kodu. Napisz na stole montażowym „Witaj, świecie” za pomocą narzędzia Tekst.

Ustawianie właściwości aplikacji

- 1 Wybierz polecenie Plik > Ustawienia AIR dla systemu Android.
- 2 Na karcie Ogólne wprowadź następujące ustawienia:
 - Plik wyjściowy: HelloWorld.apk
 - Nazwa aplikacji: HelloWorld
 - Identyfikator aplikacji: HelloWorld
 - Numer wersji: 0.0.1
 - Orientacja: Pionowa

3 Na karcie Instalacja określ następujące ustawienia:

- Certyfikat: Wskaż poprawny certyfikat podpisywania kodu środowiska AIR. Można kliknąć przycisk Utwórz w celu utworzenia nowego certyfikatu. (Aplikacje dla systemu Android wdrażane za pośrednictwem sklepu Android Marketplace muszą mieć certyfikaty ważne co najmniej do roku 2033). Wpisz hasło certyfikatu w polu Hasło.
- Typ wdrożenia dla systemu Android: Debugowanie
- Po opublikowaniu: Zaznacz obie opcje
- Wpisz ścieżkę do narzędzia ADB w podkatalogu tools zestawu SDK systemu Android.

4 Zamknij okno dialogowe ustawień systemu Android i kliknij przycisk OK.

Na tym etapie wdrażania aplikacji nie są wymagane ikony ani uprawnienia. Większość aplikacji AIR dla systemu Android wymaga pewnych uprawnień w celu uzyskania dostępu do funkcji chronionych. Należy ustawiać tylko te uprawnienia, których naprawdę wymaga aplikacja, ponieważ użytkownicy mogą odrzucić aplikację, jeśli zażąda ona zbyt wielu uprawnień.

5 Zapisz plik.

Pakowanie i instalowanie aplikacji na urządzeniu z systemem Android

1 Upewnij się, że na urządzeniu jest włączone debugowanie przez USB. Debugowanie przez USB można włączyć w aplikacji Ustawienia za pomocą polecenia Aplikacje > Programowanie.

2 Podłącz urządzenie do komputera za pośrednictwem kabla USB.

3 Jeśli nie zostało jeszcze zainstalowane środowisko wykonawcze AIR, zainstaluj je, przechodząc do sklepu Android Market i pobierając środowisko Adobe AIR. (Środowisko AIR można również zainstalować lokalnie za pomocą „Polecenie installRuntime narzędzia ADT” na stronie 188. W zestawie SDK środowiska AIR znajdują się pakiety dla systemu Android przeznaczone do użytku w emulatorach i na urządzeniach z systemem Android).

4 Wybierz opcję Plik > Publikuj.

Program Flash Professional utworzy plik APK, zainstaluje aplikację na podłączonym urządzeniu z systemem Android i uruchomi ją.

Tworzenie pierwszej aplikacji AIR dla systemu iOS

AIR 2.6 lub nowsza wersja, iOS 4.2 lub nowsza wersja

Korzystając wyłącznie z narzędzi firmy Adobe, można pisać kod podstawowych funkcji aplikacji dla systemu iOS oraz tworzyć i testować te funkcje. Aby było możliwe zainstalowanie na urządzeniu aplikacji dla systemu iOS i rozpowszechnienie jej, należy zostać uczestnikiem programu Apple iOS Developer (który jest usługą płatną). Po zostaniu członkiem programu iOS Developer można uzyskiwać dostęp do witryny iOS Provisioning Portal, gdzie można uzyskać od firmy Apple wymienione poniżej elementy oraz pliki, które są wymagane do zainstalowania aplikacji na telefonie w celu jej przetestowania i późniejszego rozpowszechnienia. Tymi elementami i plikami są między innymi:

- certyfikaty programistów i rozpowszechniania,
- identyfikatory aplikacji,
- pliki informacyjne dotyczące programowania i rozpowszechniania.

Tworzenie zawartości aplikacji

Utwórz plik SWF, który spowoduje wyświetlenie tekstu Hello world! (Witaj świecie!). To zadanie można wykonać przy użyciu środowiska Flash Professional, Flash Builder lub innego środowiska programistycznego. W tym przykładzie użyto edytora tekstu i kompilatora SWF wiersza poleceń, który jest częścią zestawu SDK środowiska Flex.

- 1 Utwórz w wygodnym miejscu katalog przeznaczony do przechowywania plików aplikacji. Utwórz plik o nazwie *HelloWorld.as* i przeprowadź edycję tego pliku w ulubionym edytorze kodu.
- 2 Dodaj następujący kod:

```
package{

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

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

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

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

- 3 Skompiluj klasę za pomocą kompilatora amxmlc.

```
amxmlc HelloWorld.as
```

W tym samym folderze zostanie utworzony plik SWF o nazwie *HelloWorld.swf*.

Uwaga: W tym przykładzie założono, że zmienna środowiskowa ścieżki została skonfigurowana w taki sposób, aby zawierała katalog, w którym znajduje się kompilator amxmlc. Więcej informacji na temat ustawiania ścieżki można znaleźć w sekcji „Zmienne środowiskowe ścieżek” na stronie 320. Można również wpisać pełną ścieżkę do kompilatora amxmlc i innych narzędzi wiersza poleceń użytych w tym przykładzie.

Tworzenie grafiki ikony i ekranu początkowego aplikacji

Wszystkie aplikacje dla systemu iOS mają ikony pojawiające się w interfejsie użytkownika aplikacji iTunes oraz na wyświetlaczu urządzenia.

- 1 Utwórz katalog w katalogu projektu i nadaj mu nazwę icons.
- 2 Utwórz trzy pliki PNG w katalogu icons. Nadaj im nazwy: Icon_29.png, Icon_57.png i Icon_512.png.
- 3 Zmodyfikuj pliki PNG, aby utworzyć odpowiednie grafiki dla aplikacji. Pliki muszą mieć rozmiary 29 na 29 pikseli, 57 na 57 pikseli oraz 512 na 512 pikseli. Na potrzeby tego testu możesz po prostu użyć jako grafiki kwadratów o jednolitym wypełnieniu.

Uwaga: Przesyłając aplikację do sklepu Apple App Store, korzysta się z wersji JPG (nie zaś wersji PNG) pliku 512 pikseli. Do testowania wersji roboczych aplikacji używa się wersji PNG.

W trakcie ładowania każdej aplikacji w telefonie iPhone wyświetlany jest ekran początkowy zawierający obraz. Aby zdefiniować obraz powitalny w pliku PNG.

- 1 W głównym katalogu na komputerze służącym do programowania utwórz plik PNG o nazwie Default.png. (Nie należy umieszczać tego pliku w podkatalogu icons. Należy koniecznie nazwać ten plik Default.png; nazwa powinna rozpoczynać się od wielkiej litery.)
- 2 Dokonaj edycji pliku, tak aby miał on szerokość 320 pikseli i wysokość 480 pikseli. Tymczasowo treść może stanowić zwykły biały prostokąt. (Zostanie to zmienione później.)

Szczegółowe informacje na temat tych elementów graficznych zawiera sekcja „Ikony aplikacji” na stronie 93.

Tworzenie pliku deskryptora aplikacji

Utwórz plik deskryptora aplikacji, który określa podstawowe właściwości aplikacji. To zadanie można wykonać przy użyciu edytora tekstu lub środowiska programistycznego takiego jak program Flash Builder.

- 1 W folderze projektu zawierającym plik HelloWorld.as utwórz plik XML o nazwie *HelloWorld-app.xml*. Przeprowadź edycję tego pliku w ulubionym edytorze XML.
- 2 Dodaj następujący kod XML:

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

Ze względu na przejrzystość w tym przykładzie jest ustawianych tylko kilka z dostępnych opcji.

Uwaga: W przypadku korzystania ze środowiska AIR 2 lub starszego należy użyć elementu `<version>` zamiast elementu `<versionNumber>`.

- 3 Zmień identyfikator aplikacji w taki sposób, aby był zgodny z identyfikatorem aplikacji określonym w witrynie iOS Provisioning Portal. (Na początku identyfikatora nie należy umieszczać losowej wartości zarodkowej).
- 4 Przeprowadź test aplikacji za pomocą narzędzia ADL:

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

Narzędzie ADL powinno spowodować otwarcie na komputerze okna, w którym zostanie wyświetlony komunikat *Hello World!* Jeśli tak się nie stanie, poszukaj błędów w kodzie źródłowym i w deskrypcie aplikacji.

Kompilowanie pliku IPA

Teraz można skompilować plik instalatora IPA za pomocą narzędzia ADT. W tym celu należy mieć certyfikat programisty Apple i klucz prywatny w formacie pliku P12, a także programistyczny profil informacyjny Apple.

Należy uruchomić narzędzie ADT przy użyciu poniższych opcji, zastępując wartości keystore, storepass i provisioning-profile własnymi wartościami.

```
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
```

(Należy użyć pojedynczego wiersza z poleceniem. Podziały wierszy zostały dodane w tym przykładzie wyłącznie w celu zwiększenia jego czytelności).

Narzędzie ADT wygeneruje w katalogu projektu plik instalatora aplikacji dla systemu iOS, *HelloWorld.ipa*. Kompilowanie pliku IPA może zająć kilka minut.

Instalowanie aplikacji na urządzeniu

Aby zainstalować aplikację dla systemu iOS w celu przeprowadzenia testów:

- 1 Otwórz aplikację iTunes.
- 2 Jeśli wcześniej nie wykonano tej czynności, dodaj plik informacyjny na potrzeby tej aplikacji do programu iTunes. W programie iTunes wybierz kolejno opcje Plik > Dodaj do biblioteki. Następnie wybierz plik profilu informacyjnego (dla którego jako typ pliku wybrano mobileprovision).

Do testowania aplikacji na urządzeniu programisty użyj programistycznego profilu informacyjnego.

W dalszej części, podczas przekazywania aplikacji do sklepu iTunes Store, użyj profilu dystrybucyjnego. W celu dystrybucji aplikacji ad-hoc (na wiele urządzeń, z wyłączeniem pośrednictwa sklepu iTunes Store), należy użyć profilu informacyjnego ad-hoc.

Więcej informacji na temat profili informacyjnych zawiera sekcja „[Konfiguracja w systemie iOS](#)” na stronie 70.

- 3 Niektóre wersje iTunes nie dopuszczają do zastępowania aplikacji w sytuacji, gdy ta sama jej wersja jest już zainstalowana. W takim przypadku należy usunąć aplikację z urządzenia oraz z listy aplikacji w iTunes.
- 4 Kliknij dwukrotnie plik IPA dla swojej aplikacji. Powinien on pojawić się na liście aplikacji w programie iTunes.
- 5 Podłącz urządzenie do portu USB komputera.
- 6 W iTunes sprawdź kartę Aplikacja dla urządzenia i upewnij się, że aplikacja została wybrana z listy aplikacji do zainstalowania.
- 7 Wybierz urządzenie z listy po lewej stronie aplikacji iTunes. Następnie kliknij przycisk Synchronizacja. Po zakończeniu synchronizacji aplikacja Hello World pojawi się na telefonie iPhone.

Jeśli nowa wersja nie została zainstalowana, usuń ją z urządzenia i z listy aplikacji w programie iTunes, a następnie wykonaj tę procedurę ponownie. Może mieć to miejsce, jeśli obecnie zainstalowana wersja ma ten sam identyfikator aplikacji i numer wersji.

Edycja grafiki ekranu początkowego

Przed skompilowaniem aplikacji utworzony został plik Default.png (patrz „[Tworzenie grafiki ikony i ekranu początkowego aplikacji](#)” na stronie 29). Obraz zawarty w tym pliku PNG jest wyświetlany w trakcie ładowania aplikacji. Podczas testowania aplikacji na telefonie iPhone można było zaobserwować pusty ekran — to właśnie był obraz początkowy.

Należy teraz zmienić go odpowiednio do potrzeb tworzonej aplikacji („Hello World!”):

- 1 Otwórz aplikację na urządzeniu. Po pojawieniu się po raz pierwszy napisu „Hello World” naciśnij i przytrzymaj przycisk Home (poniżej ekranu). Przytrzymując przycisk Home, naciśnij przycisk Power/Sleep (w górnej części telefonu iPhone). Zrzut ekranu zostanie przesłany do folderu Rolka z aparatu.
- 2 Prześlij obraz na komputer programisty, wysyłając zdjęcia z programu iPhoto lub innej aplikacji do transferu zdjęć. (W systemie Mac OS można również skorzystać z aplikacji Pobieranie obrazu.)

Zdjęcie można także przesłać na komputer programisty za pomocą poczty elektronicznej.

- Otwórz aplikację Zdjęcia.
- Otwórz Rolkę z aparatu.
- Otwórz obraz zrzutu ekranu.
- Stuknij obraz, a następnie przycisk (strzałkę) „dalej” w dolnym lewym rogu. Następnie kliknij przycisk Wyślij zdjęcie (email), aby wysłać obraz do siebie.

- 3 Zastąp plik Default.png (w katalogu programistycznym) wersją PNG obrazu zrzutu ekranowego.

- 4 Ponownie skompiluj aplikację (zobacz „[Kompilowanie pliku IPA](#)” na stronie 31) i jeszcze raz zainstaluj ją na urządzeniu.

Podczas ładowania aplikacji jest teraz wyświetlany nowy ekran powitalny.

Uwaga: *Możliwe jest utworzenie dowolnej grafiki do zapisania w pliku Default.png; jedynym warunkiem są prawidłowe wymiary (320 na 480 pikseli). Najlepiej jednak by treść pliku Default.png odpowiadała początkowemu stanowi aplikacji.*

Tworzenie pierwszej aplikacji AIR opartej na kodzie HTML w programie Dreamweaver

W celu uzyskania szybkich i niezawodnych instrukcji na temat działania Adobe® AIR® można wykorzystać poniższe instrukcje w celu utworzenia i spakowania prostej aplikacji „Hello World” HTML AIR za pomocą rozszerzenia Adobe® AIR® dla programu Dreamweaver®.

Przed przystąpieniem do wykonywania tej procedury należy pobrać i zainstalować środowisko Adobe AIR, które jest dostępne na stronie www.adobe.com/go/air_pl.

Instrukcje dotyczące instalowania rozszerzenia Adobe AIR dla programu Dreamweaver zawiera sekcja [Instalowanie rozszerzenia AIR dla programu Dreamweaver](#).

Ogólne informacje o rozszerzeniu, w tym także informacje o wymaganiach systemowych, zawiera sekcja [Rozszerzenie AIR dla programu Dreamweaver](#).

Uwaga: *Aplikacje AIR oparte na języku HTML mogą być programowane wyłącznie na potrzeby profili komputerowego i rozszerzonego komputerowego. Profil mobile nie jest obsługiwany.*

Przygotowanie plików aplikacji

Stronę startową aplikacji Adobe AIR oraz wszystkie powiązane strony należy zdefiniować w serwisie programu Dreamweaver:

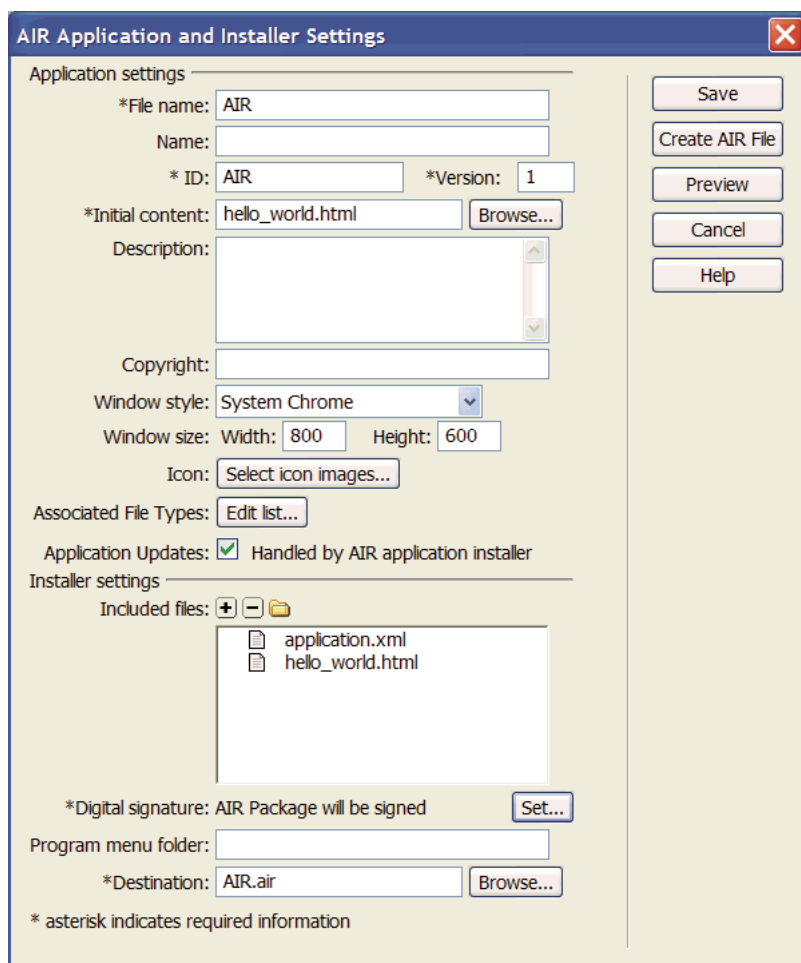
- 1 Uruchom program Dreamweaver i upewnij się, że zdefiniowany jest serwis.
- 2 Otwórz nową stronę HTML, wybierając opcję Plik > Nowe. Następnie wybierz opcję HTML w kolumnie typu strony oraz opcję Brak w kolumnie układu. Następnie kliknij przycisk Utwórz.
- 3 Na nowej stronie wpisz **Hello World!**
Ten przykład jest bardzo prosty, można jednak określić dowolny styl tekstu, dodać więcej treści do strony, połączyć inne strony ze stroną startową itp.
- 4 Zapisz stronę (Plik > Zapisz) jako hello_world.html. Upewnij się, że plik został zapisany w serwisie programu Dreamweaver.

Więcej informacji na temat serwisów programu Dreamweaver zawiera pomoc programu Dreamweaver.

Tworzenie aplikacji Adobe AIR

- 1 Upewnij się, że strona hello_world.html jest otwarta w oknie Dokument programu Dreamweaver. (Instrukcje na temat tworzenia tej strony zawiera poprzednia sekcja).
- 2 Wybierz opcje Serwis > Ustawienia aplikacji Air.
Większość wymaganych ustawień w oknie dialogowym aplikacji i ustawień AIR program wprowadza automatycznie. Użytkownik musi wybrać początkową treść (lub stronę startową) aplikacji.
- 3 Kliknij przycisk Przeglądaj obok opcji Treść początkowa, przejdź do strony hello_world.html i zaznacz ją.
- 4 Obok opcji Podpis cyfrowy kliknij przycisk Ustaw.
Podpis cyfrowy gwarantuje, że kod aplikacji nie został zmodyfikowany ani uszkodzony od czasu jego utworzenia przez autora oprogramowania. Podpis jest wymagany we wszystkich aplikacjach Adobe AIR.
- 5 W oknie dialogowym Podpis cyfrowy wybierz opcję Podpisz pakiet AIR certyfikatem cyfrowym, a następnie kliknij przycisk Utwórz. (Jeśli masz dostęp do certyfikatu cyfrowego kliknij przycisk Przeglądaj, aby go po prostu wybrać).
- 6 Wypełnij wymagane pola w oknie dialogowym tworzenia cyfrowego certyfikatu samopodpisanego. Konieczne będzie wprowadzenie nazwy oraz hasła, a następnie potwierdzenie hasła, a także wprowadzenie nazwy dla pliku certyfikatu cyfrowego. Program Dreamweaver zapisuje certyfikat cyfrowy w katalogu głównym serwisu.
- 7 Kliknij OK, aby powrócić do okna Podpis cyfrowy.
- 8 W oknie dialogowym Podpis cyfrowy wprowadź hasło określone dla certyfikatu cyfrowego i kliknij przycisk OK.

Wypełnione okno dialogowe Aplikacja AIR i Ustawienia instalatora może wyglądać, jak poniżej:



Bardziej wyczerpujące omówienie wszystkich opcji okna dialogowego i sposobów ich edytowania zawiera sekcja [Tworzenie aplikacji AIR w programie Dreamweaver](#).

9 Kliknij przycisk Utwórz plik AIR.

Program Dreamweaver utworzy plik aplikacji Adobe AIR i zapisze go w katalogu głównym serwisu. Program Dreamweaver utworzy również plik application.xml i zapisze go w tym samym miejscu. Ten plik służy jako manifest, ponieważ definiuje różne właściwości aplikacji.

Instalowanie aplikacji na komputerze stacjonarnym

Aplikację, dla której utworzono plik aplikacji, można zainstalować na dowolnym komputerze stacjonarnym.

- 1 Przenieś plik aplikacji Adobe AIR na zewnątrz serwisu Dreamweaver, na lokalny komputer stacjonarny lub inny komputer stacjonarny.

Ten krok jest opcjonalny. Możliwe jest również zainstalowanie nowej aplikacji na komputerze bezpośrednio z katalogu serwisu Dreamweaver.

- 2 Kliknij dwukrotnie plik wykonywalny aplikacji (.air), aby zainstalować aplikację.

Wyświetlanie podglądu aplikacji Adobe AIR

Podgląd stron, które będą częścią aplikacji AIR, można wyświetlić w dowolnym momencie. Oznacza to, że w celu sprawdzenia wyglądu zainstalowanej aplikacji, nie trzeba jej pakować.

- 1 Upewnij się, że strona `hello_world.html` jest otwarta w oknie Dokument programu Dreamweaver.
- 2 Na pasku narzędzi Dokument kliknij przycisk Wyświetl podgląd/debuguj w przeglądarce, a następnie wybierz opcję podglądu w AIR.

Można również nacisnąć kombinację klawiszy `Ctrl+Shift+F12` (Windows) lub `Cmd+Shift+F12` (Macintosh).

Na podglądzie widoczny jest obraz strony startowej aplikacji po zainstalowaniu aplikacji na komputerze stacjonarnym przez użytkownika końcowego.

Tworzenie pierwszej aplikacji AIR w formacie HTML za pomocą programu AIR SDK

W celu uzyskania szybkich i niezawodnych instrukcji na temat działania Adobe® AIR® można wykorzystać poniższe instrukcje w celu utworzenia i spakowania prostej aplikacji HTML AIR o nazwie „Hello World”.

Przed rozpoczęciem należy zainstalować środowisko wykonawcze i skonfigurować program AIR SDK. Podczas tego kursu będzie używany program *AIR Debug Launcher* (ADL) oraz narzędzie *AIR Developer Tool* (ADT). ADL i ADT to programy narzędziowe dostępne z poziomu wiersza poleceń, które można znaleźć w katalogu `bin` pakietu AIR SDK (zobacz „[Instalowanie AIR SDK](#)” na stronie 18). Twórcy niniejszego kursu założyli, że czytelnik zna podstawy uruchamiania programów z wiersza poleceń i wie, w jaki sposób należy skonfigurować wymagane zmienne środowiskowe ścieżki dla systemu operacyjnego.

Uwaga: Użytkownicy programu Adobe® Dreamweaver® powinni zapoznać się z sekcją „[Tworzenie pierwszej aplikacji AIR opartej na kodzie HTML w programie Dreamweaver](#)” na stronie 32.

Uwaga: Aplikacje AIR oparte na języku HTML mogą być programowane wyłącznie na potrzeby profili komputerowego i rozszerzonego komputerowego. Profil mobile nie jest obsługiwany.

Tworzenie plików projektu

Każdy projekt HTML AIR musi zawierać następujące dwa pliki: plik deskryptora aplikacji, który określa metadane aplikacji, oraz stronę HTML najwyższego poziomu. Oprócz wymaganych plików projekt zawiera plik kodu JavaScript `AIRAliases.js`, który definiuje wygodne zmienne alias dla klas API AIR.

- 1 Utwórz katalog o nazwie `HelloWorld`, który będzie zawierał pliki projektu.
- 2 Utwórz plik XML o nazwie `HelloWorld-app.xml`.
- 3 Utwórz plik HTML o nazwie `HelloWorld.html`.
- 4 Skopiuj plik `AIRAliases.js` z folderu struktury programu AIR SDK do katalogu projektu.

Tworzenie pliku deskryptora dla aplikacji AIR

Aby rozpocząć tworzenie aplikacji AIR, należy utworzyć plik deskryptora aplikacji XML o następującej strukturze:


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

1 Otwórz plik HelloWorld-app.xml do edycji.

2 Dodaj główny element `<application>` zawierający atrybut przestrzeni nazw AIR:

`<application xmlns="http://ns.adobe.com/air/application/2.7">` Ostatni segment przestrzeni nazw — „2.7” — określa wymaganą wersję środowiska wykonawczego.

3 Dodaj element `<id>`:

`<id>examples.html.HelloWorld</id>` Identyfikator aplikacji w unikalny sposób identyfikuje aplikację wraz z identyfikatorem wydawcy (pobieranym przez AIR z certyfikatu używanego w celu podpisania pakietu aplikacji). Identyfikator aplikacji jest używany w celu zainstalowania, uzyskania dostępu do prywatnego katalogu pamięci masowej systemu plików aplikacji, uzyskania dostępu do prywatnej zaszyfrowanej pamięci masowej, a także na potrzeby komunikacji wewnątrz aplikacji.

4 Dodaj element `<versionNumber>`.

`<versionNumber>0.1</versionNumber>` Ułatwia użytkownikom określenie, którą wersję aplikacji instalują.

Uwaga: W przypadku korzystania ze środowiska AIR 2 lub starszego należy użyć elementu `<version>` zamiast elementu `<versionNumber>`.

5 Dodaj element `<filename>`:

`<filename>HelloWorld</filename>` Nazwa używana dla pliku wykonywalnego aplikacji, katalogu instalacyjnego, a także dla innych odwołań do aplikacji w systemie operacyjnym.

6 Dodaj element `<initialWindow>` zawierający następujące elementy potomne, w celu określenia właściwości dla początkowego okna aplikacji:

`<content>HelloWorld.html</content>` Identyfikuje główny plik HTML, który ładuje środowisko AIR.

`<visible>true</visible>` Powoduje natychmiastowe wyświetlenie okna.

`<width>400</width>` Ustawia szerokość okna (w pikselach).

`<height>200</height>` Ustawia wysokość okna.

7 Zapisz plik. Zakończony plik deskryptora aplikacji powinien wyglądać, jak poniżej:

Tworzenie pierwszej aplikacji

```
<?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>
```

W tym przykładzie przedstawiono ustawianie tylko kilku możliwych właściwości aplikacji. Opis wszystkich właściwości aplikacji, które umożliwiają ustawianie takich elementów, jak karnacja okna, wielkość okna, przezroczystość, domyślny katalog instalacyjny, skojarzone typy plików oraz ikony aplikacji, zawiera sekcja „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Tworzenie strony HTML aplikacji

Kolejnym etapem jest utworzenie prostej strony HTML, która będzie pełniła rolę pliku głównego dla aplikacji AIR.

- 1 Otwórz plik `HelloWorld.html` do edycji. Dodaj następujący kod HTML:

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

- 2 W sekcji `<head>` pliku HTML zaimportuj plik `AIRAliases.js`:

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

AIR definiuje właściwość o nazwie `runtime` w obiekcie okna HTML. Właściwość `runtime` zapewnia dostęp do wbudowanych klas AIR za pomocą pełnej nazwy pakietu klasy. Na przykład: w celu utworzenia obiektu pliku AIR można dodać następującą instrukcję w kodzie JavaScript:

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

Plik `AIRAliases.js` definiuje wygodne aliasy dla większości użytecznych interfejsów API AIR. Za pomocą pliku `AIRAliases.js` można skrócić odwołanie do klasy `File` w następujący sposób:

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

- 3 Poniżej znacznika `script` w pliku `AIRAliases` dodaj kolejny znacznik `script` zawierający funkcję JavaScript przeznaczoną do obsługi zdarzenia `onLoad`:

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

Funkcja `appLoad()` wywołuje funkcję `air.trace()`. Komunikat `trace` zostanie wyświetlony w konsoli poleceń po uruchomieniu aplikacji za pomocą ADL. Instrukcje `trace` mogą być bardzo użyteczne podczas debugowania.

4 Zapisz plik.

Plik `HelloWorld.html` powinien wyglądać w następujący sposób:

```
<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>
```

Testowanie aplikacji

W celu uruchomienia i przetestowania aplikacji z wiersza poleceń należy użyć programu narzędziowego ADL (AIR Debug Launcher). Plik wykonywalny programu ADL można znaleźć w katalogu `bin` programu AIR SDK. Jeśli zestaw SDK środowiska AIR nie został jeszcze skonfigurowany, zobacz „[Instalowanie AIR SDK](#)” na stronie 18.

- 1 Otwórz konsolę lub powłokę poleceń. Przejdź do katalogu, który został utworzony dla danego projektu.
- 2 Uruchom następujące polecenie:

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

Zostanie otwarte okno AIR zawierające aplikację. Zostanie również wyświetlone okno konsoli zawierające komunikat wynikający z wywołania metody `air.trace()`.

Więcej informacji zawiera sekcja „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Tworzenie pliku instalacyjnego AIR

Jeśli aplikacja działa pomyślnie, można użyć programu narzędziowego ADL w celu spakowania aplikacji do pliku instalacyjnego AIR. Plik instalacyjny AIR jest plikiem archiwum, który zawiera wszystkie pliki aplikacji, które można rozpowszechniać wśród użytkowników. Przed zainstalowaniem spakowanego pliku AIR należy zainstalować środowisko Adobe AIR.

Aby zapewnić bezpieczeństwo aplikacji, wszystkie pliki instalacyjne AIR muszą być podpisane cyfrowo. Programiści mogą generować podstawowe, samopodpisane certyfikaty za pomocą ADT lub innego narzędzia do generowania certyfikatów. Można również zakupić dostępny w sprzedaży certyfikat podpisujący kod z ośrodka certyfikacji, takiego jak VeriSign lub Thawte. Gdy użytkownicy instalują samopodpisany plik AIR, podczas procesu instalowania wydawca jest określony jako „unknown”. Jest to spowodowane tym, że certyfikat samopodpisany gwarantuje tylko, że plik AIR nie został zmodyfikowany od czasu utworzenia. Jednak nic nie gwarantuje, że ktoś nie utworzył fałszywego samopodpisanego pliku AIR, który został przedstawiony jako aplikacja innego użytkownika. W przypadku rozpowszechnianych publicznie plików AIR zalecane jest stosowanie możliwego do sprawdzenia certyfikatu komercyjnego. Przegląd zagadnień związanych z zabezpieczeniami AIR zawiera sekcja [Bezpieczeństwo w AIR](#) (dla programistów ActionScript) oraz sekcja [Bezpieczeństwo w AIR](#) (dla programistów HTML).

Generowanie pary: certyfikat samopodpisany i klucz

- ❖ Do wiersza poleceń wprowadź następujące polecenie (plik wykonywalny ADT znajduje się w katalogu bin programu AIR SDK):

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

ADT wygeneruje plik kluczy o nazwie *sampleCert.pfx*, który będzie zawierał certyfikat i powiązany z nim klucz prywatny.

W tym przykładzie wykorzystano minimalną liczbę atrybutów, jakie można ustawić dla certyfikatu. Typem klucza musi być *1024-RSA* lub *2048-RSA* (zobacz „[Podpisywanie aplikacji AIR](#)” na stronie 201).

Tworzenie pliku instalacyjnego AIR

- ❖ W wierszu polecenia wpisz następujące polecenie (w pojedynczym wierszu):

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

Zostanie wyświetlona zachęta do wprowadzenia hasła pliku kluczy.

Argument *HelloWorld.air* to plik AIR wygenerowany przez ADT. *HelloWorld-app.xml* to plik deskryptora aplikacji. Kolejne argumenty to pliki używane przez aplikację. W tym przykładzie wykorzystano tylko dwa pliki, ale można dołączyć dowolną liczbę plików i katalogów. Narzędzie ADT sprawdza, czy w pakiecie znajduje się główny plik z zawartością, *HelloWorld.html*. Jeśli nie zostanie dołączony plik *AIRAliases.js*, wówczas aplikacja po prostu nie będzie działać.

Po utworzeniu pakietu AIR można zainstalować i uruchomić aplikację poprzez dwukrotne kliknięcie pliku pakietu. Nazwę pliku AIR można również wpisać jako polecenie do powłoki lub okna poleceń.

Kolejne czynności

W środowisku AIR kody HTML i JavaScript zwykle działają tak samo, jak w typowej przeglądarce sieci Web. (Środowisko AIR wykorzystuje ten sam mechanizm renderowania WebKit, z którego korzysta przeglądarka Safari). Jednak istnieją pewne różnice, które należy poznać przed przystąpieniem do programowania aplikacji HTML w środowisku AIR. Więcej informacji na temat tych różnic i inne ważne tematy zawiera sekcja [Programowanie w HTML i JavaScript](#).

Tworzenie pierwszej aplikacji AIR na komputery stacjonarne za pomocą zestawu SDK środowiska Flex

W celu uzyskania skróconych i praktycznych informacji o działaniu programu Adobe® AIR® należy wykorzystać te instrukcje w celu utworzenia jednej aplikacji AIR „Hello World” (wykorzystującej pliki SWF) w pakiecie Flex SDK. W tym samouczku pokazano, w jaki sposób kompilować, testować i pakować aplikacje AIR za pomocą narzędzi wiersza poleceń wchodzących w skład zestawu SDK środowiska Flex. (Zestaw SDK środowiska Flex zawiera zestaw SDK środowiska AIR).

Przed rozpoczęciem należy zainstalować środowisko wykonawcze i skonfigurować program Adobe® Flex™. W niniejszym samouczku wykorzystano kompilator *AMXMLC*, program *AIR Debug Launcher* (ADL) oraz program *AIR Developer Tool* (ADT). Te programy można znaleźć w katalogu *bin* pakietu Flex SDK (patrz „[Konfigurowanie Flex SDK](#)” na stronie 20).

Tworzenie pliku deskryptora dla aplikacji AIR

W niniejszej sekcji opisano sposób tworzenia deskryptora aplikacji, który jest plikiem XML o następującej strukturze:

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

1 Utwórz plik XML o nazwie `HelloWorld-app.xml` i zapisz go w katalogu projektu.

2 Dodaj element `<application>` zawierający atrybut przestrzeni nazw AIR:

`<application xmlns="http://ns.adobe.com/air/application/2.7">` Ostatni segment przestrzeni nazw — „2.7” — określa wymaganą wersję środowiska wykonawczego.

3 Dodaj element `<id>`:

`<id>samples.flex.HelloWorld</id>` Identyfikator aplikacji w unikalny sposób identyfikuje aplikację wraz z identyfikatorem wydawcy (pobieranym przez AIR z certyfikatu używanego w celu podpisania pakietu aplikacji). Zalecaną formą jest rozdzielony kropkami ciąg znaków w stylu odwróconego adresu DNS, taki jak `com.firma.nazwa_aplikacji`. Identyfikator aplikacji jest używany w celu zainstalowania, uzyskania dostępu do prywatnego katalogu pamięci masowej systemu plików aplikacji, uzyskania dostępu do prywatnej zaszyfrowanej pamięci masowej, a także na potrzeby komunikacji wewnątrz aplikacji.

4 Dodaj element `<versionNumber>`.

`<versionNumber>1.0</versionNumber>` Ułatwia użytkownikom określenie, którą wersję aplikacji instalują.

Uwaga: W przypadku korzystania ze środowiska AIR 2 lub starszego należy użyć elementu `<version>` zamiast elementu `<versionNumber>`.

5 Dodaj element `<filename>`:

`<filename>HelloWorld</filename>` Nazwa używana dla pliku wykonywalnego aplikacji, katalogu instalacyjnego, a także dla podobnych odwołań w systemie operacyjnym.

6 Dodaj element `<initialWindow>` zawierający następujące elementy potomne w celu określenia właściwości dla początkowego okna aplikacji:

`<content>HelloWorld.swf</content>` Identyfikuje główny plik SWF, który wczytuje środowisko AIR.

`<visible>true</visible>` Powoduje natychmiastowe wyświetlenie okna.

`<width>400</width>` Ustawia szerokość okna (w pikselach).

`<height>200</height>` Ustawia wysokość okna.

7 Zapisz plik. Zakończony plik deskryptora aplikacji powinien być następujący:

```
<?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>
```

W tym przykładzie przedstawiono ustawianie tylko kilku możliwych właściwości aplikacji. Opis wszystkich właściwości aplikacji, które umożliwiają ustawianie takich elementów, jak karnacja okna, wielkość okna, przezroczystość, domyślny katalog instalacyjny, skojarzone typy plików oraz ikony aplikacji, zawiera sekcja „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Pisanie kodu aplikacji

Uwaga: Aplikacje AIR oparte na plikach SWF mogą korzystać z głównej klasy zdefiniowanej w języku MXML lub w języku Adobe® ActionScript® 3.0. W tym przykładzie przedstawiono wykorzystanie pliku MXML w celu zdefiniowania klasy głównej. Proces tworzenia aplikacji AIR za pomocą klasy głównej ActionScript przebiega podobnie. Zamiast kompilować plik MXML do postaci pliku SWF należy skompilować plik klasy ActionScript. W przypadku korzystania z języka ActionScript główna klasa musi stanowić rozszerzenie klasy `flash.display.Sprite`.

Aplikacje AIR utworzone w środowisku Flex — podobnie jak wszystkie aplikacje oparte na Flex — zawierają główny plik MXML. Jako element główny w aplikacjach AIR na komputery stacjonarne zamiast składnika `Application` jest stosowany składnik `WindowedApplication`. Komponent `WindowedApplication` udostępnia właściwości, metody i zdarzenia przeznaczone do kontrolowania aplikacji i jej początkowego okna. Na platformach i w profilach, dla których środowisko AIR nie obsługuje wielu okien, należy nadal używać składnika `Application`. W aplikacjach Flex dla urządzeń przenośnych można również używać składników `View` lub `TabbedViewNavigatorApplication`.

Poniższa procedura tworzy aplikację Hello World:

- 1 Za pomocą edytora tekstu utwórz plik o nazwie `HelloWorld.mxml` i dodaj następujący kod MXML:

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

- 2 Następnie dodaj komponent `Label` do aplikacji (umieść go w znaczniku `WindowedApplication`).
- 3 Ustaw właściwość `text` komponentu `Label` na `"Hello AIR"`.
- 4 Zdefiniuj ograniczenia układu w taki sposób, aby napis był zawsze umieszczany na środku.

Poniższy przykład prezentuje utworzony dotychczas kod:

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

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

Kompilowanie aplikacji

Zanim możliwe będzie uruchomienie i zdebugowanie aplikacji należy skompilować kod MXML — w celu utworzenia pliku SWF — za pomocą kompilatora amxmlc. Kompilator amxmlc jest dostępny w katalogu bin pakietu Flex SDK. W razie potrzeby można ustawić ścieżkę środowiska komputera w taki sposób, aby uwzględniała katalog bin pakietu Flex SDK. Ustawienie ścieżki ułatwia uruchamianie programów narzędziowych z wiersza poleceń.

- 1 Otwórz powłokę lub terminal poleceń i przejdź do folderu projektu aplikacji AIR.
- 2 Wprowadź następujące polecenie:

```
amxmlc HelloWorld.mxml
```

Uruchomienie kompilatora amxmlc spowoduje utworzenie pliku HelloWorld.swf, który zawiera skompilowany kod aplikacji.

Uwaga: Jeśli nie można skompilować aplikacji, usuń błędy składniowe lub błędy pisowni. W oknie konsoli używanym do uruchamiania kompilatora amxmlc zostaną wyświetlone błędy i ostrzeżenia.

Więcej informacji zawiera rozdział „[Kompilowanie plików źródłowych MXML i ActionScript dla środowiska AIR](#)” na stronie 166.

Testowanie aplikacji

W celu uruchomienia i przetestowania aplikacji z wiersza poleceń należy użyć programu AIR Debug Launcher (ADL), aby uruchomić aplikację za pomocą pliku deskryptora aplikacji. (Program ADL znajduje się w katalogu bin pakietu Flex SDK).

- ❖ Do okna wiersza poleceń wprowadź następujące polecenie:

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

Uzyskana wynikowa aplikacja AIR wygląda podobnie do przedstawionej na ilustracji:



Za pomocą właściwości horizontalCenter i verticalCenter elementu sterującego Label tekst należy umieścić na środku okna. Okno można przesuwać i powiększać/zmniejszać tak samo, jak okno każdej innej aplikacji.

Więcej informacji zawiera sekcja „[AIR Debug Launcher \(ADL\)](#)” na stronie 169.

Tworzenie pliku instalacyjnego AIR

Jeśli aplikacja działa pomyślnie, można użyć programu narzędziowego ADL w celu spakowania aplikacji do pliku instalacyjnego AIR. Plik instalacyjny AIR jest plikiem archiwum, który zawiera wszystkie pliki aplikacji, które można rozpowszechnić wśród użytkowników. Przed zainstalowaniem spakowanego pliku AIR należy zainstalować środowisko Adobe AIR.

Aby zapewnić bezpieczeństwo aplikacji, wszystkie pliki instalacyjne AIR muszą być podpisane cyfrowo. Programiści mogą generować podstawowe, samopodpisane certyfikaty za pomocą ADT lub innego narzędzia do generowania certyfikatów. Można również zakupić komercyjny certyfikat podpisujący kod w ośrodku certyfikacji. Gdy użytkownicy instalują samopodpisany plik AIR, podczas procesu instalowania wydawca jest określony jako „unknown”. Jest to spowodowane tym, że certyfikat samopodpisany gwarantuje tylko, że plik AIR nie został zmodyfikowany od czasu utworzenia. Jednak nic nie gwarantuje, że ktoś nie utworzył fałszywego samopodpisanego pliku AIR, który został przedstawiony jako aplikacja innego użytkownika. W przypadku rozpowszechnianych publicznie plików AIR zalecane jest stosowanie możliwego do sprawdzenia certyfikatu komercyjnego. Przegląd zagadnień związanych z zabezpieczeniami AIR zawiera sekcja [Bezpieczeństwo w AIR](#) (dla programistów ActionScript) oraz sekcja [Bezpieczeństwo w AIR](#) (dla programistów HTML).

Generowanie pary: certyfikat samopodpisany i klucz

- ❖ Do wiersza poleceń wprowadź następujące polecenie (plik wykonywalny ADT można znaleźć w katalogu bin pakietu Flex SDK):

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

W tym przykładzie wykorzystano minimalną liczbę atrybutów, jakie można ustawić dla certyfikatu. Typem klucza musi być *1024-RSA* lub *2048-RSA* (zobacz „[Podpisywanie aplikacji AIR](#)” na stronie 201).

Tworzenie pakietu AIR

- ❖ W wierszu polecenia wpisz następujące polecenie (w pojedynczym wierszu):

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

Zostanie wyświetlona zachęta do wprowadzenia hasła pliku kluczy. Wpisz hasło i naciśnij klawisz Enter. Ze względów bezpieczeństwa znaki hasła nie są wyświetlane.

Argument HelloWorld.air to plik AIR wygenerowany przez ADT. HelloWorld-app.xml to plik deskryptora aplikacji. Kolejne argumenty to pliki używane przez aplikację. W tym przykładzie wykorzystano tylko trzy pliki, ale można dołączyć dowolną liczbę plików i katalogów.

Po utworzeniu pakietu AIR można zainstalować i uruchomić aplikację poprzez dwukrotne kliknięcie pliku pakietu. Nazwę pliku AIR można również wpisać jako polecenie do powłoki lub okna poleceń.

Więcej informacji zawiera rozdział „[Pakowanie pliku instalacyjnego AIR na komputery stacjonarne](#)” na stronie 56.

Tworzenie pierwszej aplikacji AIR dla systemu Android za pomocą zestawu SDK środowiska Flex

Do rozpoczęcia pracy są wymagane zainstalowane i skonfigurowane zestawy SDK środowisk AIR i Flex. W tym samouczku są stosowane kompilator *AMXMLC* z zestawu SDK środowiska Flex oraz narzędzia *AIR Debug Launcher* (ADL) i *AIR Developer Tool* (ADT) z zestawu SDK środowiska AIR. Zobacz „[Konfigurowanie Flex SDK](#)” na stronie 20.

Należy również pobrać i zainstalować zestaw SDK systemu Android z witryny internetowej systemu Android, zgodnie z opisem w temacie [Programiści systemu Android: Instalowanie zestawu SDK](#).

Uwaga: Informacje dotyczące pisania programów na telefon iPhone zawiera artykuł [Tworzenie aplikacji Hello World na telefon iPhone w programie Flash Professional CS5](#).

Tworzenie pliku deskryptora dla aplikacji AIR

W niniejszej sekcji opisano sposób tworzenia deskryptora aplikacji, który jest plikiem XML o następującej strukturze:

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

1 Utwórz plik XML o nazwie HelloWorld-app.xml i zapisz go w katalogu projektu.

2 Dodaj element `<application>` zawierający atrybut przestrzeni nazw AIR:

`<application xmlns="http://ns.adobe.com/air/application/2.7">` Ostatni segment przestrzeni nazw — „2.7” — określa wymaganą wersję środowiska wykonawczego.

3 Dodaj element `<id>`:

`<id>samples.android.HelloWorld</id>` Identyfikator aplikacji w unikalny sposób określa aplikację wraz z identyfikatorem wydawcy (uzyskiwanym przez środowisko AIR z certyfikatu używanego w celu podpisania pakietu aplikacji). Zalecaną formą jest rozdzielony kropkami ciąg zapisany w odwrotnej notacji DNS, taki jak `com.firma.nazwa_aplikacji`.

4 Dodaj element `<versionNumber>`.

`<versionNumber>0.0.1</versionNumber>` Ułatwia użytkownikom określenie, którą wersję aplikacji instalują.

5 Dodaj element `<filename>`:

`<filename>HelloWorld</filename>` Nazwa używana dla pliku wykonywalnego aplikacji, katalogu instalacyjnego, a także dla podobnych odwołań w systemie operacyjnym.

6 Dodaj element `<initialWindow>` zawierający następujące elementy potomne w celu określenia właściwości dla początkowego okna aplikacji:

`<content>HelloWorld.swf</content>` Identyfikuje główny plik HTML, który ładuje środowisko AIR.

7 Dodaj element `<supportedProfiles>`.

`<supportedProfiles>mobileDevice</supportedProfiles>` Wskazuje, że aplikacja działa wyłącznie w profilu urządzeń przenośnych.

8 Zapisz plik. Zakończony plik deskryptora aplikacji powinien być następujący:

```
<?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>
```

W tym przykładzie przedstawiono ustawianie tylko kilku możliwych właściwości aplikacji. W pliku deskryptora aplikacji można również używać innych ustawień. Na przykład do elementu `initialWindow` można dodać ustawienie `<fullScreen>true</fullScreen>`, co pozwoli utworzyć aplikację pełnoekranową. W celu włączenia w systemie Android zdalnego debugowania i funkcji o kontrolowanym dostępie do deskryptora aplikacji trzeba również dodać uprawnienia systemu Android. Uprawnienia nie są wymagane w przypadku tej prostej aplikacji, więc na razie nie trzeba ich dodawać.

Więcej informacji zawiera rozdział „[Konfigurowanie właściwości aplikacji dla urządzeń przenośnych](#)” na stronie 75.

Pisanie kodu aplikacji

Utwórz plik o nazwie `HelloWorld.as` i dodaj poniższy kod, korzystając z edytora tekstu:

```
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 );
        }
    }
}
```

Kompilowanie aplikacji

Zanim możliwe będzie uruchomienie i zdebugowanie aplikacji należy skompilować kod MXML — w celu utworzenia pliku SWF — za pomocą kompilatora `amxmlc`. Kompilator `amxmlc` jest dostępny w katalogu `bin` pakietu Flex SDK. W razie potrzeby można ustawić ścieżkę środowiska komputera w taki sposób, aby uwzględniła katalog `bin` pakietu Flex SDK. Ustawienie ścieżki ułatwia uruchamianie programów narzędziowych z wiersza poleceń.

- 1 Otwórz powłokę lub terminal poleceń i przejdź do folderu projektu aplikacji AIR.
- 2 Wprowadź następujące polecenie:

```
amxmlc HelloWorld.as
```

Uruchomienie kompilatora `amxmlc` spowoduje utworzenie pliku `HelloWorld.swf`, który zawiera skompilowany kod aplikacji.

Uwaga: Jeśli nie można skompilować aplikacji, usuń błędy składniowe lub błędy pisowni. W oknie konsoli używanym do uruchamiania kompilatora `amxmlc` zostaną wyświetlone błędy i ostrzeżenia.

Więcej informacji zawiera rozdział „[Kompilowanie plików źródłowych MXML i ActionScript dla środowiska AIR](#)” na stronie 166.

Testowanie aplikacji

W celu uruchomienia i przetestowania aplikacji z wiersza poleceń należy użyć programu AIR Debug Launcher (ADL), aby uruchomić aplikację za pomocą pliku deskryptora aplikacji. (Narzędzie ADL znajduje się w katalogu bin zestawu SDK środowisk AIR i Flex).

- ❖ Do okna wiersza polecenia wprowadź następujące polecenie:

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

Więcej informacji można znaleźć w sekcji „[Symulowanie urządzenia przy użyciu narzędzia ADL](#)” na stronie 106.

Tworzenie pliku pakietu APK

Jeśli aplikacja jest prawidłowo uruchamiana, można użyć narzędzia ADL do spakowania aplikacji do pliku pakietu APK. Plik pakietu APK jest natywnym formatem pliku systemu Android, który można rozpowszechniać wśród użytkowników.

Wszystkie aplikacje dla systemu Android muszą być podpisywane. W przeciwieństwie do plików AIR aplikacje dla systemu Android są zwyczajowo podpisywane za pomocą certyfikatu podpisanego automatycznie. System operacyjny Android nie próbuje ustalić tożsamości autora aplikacji. Do podpisywania pakietów dla systemu Android można używać certyfikatu utworzonego przez narzędzie ADT. Certyfikaty używane na potrzeby aplikacji wysyłanych do sklepu Android Market muszą być ważne przez co najmniej 25 lat.

Generowanie pary: certyfikat samopodpisany i klucz

- ❖ Do wiersza poleceń wprowadź następujące polecenie (plik wykonywalny ADT można znaleźć w katalogu bin pakietu Flex SDK):

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

W tym przykładzie zastosowano minimalną liczbę atrybutów, jakie można ustawić dla certyfikatu. Typem klucza musi być `1024-RSA` lub `2048-RSA` (zobacz „[Polecenie certificate narzędzia ADT](#)” na stronie 184).

Tworzenie pakietu AIR

- ❖ W wierszu polecenia wpisz następujące polecenie (w pojedynczym wierszu):

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

Zostanie wyświetlona zachęta do wprowadzenia hasła pliku kluczy. Wpisz hasło i naciśnij klawisz Enter.

Więcej informacji zawiera rozdział „[Pakowanie aplikacji AIR dla urządzeń przenośnych](#)” na stronie 99.

Instalowanie środowiska wykonawczego AIR

Na urządzeniu można zainstalować najnowszą wersję środowiska wykonawczego AIR ze sklepu Android Market. Środowisko wykonawcze będące częścią zestawu SDK można zainstalować na urządzeniu lub w emulatorze systemu Android.

- ❖ W wierszu poleceń wpisz następujące polecenie (w pojedynczym wierszu):

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

Ustaw flagę `-platformsdk` wskazującą katalog zestawu SDK środowiska Android. (Określ folder nadrzędny folderu `tools`).

Narzędzie ADT zainstaluje plik `Runtime.apk` będący częścią zestawu SDK.

Więcej informacji zawiera sekcja „[Instalowanie środowiska wykonawczego AIR i aplikacji na potrzeby programowania](#)” na stronie 115.

Instalowanie aplikacji AIR

❖ W wierszu poleceń wpisz następujące polecenie (w pojedynczym wierszu):

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

Ustaw flagę `-platformsdk` wskazującą katalog zestawu SDK środowiska Android. (Określ folder nadrzędny folderu `tools`).

Więcej informacji zawiera sekcja „[Instalowanie środowiska wykonawczego AIR i aplikacji na potrzeby programowania](#)” na stronie 115.

Aplikację można uruchomić, dotykając ikony aplikacji na ekranie urządzenia lub emulatora.

Rozdział 6: Programowanie aplikacji AIR na komputery stacjonarne

Obieg pracy dotyczący programowania aplikacji AIR na komputery stacjonarne

Podstawowy obieg pracy dotyczący programowania aplikacji AIR jest taki sam jak w przypadku większości tradycyjnych modeli programowania: Obejmuje pisanie kodu, kompilowanie, testowanie i, w końcowej części cyklu, pakowanie do pliku instalatora.

Kod aplikacji można pisać w technologii Flash, Flex i ActionScript oraz kompilować za pomocą programów Flash Professional i Flash Builder lub za pomocą kompilatorów wiersza poleceń mxmlec i compc. Kod aplikacji można również pisać w technologii HTML i JavaScript, co pozwala ominąć etap kompilacji.

Aplikacje AIR na komputery stacjonarne można testować za pomocą narzędzia ADL, które pozwala uruchamiać aplikacje bez konieczności ich wcześniejszego pakowania i instalowania. Programy Flash Professional, Flash Builder i Dreamweaver oraz środowisko programistyczne Aptana oferują zintegrowany debugger Flash. Korzystając z narzędzia ADL z poziomu wiersza poleceń, można również ręcznie uruchomić narzędzie do debugowania — program FDB. Samo narzędzie ADL wyświetla błędy i wyniki działania instrukcji trace.

Każda aplikacja AIR musi zostać spakowana do pliku instalacyjnego. Zalecany jest format pliku AIR zgodny z wieloma platformami, z wyjątkiem następujących przypadków:

- Konieczny jest dostęp do interfejsów API zależnych od platformy, takich jak klasa NativeProcess.
- Aplikacja korzysta z rozszerzeń natywnych.

W takich sytuacjach aplikację AIR można spakować w postaci przeznaczonego dla konkretnej platformy pliku instalatora natywnego.

Aplikacje oparte na plikach SWF

- 1 Napisz kod MXML lub ActionScript.
- 2 Utwórz wymagane zasoby, takie jak pliki bitmap ikon.
- 3 Utwórz deskryptor aplikacji.
- 4 Skompiluj kod ActionScript.
- 5 Przetestuj aplikację.
- 6 Spakuj i podpisz aplikację jako plik AIR przy użyciu typu docelowego *air*.

Aplikacje oparte na plikach HTML

- 1 Napisz kod HTML i JavaScript.
- 2 Utwórz wymagane zasoby, takie jak pliki bitmap ikon.
- 3 Utwórz deskryptor aplikacji.
- 4 Przetestuj aplikację.

- 5 Spakuj i podpisz aplikację jako plik AIR przy użyciu typu docelowego *air*.

Tworzenie instalatorów natywnych dla aplikacji AIR

- 1 Napisz kod (ActionScript lub HTML i JavaScript).
- 2 Utwórz wymagane zasoby, takie jak pliki bitmap ikon.
- 3 Utwórz deskryptor aplikacji, określając profil *extendedDesktop*.
- 4 Skompiluj kod ActionScript, jeśli występuje.
- 5 Przetestuj aplikację.
- 6 Spakuj aplikację na każdej platformie docelowej, używając typu docelowego *native*.

Uwaga: Instalator natywny dla danej platformy docelowej musi zostać utworzony na tej platformie. Nie można na przykład utworzyć instalatora dla systemu Windows na komputerze Mac. Do uruchamiania wielu platform na tym samym komputerze można użyć maszyny wirtualnej, takiej jak oprogramowanie firmy VMWare.

Tworzenie aplikacji AIR przy użyciu pakietu z dołączonym środowiskiem wykonawczym

- 1 Napisz kod (ActionScript lub HTML i JavaScript).
- 2 Utwórz wymagane zasoby, takie jak pliki bitmap ikon.
- 3 Utwórz deskryptor aplikacji, określając profil *extendedDesktop*.
- 4 Skompiluj kod ActionScript, jeśli występuje.
- 5 Przetestuj aplikację.
- 6 Spakuj aplikację na każdej platformie docelowej, używając typu docelowego *bundle*.
- 7 Utwórz program instalacyjny przy użyciu plików pakietu. (Zestaw SDK środowiska AIR nie zawiera narzędzi do tworzenia takiego instalatora, ale dostępnych jest wiele zestawów narzędzi innych firm).

Uwaga: Pakiet dla danej platformy docelowej musi zostać utworzony na tej platformie. Nie można na przykład utworzyć pakietu dla systemu Windows na komputerze Mac. Do uruchamiania wielu platform na tym samym komputerze można użyć maszyny wirtualnej, takiej jak oprogramowanie firmy VMWare.

Ustawianie właściwości aplikacji na komputery stacjonarne

Podstawowe właściwości aplikacji są ustawiane w pliku deskryptora aplikacji. W tej sekcji są opisane właściwości odnoszące się do aplikacji AIR na komputery stacjonarne. Elementy pliku deskryptora aplikacji są szczegółowo opisane w sekcji „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Wymagana wersja środowiska wykonawczego AIR

Wersję środowiska wykonawczego AIR, która jest wymagana przez aplikację, należy określić przy użyciu przestrzeni nazw pliku deskryptora aplikacji.

Przestrzeń nazw (przypisana w elemencie `application`) w dużej mierze określa funkcje, których aplikacja może używać. Jeśli na przykład aplikacja korzysta z przestrzeni nazw AIR 1.5, a użytkownik zainstalował środowisko AIR 3.0, wówczas aplikacja rozpoznaje zachowanie środowiska AIR 1.5 (nawet jeśli to zachowanie zostało zmienione w środowisku AIR 3.0). Dopiero zmiana przestrzeni nazw i opublikowanie aktualizacji umożliwiają aplikacji uzyskanie dostępu do nowych zachowań i funkcji. Głównymi wyjątkami od tej zasady są zmiany dotyczące zabezpieczeń i mechanizmu WebKit.

Przestrzeń nazw należy określić przy użyciu atrybutu `xmlns` głównego elementu `application`.

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

Więcej tematów Pomocy

„[application](#)” na stronie 223

Tożsamość aplikacji

Niektóre ustawienia powinny być niepowtarzalne dla każdej publikowanej aplikacji. Tymi unikatowymi ustawieniami są między innymi identyfikator, nazwa i nazwa pliku.

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

Więcej tematów Pomocy

„[id](#)” na stronie 239

„[filename](#)” na stronie 234

„[name](#)” na stronie 247

Wersja aplikacji

W środowisku AIR w wersji wcześniejszej niż 2.5 wersja aplikacji jest określana w elemencie `version`. Można w tym celu użyć dowolnego ciągu. Środowisko wykonawcze AIR nie interpretuje tego ciągu. Wersja „2.0” nie jest traktowana jako wersja wyższa niż „1.0”.

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

W środowisku AIR 2.5 lub nowszym wersję aplikacji należy określić w elemencie `versionNumber`. Nie można już stosować elementu `version`. Określając wartość elementu `versionNumber`, należy użyć sekwencji składającej się z maksymalnie trzech liczb rozdzielonych kropkami, na przykład „0.1.2”. Każda z tych liczb w numerze wersji może zawierać do trzech cyfr. Innymi słowy, największym dozwolonym numerem wersji jest „999.999.999”. Numer wersji nie musi zawierać wszystkich trzech liczb. Zarówno „1”, jak i „1.0” stanowią prawidłowe numery wersji.

Przy użyciu elementu `versionLabel` można określić etykietę wersji. W przypadku dodania etykiety wersji jest ona wyświetlana zamiast numeru wersji w takich miejscach jak okna dialogowe instalatora aplikacji AIR.

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

Więcej tematów Pomocy

„[version](#)” na stronie 255

„[versionLabel](#)” na stronie 256

„[versionNumber](#)” na stronie 256

Właściwości okna głównego

Gdy środowisko AIR uruchamia aplikację na komputerze stacjonarnym, tworzy okno, do którego wczytuje główny plik SWF lub stronę HTML. Do kontroli początkowego wyglądu i zachowania tego początkowego okna aplikacji środowisko AIR używa elementów potomnych elementu `initialWindow`.

- **content** — Główny plik SWF aplikacji w elemencie potomnym `content` elementu `initialWindow`. Jeśli aplikacja jest tworzona dla urządzeń docelowych o profilu komputerowy, można użyć pliku SWF lub HTML.

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

Plik musi zostać umieszczony w pakiecie aplikacji AIR (przy użyciu narzędzia ADT lub środowiska programistycznego). Samo umieszczenie odniesienia do nazwy w deskrytorze aplikacji nie powoduje automatycznego zawarcia pliku w pakiecie.

- **depthAndStencil** — Określa, czy ma być używany bufor głębi czy bufor szablonu. Te bufony są zazwyczaj używane podczas pracy z zawartością 3D.

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

- **height** — Wysokość okna początkowego.
- **maximizable** — Określa, czy jest wyświetlana opcja karnacji systemowej służąca do maksymalizowania okna.
- **maxSize** — Maksymalny dozwolony rozmiar.
- **minimizable** — Określa, czy jest wyświetlana opcja karnacji systemowej służąca do minimalizowania okna.
- **minSize** — Minimalny dozwolony rozmiar okna.
- **renderMode** — W środowisku AIR 3 lub nowszym można ustawić tryb renderowania *auto*, *cpu*, *direct* lub *gpu* w przypadku aplikacji dla komputerów. W starszych wersjach środowiska AIR to ustawienie jest ignorowane na komputerach. Ustawienia `renderMode` nie można zmienić w czasie wykonywania.

- **auto** — Działanie jest zasadniczo takie same jak w przypadku trybu *cpu*.
- **cpu** — Wyświetlane obiekty są renderowane i kopiowane do pamięci wyświetlania przy użyciu oprogramowania. Klasa `StageVideo` jest dostępna tylko w przypadku wyświetlania okna w trybie pełnoekranowym. Klasa `Stage3D` korzysta z renderowania programowego.
- **direct** — Wyświetlane obiekty są renderowane przez oprogramowanie środowiska wykonawczego, ale kopiowanie renderowanych klatek do pamięci wyświetlania (kopiowanie bitowe) jest przyspieszane sprzętowo. Klasa `StageVideo` jest dostępna. Klasa `Stage3D` korzysta z przyspieszania sprzętowego, jeśli jest to możliwe. Jeśli dla przezroczystości okien ustawiono wartość `true`, stosowane jest programowe renderowanie i kopiowanie bitowe okien.

Uwaga: Aby można było stosować przyspieszenie GPU zawartości Flash w środowisku AIR dla platform przenośnych, firma Adobe zaleca, aby zamiast opcji `renderMode="direct"` (obiektu `Stage3D`) użyć opcji `renderMode="gpu"`. Firma Adobe oficjalnie oferuje pomoc techniczną dla następujących architektur opartych na obiekcie `Stage3D` i zaleca ich stosowanie: *Starling (2D)* i *Away3D (3D)*. Więcej informacji o architekturach `Stage3D` oraz *Starling/Away3D* znajduje się w dokumencie <http://gaming.adobe.com/getstarted/>.

- Wartość `gpu` powoduje, że przyspieszanie sprzętowe jest stosowane, jeśli jest dostępne.
- **`requestedDisplayResolution`** Wskazuje, czy aplikacja powinna używać rozdzielczości *standard* (standardowej) czy *high* (wysokiej) na komputerach MacBook Pro z ekranami o wysokiej rozdzielczości. Ta wartość jest ignorowana na innych platformach. W przypadku wartości *standard* każdy piksel stołu montażowego jest renderowany jako cztery piksele na ekranie. W przypadku wartości *high* każdy piksel stołu montażowego odpowiada pojedynczemu fizycznemu pikselowi na ekranie. Ustawiona wartość jest używana dla wszystkich okien aplikacji. Od wersji 3.6 środowiska AIR można używać elementu `requestedDisplayResolution` dla aplikacji komputerowych AIR (jako elementu potomnego elementu `initialWindow`).
- **`resizable`** — Określa, czy jest wyświetlana opcja karnacji systemowej służąca do zmiany rozmiaru okna.
- **`systemChrome`** — Określa, czy jest używany standardowy wygląd okna systemu operacyjnego. W czasie wykonywania nie można zmieniać ustawienia `systemChrome` okna.
- **`title`** — Tytuł okna.
- **`transparent`** — Określa, czy względem okna jest stosowane mieszanie alfa z tłem. W przypadku włączenia przezroczystości nie można używać karnacji systemowej. W czasie wykonywania nie można zmieniać ustawienia `transparent` okna.
- **`visible`** — Określa, czy okno jest widoczne od razu po utworzeniu. Domyślnie okno nie jest od razu widoczne, aby aplikacja mogła narysować jego zawartość, zanim stanie się widoczne.
- **`width`** — Szerokość okna.
- **`x`** — Pozycja okna w poziomie.
- **`y`** — Pozycja okna w pionie.

Więcej tematów Pomocy

[„content”](#) na stronie 228

[„depthAndStencil”](#) na stronie 230

[„height”](#) na stronie 238

[„maximizable”](#) na stronie 246

[„maxSize”](#) na stronie 246

[„minimizable”](#) na stronie 247

[„minimizable”](#) na stronie 247

[„minSize”](#) na stronie 247

[„renderMode”](#) na stronie 250

[„requestedDisplayResolution”](#) na stronie 250

[„resizable”](#) na stronie 251

[„systemChrome”](#) na stronie 254

[„title”](#) na stronie 255

[„transparent”](#) na stronie 255

[„visible”](#) na stronie 257

[„width”](#) na stronie 257

„x” na stronie 257

„y” na stronie 258

Funkcje na komputerze stacjonarnym

Poniższe elementy kontrolują funkcje aktualizowania i instalowania na komputerze stacjonarnym.

- `customUpdateUI` — Umożliwia zastosowanie własnych okien przeznaczonych do aktualizowania aplikacji. Jeśli dla tego elementu jest ustawiona wartość `false` (domyślna), wówczas są używane standardowe okna dialogowe środowiska AIR.
- `fileTypes` — Określa typy plików, dla których aplikacja ma zostać zarejestrowana jako domyślna aplikacja służąca do otwierania. Jeśli dla danego typu plików występuje już domyślna aplikacja otwierająca, środowisko AIR nie nadpisuje istniejącego zarejestrowanego ustawienia. Aplikacja może jednak nadpisać ustawienie rejestracji w czasie wykonywania, korzystając z metody `setAsDefaultApplication()` obiektu `NativeApplication`. Zalecane jest pytanie o zezwolenie użytkownika przed nadpisaniem istniejących powiązań z typami plików.

***Uwaga:** Rejestrowanie typów plików jest ignorowane podczas tworzenia pakietu aplikacji z dołączonym środowiskiem wykonawczym (przy użyciu typu docelowego `-bundle`). Aby zarejestrować dany typ pliku, należy utworzyć program instalacyjny wykonujący rejestrację.*

- `installFolder` — Określa ścieżkę, w której jest instalowana aplikacja, względem standardowego folderu instalacyjnego aplikacji. Za pomocą tego ustawienia można określić własną nazwę folderu, a także pogrupować wiele aplikacji we wspólnym folderze.
- `programMenuFolder` — Określa hierarchię menu na potrzeby menu Wszystkie programy systemu Windows. Za pomocą tego ustawienia można pogrupować wiele aplikacji we wspólnym menu. Jeśli nie zostanie określony żaden folder, skrót do aplikacji zostanie dodany bezpośrednio do menu głównego.

Więcej tematów Pomocy

„`customUpdateUI`” na stronie 229

„`fileTypes`” na stronie 236

„`installFolder`” na stronie 243

„`programMenuFolder`” na stronie 249

Obsługiwane profile

Jeśli aplikacja jest przydatna wyłącznie na komputerze stacjonarnym, można uniemożliwić zainstalowanie jej na urządzeniu o innym profilu, wykluczając go z listy obsługiwanych profili. Jeśli aplikacja korzysta z klasy `NativeProcess` lub rozszerzeń natywnych, musi być obsługiwany profil `extendedDesktop`.

Jeśli element `supportedProfile` nie jest częścią deskryptora aplikacji, wówczas obowiązuje założenie, że aplikacja obsługuje wszystkie zdefiniowane profile. W celu ograniczenia aplikacji do konkretnej listy profili należy podać te profile, oddzielając je odstępami.

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

Listę klas języka ActionScript obsługiwanych w profilach `desktop` i `extendedDesktop` można znaleźć w sekcji „[Możliwości różnych profili](#)” na stronie 260.

Więcej tematów Pomocy

„`supportedProfiles`” na stronie 253

Wymagane rozszerzenia natywne

Aplikacje obsługujące profil `extendedDesktop` mogą korzystać z rozszerzeń natywnych.

Wszystkie rozszerzenia natywne używane przez aplikację AIR należy zadeklarować w deskrytorze aplikacji. W poniższym przykładzie przedstawiono składnię umożliwiającą określenie dwóch wymaganych rozszerzeń natywnych:

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

Element `extensionID` ma wartość taką samą jak element `id` w pliku deskryptora rozszerzenia. Plik deskryptora rozszerzenia jest plikiem XML o nazwie `extension.xml`. Jest on spakowany w pliku ANE otrzymanym od programisty rozszerzenia natywnego.

Ikony aplikacji

Na komputerze stacjonarnym ikony określone w deskrytorze aplikacji są stosowane jako ikony pliku aplikacji, skrótu i menu programów. Ikona aplikacji powinna zostać udostępniona w postaci obrazów PNG o rozmiarach 16x16, 32x32, 48x48 i 128x128 pikseli. Ścieżkę do plików ikon należy określić w elemencie `icon` pliku deskryptora aplikacji.

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

Jeśli nie zostanie udostępniona ikona w danym rozmiarze, zostanie użyta większa ikona o najbliższym rozmiarze, która zostanie przeskalowana w celu dopasowania do wymaganej wielkości. W przypadku braku jakichkolwiek ikon jest używana domyślna ikona systemowa.

Więcej tematów Pomocy

„[icon](#)” na stronie 239

„[imageNxN](#)” na stronie 240

Ustawienia ignorowane

Aplikacje na komputerze stacjonarnym ignorują ustawienia aplikacji, które odnoszą się do funkcji profilu urządzeń przenośnych. Ignorowane są następujące ustawienia:

- `android`
- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `iPhone`
- `renderMode` (wersje starsze niż AIR 3)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Debugowanie aplikacji AIR na komputery stacjonarne

Narzędzia do debugowania są zazwyczaj wbudowane w środowiska programistyczne, takie jak oprogramowanie Flash Builder, Flash Professional czy Dreamweaver. Aplikację można również debugować, uruchamiając ją w trybie debugowania. W przypadku korzystania ze środowiska programistycznego, które nie obsługuje debugowania bezpośrednio, pomoc w debugowaniu aplikacji można uzyskać, używając narzędzi AIR Debug Launcher (ADL) i Flash Debugger (FDB).

Więcej tematów Pomocy

[De Monsters: Monster Debugger](#)

„[Debugowanie za pomocą introspektora HTML w środowisku AIR](#)” na stronie 298

Uruchamianie aplikacji za pomocą narzędzia ADL

Korzystając z narzędzia ADL, można uruchamiać aplikacje AIR bez pakowania i instalowania. Plik deskryptora aplikacji należy przekazać do narzędzia ADL jako parametr, zgodnie z poniższym przykładem. (Najpierw należy skompilować kod ActionScript w aplikacji).

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

Narzędzie ADL wyświetla instrukcje trace, wyjątki środowiska wykonawczego i błędy analizowania kodu HTML w oknie terminala. Jeśli proces FDB czeka na połączenie przychodzące, narzędzie ADL łączy się z debugerem.

Aplikacje korzystające z rozszerzeń natywnych można również debugować za pomocą narzędzia ADL. Na przykład:

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

Więcej tematów Pomocy

„[AIR Debug Launcher \(ADL\)](#)” na stronie 169

Wyświetlanie instrukcji trace

W celu wyświetlenia instrukcji trace w konsoli służącej do uruchamiania ADL należy dodać instrukcje trace do kodu za pomocą funkcji `trace()`.

Uwaga: Jeśli instrukcje `trace()` nie są wyświetlane w konsoli, upewnij się, że w pliku `mm.cfg` nie jest ustawiony parametr `ErrorReportingEnable` ani `TraceOutputFileEnable`. Więcej informacji o lokalizacji tego pliku na poszczególnych platformach zawiera artykuł [Edytowanie pliku mm.cfg](#).

Przykład w języku ActionScript:

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

Przykład w języku JavaScript:

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

W JavaScript można korzystać z funkcji `alert()` i `confirm()` w celu wyświetlania komunikatów debugowania z aplikacji. Ponadto w konsoli wyświetlane są numery linii z błędami składni, podobnie jak niewychwycone wyjątki JavaScript.

Uwaga: W celu użycia przedrostka `air` przedstawionego w przykładzie w języku JavaScript należy zaimportować plik `AIRAliases.js` na stronę. Plik znajduje się w katalogu `frameworks` pakietu AIR SDK.

Połączenie z debugerem Flash Debugger (FDB)

Aby debugować aplikację AIR za pomocą debugera Flash Debugger, należy uruchomić sesję FDB, a następnie uruchomić aplikację za pomocą programu ADL.

***Uwaga:** W aplikacjach AIR opartych na plikach SWF pliki źródłowe ActionScript muszą być kompilowane z flagą `-debug`. (W programie Flash Professional należy zaznaczyć opcję *Pozwól debugować w oknie dialogowym Ustawienia publikowania*).*

- 1 Uruchom program FDB. Program FDB znajduje się w katalogu `bin` pakietu Flex SDK.

Konsola wyświetli monit programu FDB: `<fdb>`

- 2 Wykonaj polecenie `run: <fdb>run [Enter]`

- 3 W innej konsoli poleceń lub powłóce uruchom wersję aplikacji po debugowaniu:

```
adl myApp.xml
```

- 4 Za pomocą poleceń FDB ustaw punkty zatrzymania zgodnie z potrzebami.

- 5 Wpisz: `continue [Enter]`

Jeśli aplikacja ma format SWF, debuger steruje jedynie wykonaniem kodu ActionScript. Jeśli aplikacja ma format HTML, wówczas debuger steruje jedynie wykonaniem kodu JavaScript.

Aby uruchomić program ADL bez połączenia z debugerem, należy dołączyć opcję `-nodebug`:

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

Aby uzyskać podstawowe informacje na temat poleceń programu FDB, należy wywołać polecenie `help`:

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

Szczegółowe informacje na temat poleceń programu FDB zawiera sekcja [Korzystanie z poleceń debugera wiersza poleceń](#) w dokumentacji środowiska Flex.

Pakowanie pliku instalacyjnego AIR na komputery stacjonarne

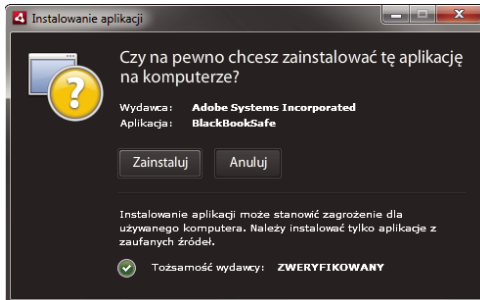
Każda aplikacja AIR musi zawierać przynajmniej plik deskryptora aplikacji oraz główny plik SWF lub HTML. Wszelkie inne zasoby przeznaczone do zainstalowania z aplikacją również muszą zostać umieszczone w pakiecie pliku AIR.

W tym artykule omówiono pakowanie aplikacji AIR za pomocą narzędzi wiersza poleceń znajdujących się w zestawie SDK. Informacje na temat pakowania aplikacji za pomocą jednego z narzędzi firmy Adobe do opracowywania zawartości można znaleźć w następujących materiałach:

- Adobe® Flex® Builder™, patrz [Pakowanie aplikacji AIR za pomocą programu Flex Builder](#).
- Adobe® Flash® Builder™, patrz [Pakowanie aplikacji AIR za pomocą programu Flash Builder](#).
- Adobe® Flash® Professional — dokument [Publikowanie dla środowiska Adobe AIR](#).
- Adobe® Dreamweaver® — dokument [Tworzenie aplikacji AIR w programie Dreamweaver](#).

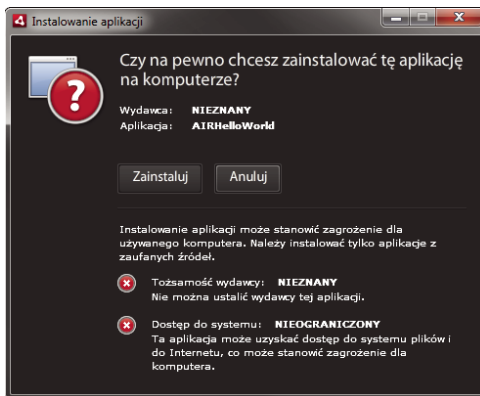
Wszystkie pliki instalatora AIR muszą być podpisane certyfikatem elektronicznym. Instalator AIR korzysta z podpisu w celu sprawdzenia, czy po podpisaniu plik aplikacji nie został zmodyfikowany. Użytkownik może użyć certyfikatu podpisującego kod wydany przez ośrodek certyfikacji lub certyfikatu samopodpisanego.

Jeśli używany jest certyfikat wydany przez zaufany ośrodek certyfikacji, ten certyfikat stanowi dla użytkowników aplikacji poświadczenie tożsamości wydawcy. Okno dialogowe instalacji odzwierciedla fakt, że tożsamość wydawcy została zatwierdzona przez ośrodek certyfikacji:



Okno dialogowe instalacji dla aplikacji podpisanych za pomocą zaufanego certyfikatu

Gdy stosowany jest certyfikat samopodpisany, użytkownicy nie mogą sprawdzić tożsamości wydawcy, jako strony podpisującej. Certyfikat samopodpisany dodatkowo zmniejsza pewność, że pakiet nie został zmodyfikowany. (Jest tak dlatego, że prawowity plik instalacyjny mógł zostać zamieniony na falsyfikat zanim dotarł do użytkownika). Okno dialogowe zawiera informację o tym, że nie można sprawdzić tożsamości wydawcy. Użytkownicy instalujący taką aplikację podejmują większe ryzyko:



Okno dialogowe instalacji dla aplikacji podpisanych za pomocą certyfikatu samopodpisanego

Plik AIR można spakować i podpisać w jednym kroku, korzystając z polecenia `-package` narzędzia ADT. Można również utworzyć pośredni, niepodpisany pakiet za pomocą polecenia `-prepare`, a następnie podpisać go za pomocą polecenia `-sign` w osobnym kroku.

Uwaga: Środowisko Java w wersji 1.5 i wyższych nie akceptuje znaków o wysokich kodach ASCII w hasłach chroniących pliki certyfikatów PKCS12. Podczas tworzenia lub eksportowania pliku certyfikatu podpisującego kod należy używać w hasła zwykłych znaków ASCII.

Podczas podpisywania pakietu instalacyjnego narzędzie ADT automatycznie łączy się z serwerem znaczników czasu w celu weryfikacji czasu. Informacje znacznika czasu są dołączone do pliku AIR. Plik AIR, który zawiera zweryfikowany znacznik czasu, może zostać zainstalowany w dowolnym momencie w przyszłości. Jeśli narzędzie ADT nie może nawiązać połączenia z serwerem znaczników czasu, wówczas pakowanie zostaje anulowane. Opcję ustawiania znaczników czasu można anulować, ale bez znacznika czasu zainstalowanie aplikacji AIR będzie niemożliwe po utracie ważności certyfikatu służącego do podpisania pliku instalacyjnego.

W przypadku tworzenia pakietu przeznaczonego do aktualizacji istniejącej aplikacji AIR pakiet musi zostać podpisany tym samym certyfikatem, co oryginalna aplikacja. Jeśli oryginalny certyfikat został odnowiony lub utracił ważność w ciągu ostatnich 180 dni, lub jeśli wydawca chce zmienić certyfikat na nowy, wówczas można zastosować podpis migracji. Podpis migracji obejmuje podpisanie pliku aplikacji AIR za pomocą nowego i starego certyfikatu. Za pomocą polecenia `-migrate` można zastosować podpis migracji, zgodnie z opisem w sekcji „[Polecenie migrate narzędzia ADT](#)” na stronie 184.

Ważne: Po utracie ważności oryginalnego certyfikatu istnieje okres karencji wynoszący równo 180 dni, w trakcie którego można zastosować podpis migracji. W przypadku braku podpisu migracji istniejący użytkownicy będą musieli odinstalować istniejącą aplikację przed zainstalowaniem nowej wersji. Okres karencji dotyczy tylko aplikacji utworzonych w środowisku AIR w wersji 1.5.3 i wyższych wersjach, w przestrzeni nazw deskryptora aplikacji. W przypadku wcześniejszych wersji środowiska aplikacji AIR nie obowiązuje okres karencji.

W wersjach przed AIR 1.1 podpisy migracji nie były obsługiwane. W celu zastosowania podpisu migracji należy spakować aplikację za pomocą narzędzia SDK w wersji 1.1 lub późniejszej.

Aplikacje wdrożone z wykorzystaniem plików AIR są znane jako aplikacje o profilu Stacjonarny. Narzędzia ADT nie można użyć do spakowania rodzimego instalatora dla aplikacji AIR, jeśli plik deskryptora aplikacji nie obsługuje profilu Stacjonarny. Profil można ograniczyć za pomocą elementu `supportedProfiles` w pliku deskryptora aplikacji. Więcej informacji zawierają sekcje „[Profil urządzeń](#)” na stronie 259 i „[supportedProfiles](#)” na stronie 253.

Uwaga: Ustawienia pliku deskryptora aplikacji określają tożsamość aplikacji AIR oraz jej domyślną ścieżkę instalacyjną. Zobacz „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Identyfikatory wydawcy

Począwszy od wersji AIR 1.5.3 identyfikatory wydawcy stały się nieaktualne. Nowe aplikacje (początkowo publikowane za pomocą AIR 1.5.3 lub w wersjach późniejszych) nie potrzebują i nie powinny określać identyfikatora wydawcy.

W przypadku aktualizowania aplikacji opublikowanych we wcześniejszych wersjach AIR należy określić identyfikator oryginalnego wydawcy w pliku deskryptora aplikacji. W przeciwnym wypadku zainstalowana wersja aplikacji oraz wersja z aktualizacją będą traktowane jako różne aplikacje. Jeśli zostanie zastosowany inny identyfikator lub pominięty zostanie znacznik `publisherID`, wówczas przed zainstalowaniem nowej wersji użytkownik będzie musiał odinstalować wcześniejszą wersję.

Aby ustalić oryginalny identyfikator wydawcy, należy odszukać plik `publisherid` w podkatalogu META-INF/AIR, w którym zainstalowana jest oryginalna aplikacja. Ciąg znaków w tym pliku jest identyfikatorem wydawcy. Deskryptor aplikacji powinien określać środowisko wykonawcze AIR 1.5.3 (lub późniejszą wersję) w deklaracji przestrzeni nazw pliku deskryptora aplikacji — tylko wówczas możliwe jest ręczne określenie identyfikatora wydawcy.

W przypadku aplikacji opublikowanych przed wersją AIR 1.5.3 — oraz opublikowanych za pomocą AIR 1.5.3 SDK, jeśli w przestrzeni nazw deskryptora aplikacji określono wcześniejszą wersję AIR — identyfikator wydawcy jest określany na podstawie certyfikatu podpisującego. Ten identyfikator jest używany wraz z identyfikatorem aplikacji w celu określenia tożsamości aplikacji. Identyfikator wydawcy — jeśli jest dostępny — jest używany do następujących celów:

- Sprawdzanie, czy plik AIR jest aktualizacją, a nie nową aplikacją do zainstalowania
- Jako część klucza szyfrowania dla zaszyfrowanej składnicy lokalnej
- Jako część ścieżki katalogu zapisu aplikacji
- Jako część ciągu znaków połączenia dla połączeń lokalnych
- Jako część ciągu znaków tożsamości używanego w celu wywołania aplikacji z interfejsem API AIR dostępnym w przeglądarce

- Jako część OSID (używany podczas tworzenia niestandardowych programów instalacyjnych/deinstalacyjnych)

Przed wersją AIR 1.5.3 identyfikator wydawcy aplikacji mógł zostać zmieniony, jeśli aktualizację aplikacji podpisano za pomocą podpisu migracji przy użyciu nowego lub odnowionego certyfikatu. Gdy dochodzi do zmiany identyfikatora wydawcy, następuje również zmiana działania wszystkich funkcji AIR zależnych od tego identyfikatora. Na przykład: dochodzi do utraty dostępu do danych w istniejącej składnicy lokalnej, a wszelkie instancje Flash lub AIR, które tworzą lokalne połączenie z aplikacją, muszą stosować nowy identyfikator w ciągu znaków połączenia.

W wersji AIR 1.5.3 i wersjach późniejszych identyfikator wydawcy nie jest oparty na certyfikacie podpisującym i jest przypisywany tylko wówczas, gdy deskryptor aplikacji zawiera znacznik publisherID. Aplikacja nie może zostać zaktualizowana, jeśli identyfikator wydawcy dla zaktualizowanego pakietu AIR nie jest zgodny z aktualnym identyfikatorem wydawcy.

Pakowanie przy użyciu narzędzia ADT

Aplikacje AIR można pakować za pomocą narzędzia wiersza poleceń ADT środowiska AIR. Przed przystąpieniem do pakowania należy skompilować kod ActionScript, MXML oraz kod wszystkich rozszerzeń. Jest również potrzebny certyfikat podpisywania kodu.

Szczegółową dokumentację poleceń i opcji narzędzia ADT zawiera sekcja „[Narzędzie ADT](#)” na stronie 175.

Tworzenie pakietu AIR

Aby utworzyć pakiet AIR, należy użyć polecenia package narzędzia ADT, ustawiając dla typu docelowego wersji do rozpowszechniania wartość *air*.

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

W tym przykładzie przyjęto, że ścieżka do narzędzia ADT znajduje się w definicji ścieżki powłoki wiersza poleceń. (W celu uzyskania pomocy zapoznaj się z tematem „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Pliki aplikacji użyte w tym przykładzie to myApp-app.xml (plik deskryptora aplikacji), myApp.swf i katalog ikon.

Po uruchomieniu polecenia w przedstawiony sposób narzędzie ADT wyświetli monit o podanie hasła magazynu kluczy. (Wpisywane znaki hasła czasami nie są wyświetlane. Po zakończeniu wpisywania wystarczy nacisnąć klawisz Enter).

Tworzenie pakietu AIR z pliku AIRI

W celu uzyskania pakietu AIR z możliwością instalowania można utworzyć i podpisać plik AIRI.

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

Pakowanie instalatora natywnego dla komputerów

W środowisku AIR 2 lub nowszym można za pomocą narzędzia ADT tworzyć instalatory macierzyste przeznaczone do rozpowszechniania aplikacji AIR. Na przykład: można zbudować plik instalatora EXE w celu dystrybuowania aplikacji AIR w systemach Windows. Można również zbudować plik instalatora DMG w celu dystrybuowania aplikacji AIR w systemie Mac OS. W środowiskach AIR 2.5 i AIR 2.6 można utworzyć plik instalatora DEB lub RPM na potrzeby rozpowszechniania aplikacji AIR w systemach Linux.

Aplikacje zainstalowane za pomocą rodzimego instalatora są znane jako aplikacje o profilu Rozszerzony stacjonarny. Narzędzia ADT nie można użyć do spakowania rodzimego instalatora dla aplikacji AIR, jeśli plik deskryptora aplikacji nie obsługuje profilu Rozszerzony stacjonarny. Profil można ograniczyć za pomocą elementu `supportedProfiles` w pliku deskryptora aplikacji. Więcej informacji zawierają sekcje „[Profile urządzeń](#)” na stronie 259 i „[supportedProfiles](#)” na stronie 253.

Istnieją dwa podstawowe sposoby tworzenia aplikacji AIR w wersji z rodzimym instalatorem:

- Rodzimy instalator można utworzyć na podstawie pliku deskryptora aplikacji i innych plików źródłowych. (Do innych plików źródłowych mogą należeć pliki SWF, pliki HTML oraz inne zasoby).
- Rodzimy instalator można utworzyć na podstawie pliku AIR lub pliku AIRI.

Narzędzie ADT musi być używane na tym samym systemie operacyjnym, na którym ma działać plik rodzimego instalatora, jaki ma zostać wygenerowany. Więc w celu utworzenia pliku EXE dla systemu Windows narzędzie ADT należy uruchomić w systemie Windows. W celu utworzenia pliku DMG dla systemu Mac OS należy uruchomić narzędzie ADT w systemie Mac OS. W celu utworzenia pliku DEB lub RPM dla systemu Linux należy uruchomić narzędzie ADT w systemie Linux z poziomu zestawu SDK środowiska AIR 2.6.

Po utworzeniu rodzimego instalatora przeznaczonego do dystrybuowania aplikacji AIR aplikacja uzyskuje następujące możliwości:

- Może uruchamiać rodzime procesy i wchodzić w interakcje z tymi procesami, korzystając z klasy `NativeProcess`. Szczegółowe informacje zawierają następujące zasoby:
 - [Komunikacja z rodzimymi procesami w AIR](#) (dla programistów ActionScript)
 - [Komunikacja z rodzimymi procesami w AIR](#) (dla programistów HTML)
- Może korzystać z rozszerzeń natywnych.
- Może użyć metody `File.openWithDefaultApplication()` do otwarcia dowolnego pliku za pomocą domyślnej aplikacji systemowej, która według definicji służy do otwierania tego pliku — bez względu na typ pliku. (Istnieją pewne ograniczenia dotyczące aplikacji, które *nie* zostały zainstalowane za pomocą rodzimego instalatora. Szczegółowe informacje zawiera wpis dotyczący `File.openWithDefaultApplication()` w skorowidzu języka).

Spakowanie w postaci instalatora natywnego powoduje jednak utratę niektórych korzyści związanych z formatem pliku AIR. Pojedynczy plik nie może być już rozpowszechniany na wszystkich komputerach stacjonarnych. Nie działa wbudowana funkcja aktualizacji (a także platforma narzędzia do aktualizowania).

Gdy użytkownik kliknie dwukrotnie plik rodzimego instalatora, instalator zainstaluje aplikację AIR. Jeśli na komputerze nie została jeszcze zainstalowana wymagana wersja środowiska Adobe AIR, instalator pobierze wersję z sieci i zainstaluje ją jako pierwszą. Jeśli nie jest dostępne połączenie sieciowe, z którego można by pobrać poprawną wersję Adobe AIR (w razie potrzeby), wówczas instalacja zakończy się niepowodzeniem. Ponadto instalacja nie powiedzie się, jeśli konkretny system operacyjny nie jest obsługiwany przez Adobe AIR 2.

Uwaga: Aby plik był plikiem wykonywalnym w zainstalowanej aplikacji, należy upewnić się, że jest on plikiem wykonywalnym w systemie plików przed utworzeniem pakietu aplikacji. (W systemach Mac i Linux można w razie potrzeby użyć polecenia `chmod` w celu ustawienia flagi pliku wykonywalnego.)

Tworzenie rodzimego instalatora z plików źródłowych aplikacji

W celu utworzenia rodzimego instalatora z plików źródłowych dla aplikacji należy użyć polecenia `-package` z następującą składnią (w jednym wierszu poleceń):

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

Ta składnia jest podobna do składni pakowania pliku AIR (bez rodzimego instalatora). Istnieją jednak pewne różnice:

- Do polecenia dodano opcję `-target native`. (Gdyby dodano `-target air`, wówczas narzędzie ADT wygenerowałoby plik AIR zamiast pliku instalatora rodzimego).
- Nazwę docelowego pliku DMG lub EXE określono jako `plik_instalatora`.
- Opcjonalnie w systemie Windows można dodać drugi zestaw opcji podpisu, oznaczonych jako `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` w listingu składni. W systemie Windows oprócz podpisania pliku AIR można również podpisać plik instalatora Windows. W tym przypadku należy użyć tego samego certyfikatu i tej samej składni opcji podpisu, która zostałaby użyta do podpisania pliku AIR (informacje zawiera rozdział „[Opcje podpisywania kodu ADT](#)” na stronie 190). W celu podpisania pliku AIR i pliku instalatora można użyć tego samego certyfikatu lub można określić inne certyfikaty. Gdy użytkownik pobierze podpisany plik instalatora Windows z sieci, system Windows zidentyfikuje źródło pliku na podstawie certyfikatu.

Szczegółowe informacje na temat opcji narzędzia ADT innych niż opcja `-target` można znaleźć w sekcji „[Narzędzie ADT](#)” na stronie 175.

Poniżej przedstawiono przykład tworzenia pliku DMG (plik rodzimego instalatora dla systemu Mac OS):

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

Poniżej przedstawiono przykład tworzenia pliku EXE (plik rodzimego instalatora dla systemu Windows):

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

Poniżej przedstawiono przykład tworzenia i podpisania pliku EXE:

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

Tworzenie instalatora natywnego dla aplikacji korzystającej z rozszerzeń natywnych

Instalator natywny można utworzyć z plików źródłowych aplikacji i pakietów rozszerzeń natywnych wymaganych przez aplikację. Należy użyć polecenia `-package` z następującą składnią (w pojedynczym wierszu polecenia):

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

Składnia jest taka sama jak w przypadku pakowania instalatora natywnego, lecz są dostępne dwie dodatkowe opcje. Opcja `-extdir katalog_rozszerze` pozwala określić katalog zawierający pliki ANE (rozszerzenia natywne) używane przez aplikację. Za pomocą opcjonalnej flagi `-migrate` i parametrów `MIGRATION_SIGNING_OPTIONS` można podpisać aktualizację aplikacji przy użyciu podpisu migracji w przypadku, gdy główny certyfikat podpisu kodu różni się od certyfikatu użytego dla poprzedniej wersji. Więcej informacji można znaleźć w rozdziale „[Podpisywanie zaktualizowanej wersji aplikacji AIR](#)” na stronie 211.

Szczegółowe informacje na temat opcji narzędzia ADT zawiera sekcja „[Narzędzie ADT](#)” na stronie 175.

W poniższym przykładzie jest tworzony plik DMG (plik instalatora natywnego dla systemu Mac OS) dla aplikacji korzystającej z rozszerzeń natywnych.

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

Tworzenie rodzimego instalatora z pliku AIR lub pliku AIRI

Za pomocą narzędzia ADT można wygenerować plik rodzimego instalatora z pliku AIR lub pliku AIRI. W celu utworzenia rodzimego instalatora z pliku AIR należy użyć polecenia `-package` narzędzia ADT z następującą składnią (w jednym wierszu poleceń):

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

Ta składnia przypomina składnię przeznaczoną do tworzenia rodzimego instalatora z plików źródłowych dla aplikacji AIR. Istnieją jednak pewne różnice:

- Jako źródło określono plik AIR, a nie plik deskryptora aplikacji ani inne pliki źródłowe dla aplikacji AIR.
- Nie należy określać opcji podpisu dla pliku AIR, ponieważ jest on już podpisany

W celu utworzenia rodzimego instalatora z pliku *AIRI* należy użyć polecenia `-package` narzędzia ADT z następującą składnią (w jednym wierszu poleceń):

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

Ta składnia przypomina składnię przeznaczoną do tworzenia rodzimego instalatora opartego na pliku AIR. Istnieją jednak pewne różnice:

- Jako źródło określono plik AIRI.
- Określono opcje podpisu dla docelowej aplikacji AIR.

Poniżej przedstawiono przykład tworzenia pliku DMG (plik rodzimego instalatora dla systemu Mac OS) na podstawie pliku AIR:

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

Poniżej przedstawiono przykład tworzenia pliku EXE (plik rodzimego instalatora dla systemu Windows) na podstawie pliku AIR:

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

Poniżej przedstawiono przykład tworzenia i podpisywania pliku EXE (na podstawie pliku AIR):

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

Poniżej przedstawiono przykład tworzenia pliku DMG (plik rodzimego instalatora dla systemu Mac OS) na podstawie pliku AIRI:

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

Poniżej przedstawiono przykład tworzenia pliku EXE (plik rodzimego instalatora dla systemu Windows) na podstawie pliku AIRI:

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

W poniższym przykładzie jest tworzony plik EXE (na podstawie pliku AIRI), który zostaje podpisany za pomocą zarówno podpisu AIR, jak i natywnego podpisu systemu Windows.

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

Tworzenie pakietu z dołączonym środowiskiem wykonawczym dla komputerów

Pakiet z dołączonym środowiskiem wykonawczym to pakiet zawierający kod aplikacji razem z dedykowaną wersją środowiska wykonawczego. Spakowana w ten sposób aplikacja używa dołączonego środowiska wykonawczego zamiast współużytkowanego środowiska wykonawczego zainstalowanego na komputerze użytkownika.

Utworzony pakiet jest autonomicznym folderem z plikami aplikacji w systemie Windows lub pakietem app w systemie Mac OS. Pakiet dla określonego docelowego systemu operacyjnego należy utworzyć podczas pracy w tym systemie. (Do uruchamiania wielu systemów operacyjnych na tym samym komputerze można użyć maszyny wirtualnej, takiej jak oprogramowanie firmy VMWare).

Aplikację można uruchomić z tego folderu lub pakietu bez instalowania.

Korzyści

- Utworzona aplikacja jest autonomiczna.
- Nie jest wymagany dostęp do Internetu na potrzeby instalacji.
- Aplikacja jest niezależna od aktualizacji środowiska wykonawczego.
- Przedsiębiorstwa mogą certyfikować określone kombinacje aplikacji i środowisk wykonawczych.
- Możliwa jest obsługa tradycyjnego modelu wdrażania oprogramowania.
- Nie jest konieczne rozpowszechnianie oddzielnego środowiska wykonawczego.
- Można korzystać z interfejsu API NativeProcess.
- Można korzystać z rozszerzeń natywnych.
- Można bez ograniczeń korzystać z funkcji `File.openWithDefaultApplication()`.
- Można uruchamiać aplikacje z dysku USB lub dysku optycznego bez instalowania.

Wady

- Krytyczne poprawki zabezpieczeń nie są automatycznie udostępniane użytkownikom po opublikowaniu przez firmę Adobe.
- Nie można korzystać z formatu pliku air.
- W razie potrzeby trzeba utworzyć własny instalator.
- Nie są obsługiwane platforma i interfejs API aktualizacji środowiska AIR.
- Nie jest obsługiwany interfejs API środowiska AIR umożliwiający instalowanie i uruchamianie aplikacji AIR w przeglądarce z poziomu strony internetowej.
- W systemie Windows rejestracja plików musi być obsługiwana przez instalatora.
- Aplikacja zajmuje więcej miejsca na dysku.

Tworzenie pakietu z dołączonym środowiskiem wykonawczym w systemie Windows

Aby utworzyć pakiet z dołączonym środowiskiem wykonawczym dla systemu Windows, należy spakować aplikację podczas pracy w tym systemie. Aplikację należy spakować przy użyciu typu docelowego *bundle*:

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

To polecenie powoduje utworzenie pakietu w katalogu o nazwie myApp. Katalog zawiera pliki aplikacji, jak również pliki środowiska wykonawczego. Można uruchomić program bezpośrednio z tego folderu. Aby jednak utworzyć pozycję menu dotyczącą programu bądź zarejestrować typy plików lub moduły obsługi schematów URI, należy utworzyć program instalacyjny ustawiający wymagane wpisy w rejestrze. Zestaw SDK środowiska AIR nie zawiera narzędzi do tworzenia takich instalatorów, ale jest dostępnych kilka rozwiązań innych firm, w tym zarówno komercyjne, jak i darmowe (open source) zestawy narzędzi instalatora.

Natywny plik wykonywalny można podpisać w systemie Windows, określając drugi zestaw opcji podpisywania umieszczony po opcji `-target bundle` w wierszu polecenia. Te opcje podpisywania identyfikują klucz prywatny i skojarzony certyfikat do użycia podczas stosowania natywnego podpisu dla systemu Windows. (Zazwyczaj można użyć certyfikatu do podpisywania kodu środowiska AIR). Podpisywany jest tylko główny plik wykonywalny. Wszelkie dodatkowe pliki wykonywalne spakowane razem z aplikacją nie są podpisywane w tym procesie.

Powiązania typów plików

Aby skojarzyć aplikację z publicznymi lub własnymi typami plików w systemie Windows, program instalacyjny musi ustawić odpowiednie wpisy w rejestrze. Te typy plików powinny być również wymienione w elemencie `fileTypes` w pliku deskryptora aplikacji.

Więcej informacji o typach plików systemu Windows zawiera dokument [Typy plików i skojarzenia plików w bibliotece MSDN](#)

Rejestrowanie modułów obsługi identyfikatorów URI

Aby aplikacja mogła obsługiwać uruchamianie adresów URL za pomocą określonego schematu URI, instalator musi ustawić wymagane wpisy w rejestrze.

Więcej informacji o rejestrowaniu aplikacji do obsługi schematu URI zawiera dokument [Rejestrowanie aplikacji dla protokołu URL w bibliotece MSDN](#)

Tworzenie pakietu z dołączonym środowiskiem wykonawczym w systemie Mac OS X

Aby utworzyć pakiet z dołączonym środowiskiem wykonawczym dla systemu Mac OS X, należy spakować aplikację podczas pracy w systemie operacyjnym Macintosh. Aplikację należy spakować przy użyciu typu docelowego `bundle`:

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

To polecenie powoduje utworzenie pakietu aplikacji o nazwie `myApp.app`. Pakiet zawiera pliki aplikacji, jak również pliki środowiska wykonawczego. W celu uruchomienia aplikacji można kliknąć dwukrotnie ikonę `myApp.app` i przeprowadzić instalację, przeciągając ikonę do odpowiedniej lokalizacji, na przykład do folderu aplikacji. Aby jednak zarejestrować typy plików lub moduły obsługi schematów URI, należy przeprowadzić edycję pliku z listą właściwości wewnątrz pakietu aplikacji.

Na potrzeby rozpowszechniania można utworzyć plik obrazu dysku (`dmg`). Zestaw SDK środowiska AIR nie zawiera narzędzi do tworzenia pliku `dmg` dla pakietu z dołączonym środowiskiem wykonawczym.

Powiązania typów plików

Aby skojarzyć aplikację z publicznymi lub własnymi typami plików w systemie Mac OS X, należy ustawić właściwość `CFBundleDocumentTypes`, edytując plik `info.plist` w pakiecie. Więcej informacji zawiera artykuł [Biblioteka programistów systemu Mac OS X: informacje o kluczach listy właściwości — CFBundleURLTypes](#).

Rejestrowanie modułów obsługi identyfikatorów URI

Aby aplikacja mogła obsługiwać uruchamianie adresów URL za pomocą określonego schematu URI, należy ustawić właściwość `CFBundleURLTypes`, edytując plik `info.plist` w pakiecie. Więcej informacji zawiera artykuł [Biblioteka programistów systemu Mac OS X: informacje o kluczach listy właściwości — CFBundleDocumentTypes](#).

Rozpowszechnianie pakietów AIR na komputery stacjonarne

Aplikacje AIR mogą być rozpowszechniane w postaci pakietów AIR, które zawierają kod aplikacji i wszystkie zasoby. Taki pakiet można rozpowszechnić, używając dowolnego standardowego sposobu — na przykład jako plik do pobrania, w wiadomości e-mail albo na nośniku fizycznym, takim jak płyta CD-ROM. Użytkownik może zainstalować aplikację, klikając dwukrotnie plik AIR. Korzystając z interfejsu API środowiska AIR w przeglądarce (internetowej biblioteki ActionScript), można umożliwić użytkownikom instalowanie aplikacji AIR (i w razie potrzeby środowiska Adobe® AIR®) przez kliknięcie pojedynczego łącza na stronie internetowej.

Aplikacje AIR mogą być również pakowane i rozpowszechniane w postaci instalatorów natywnych (czyli plików EXE w systemie Windows, plików DMG na komputerach Mac i plików DEB lub RPM w systemie Linux). Macierzyste pakiety instalacyjne mogą być rozpowszechniane i instalowane zgodnie z konwencjami obowiązującymi na poszczególnych platformach. W przypadku rozpowszechniania aplikacji w postaci pakietu natywnego nie można korzystać z pewnych zalet związanych z formatem pliku AIR. Pojedynczy plik nie może już być używany na większości platform. Nie można też korzystać z platformy aktualizacji środowiska AIR ani interfejsu API środowiska w przeglądarce.

Instalowanie i uruchamianie aplikacji AIR na komputerze stacjonarym

Plik AIR można po prostu wysłać do odbiorcy. Na przykład: można wysłać plik jako załącznik e-mail lub jako odsyłacz do strony internetowej.

Gdy użytkownik pobierze aplikację AIR, powinien wykonać poniższe czynności w celu zainstalowania aplikacji:

- 1 Kliknij dwukrotnie plik AIR.

Środowisko Adobe AIR musi być już zainstalowane na komputerze.

- 2 W oknie instalacji pozostaw ustawienia domyślne i kliknij przycisk kontynuowania.

W systemie Windows środowisko AIR automatycznie wykonuje następujące operacje:

- Instaluje aplikację w katalogu Program Files
- Tworzy skrót pulpitu dla aplikacji
- Tworzy skrót w menu Start
- Dodaje pozycję dotyczącą aplikacji do obszaru Dodaj/Usuń Programy w panelu sterowania

W systemie Mac OS domyślnie aplikacja jest dodawana do katalogu Applications.

Jeśli aplikacja jest już zainstalowana, instalator umożliwia wybór: można otworzyć istniejącą wersję aplikacji lub zaktualizować wersję do wersji z pobranego pliku AIR. Instalator identyfikuje aplikację, korzystając z jej identyfikatora oraz z identyfikatora wydawcy w pliku AIR.

- 3 Po zakończeniu instalowania kliknij przycisk Zakończ.

W celu zainstalowania zaktualizowanej wersji aplikacji w systemie Mac OS użytkownik powinien mieć odpowiednie uprawnienia dostępu do zainstalowania w katalogu aplikacji. W systemach Windows i Linux użytkownik musi mieć uprawnienia administratora.

Aplikacja może również zainstalować nową wersję za pośrednictwem kodu ActionScript lub kodu JavaScript. Więcej informacji zawiera sekcja „Aktualizowanie aplikacji AIR” na stronie 273.

Po zainstalowaniu aplikacji AIR użytkownik powinien kliknąć dwukrotnie ikonę aplikacji w celu jej uruchomienia — tak samo jak w przypadku każdej innej aplikacji.

- W systemie Windows kliknij dwukrotnie ikonę aplikacji (zainstalowana na pulpicie lub w folderze) albo wybierz aplikację z menu Start.
- W systemie Linux kliknij dwukrotnie ikonę aplikacji (zainstalowana wcześniej na pulpicie lub w folderze) albo wybierz aplikację z menu aplikacji.
- W systemie Mac OS kliknij dwukrotnie aplikację w folderze, w którym została zainstalowana. Domyślnym katalogiem instalowania jest katalog /Applications.

Uwaga: W systemie Linux można instalować tylko aplikacje AIR opracowane dla środowiska AIR 2.6 lub starszego.

Funkcja AIR *instalacji bezproblemowej* umożliwia zainstalowanie aplikacji AIR poprzez kliknięcie odsyłacza na stronie internetowej. Funkcje *wywołania przeglądarki* AIR umożliwiają użytkownikowi uruchomienie zainstalowanej aplikacji AIR poprzez kliknięcie odsyłacza na stronie internetowej. Te funkcje zostały opisane w poniższej sekcji.

Instalowanie i uruchamianie aplikacji AIR na komputery stacjonarne ze strony internetowej

Interfejs API środowiska AIR w przeglądarce umożliwia instalowanie i uruchamianie aplikacji AIR ze strony internetowej. Interfejs API środowiska AIR w przeglądarce jest oferowany w postaci udostępnianej przez firmę Adobe biblioteki SWF: *air.swf*. Zestaw SDK środowiska AIR zawiera przykładową aplikację „paska”, która używa tej biblioteki do instalowania, aktualizowania lub uruchamiania aplikacji AIR (oraz, w razie konieczności, środowiska wykonawczego). Można zmodyfikować udostępniany pasek przykładowy lub utworzyć własną aplikację internetową paska korzystając bezpośrednio z biblioteki *air.swf*.

Używając paska strony internetowej, można zainstalować dowolną aplikację AIR. Za pomocą paska internetowego można jednak uruchamiać wyłącznie aplikacje zawierające w plikach deskryptorów aplikacji element `<allowBrowserInvocation>true</allowBrowserInvocation>`.

Więcej tematów Pomocy

„[Interfejs API środowiska w przeglądarce — AIR.SWF](#)” na stronie 264

Wdrażanie na komputerach stacjonarnych w przedsiębiorstwach

Administratorzy IT mogą zainstalować (instalacja cicha) środowisko wykonawcze Adobe AIR oraz aplikacje AIR, korzystając ze standardowych narzędzi do wdrażania. Czynności, jakie mogą wykonać administratorzy IT:

- Cicha instalacja środowiska wykonawczego Adobe AIR przy użyciu narzędzi, takich jak Microsoft SMS, IBM Tivoli lub dowolnego narzędzia do wdrażania, które umożliwia wykonywanie instalacji cichej z wykorzystaniem bootstrappingu
- Cicha instalacja aplikacji AIR przy użyciu niektórych narzędzi używanych w celu wdrożenia środowiska wykonawczego

Więcej informacji zawiera dokumentacja [Podręcznik administratora środowiska Adobe AIR](#) (http://www.adobe.com/go/learn_air_admin_guide_pl).

Dzienniki instalacji na komputerach stacjonarnych

Dzienniki instalacji są zapisywane w środowisku wykonawczym AIR lub w instalowanej aplikacji AIR. Pliki dzienników ułatwiają określanie przyczyn problemów z instalowaniem lub aktualizowaniem.

Pliki dzienników są tworzone w następujących miejscach:

- Mac: standardowy dziennik systemu (`/private/var/log/system.log`)

Dziennik systemowy komputera Mac można wyświetlać, otwierając aplikację konsoli (znajdącą się zazwyczaj w folderze Narzędzia).

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

Uwaga: Te pliki dziennika nie były tworzone w wersjach AIR wcześniejszych niż AIR 2.

Rozdział 7: Programowanie aplikacji AIR dla urządzeń przenośnych

Aplikacje AIR dla urządzeń przenośnych są wdrażane jako aplikacje natywne. Korzystają one z formatu aplikacji urządzenia, a nie z formatu pliku AIR. W chwili obecnej środowisko AIR obsługuje pakiety APK systemu Android i pakiety IPA systemu iOS. Po utworzeniu ostatecznej wersji pakietu aplikacji można ją rozpowszechniać, korzystając z normalnego mechanizmu danej platformy. W przypadku systemu Android oznacza to zazwyczaj sklep Android Market, a w przypadku systemu iOS — sklep App Store firmy Apple.

Do tworzenia aplikacji AIR dla urządzeń przenośnych można używać [zestawu SDK środowiska AIR](#) i programu Flash Professional, Flash Builder lub innego narzędzia do programowania w języku ActionScript. W chwili obecnej nie są obsługiwane aplikacje AIR dla urządzeń przenośnych oparte na języku HTML.

***Uwaga:** Dla urządzeń BlackBerry PlayBook firmy Research In Motion (RIM) jest dostępny własny zestaw SDK do programowania w środowisku AIR. Więcej informacji na temat programowania aplikacji dla urządzeń PlayBook zawiera strona [RIM: Programowanie aplikacji dla systemu BlackBerry Tablet OS](#).*

***Uwaga:** W tym dokumencie opisano metody programowania aplikacji dla systemu iOS z zastosowaniem zestawu SDK środowiska AIR 2.6 lub nowszego. Aplikacje utworzone w środowisku AIR 2.6 lub nowszym można instalować na telefonach iPhone 3G i iPhone 4 oraz tabletach iPad z systemem iOS 4 lub nowszym. W celu opracowania aplikacji AIR dla wcześniejszych wersji systemu iOS należy użyć narzędzia AIR 2 Packager for iPhone, zgodnie z opisem w temacie [Tworzenie aplikacji na telefon iPhone](#).*

Więcej informacji o sprawdzonych procedurach związanych z prywatnością można uzyskać w [Poradniku dotyczącym prywatności w zestawie SDK środowiska Adobe AIR](#).

Kompletne wymagania systemowe dotyczące uruchamiania aplikacji AIR podano na stronie [wymagań systemowych środowiska Adobe AIR](#).

Konfigurowanie środowiska programistycznego

Platformy urządzeń przenośnych wymagają wykonania dodatkowych czynności konfiguracyjnych poza normalnym skonfigurowaniem środowiska programistycznego AIR, Flex czy Flash. (Więcej informacji na temat konfigurowania podstawowego środowiska programistycznego AIR zawiera sekcja „[Narzędzia platformy Adobe Flash przeznaczone do tworzenia w AIR](#)” na stronie 18).

Konfiguracja w systemie Android

Zazwyczaj nie jest wymagana żadna specjalna konfiguracja dla systemu Android w przypadku środowiska AIR 2.6 lub nowszego. Narzędzie ADB systemu Android jest zawarte w zestawie SDK środowiska AIR (w folderze lib/android/bin). Narzędzie ADB jest używane w zestawie SDK środowiska AIR do instalowania, odinstalowywania i uruchamiania pakietów aplikacji na urządzeniu. Za pomocą narzędzia ADB można również wyświetlać dzienniki systemu. Aby utworzyć i uruchomić emulator systemu Android, należy pobrać oddzielny zestaw SDK systemu Android.

Jeśli aplikacja dodaje elementy do elementu `<manifestAdditions>` w deskrytorze aplikacji, które nie są rozpoznawane jako prawidłowe przez bieżącą wersję środowiska AIR, to należy zainstalować nowszą wersję zestawu SDK systemu Android. W zmiennej środowiskowej `AIR_ANDROID_SDK_HOME` lub parametrze wiersza polecenia `-platformsdk` należy ustawić ścieżkę plików zestawu SDK. Narzędzie do tworzenia pakietów AIR (ADT) korzysta z tego zestawu SDK w celu sprawdzenia poprawności wpisów w elemencie `<manifestAdditions>`.

W środowisku AIR 2.5 należy pobrać oddzielną kopię zestawu SDK systemu Android od firmy Google. Można ustawić zmienną środowiska `AIR_ANDROID_SDK_HOME`, aby wskazywała folder zestawu SDK systemu Android. Jeśli ta zmienna środowiska nie zostanie ustawiona, należy określić ścieżkę do zestawu SDK systemu Android w argumencie `-platformsdk` w wierszu polecenia narzędzia ADT.

Więcej tematów Pomocy

„Zmienne środowiskowe narzędzia ADT” na stronie 199

„Zmienne środowiskowe ścieżek” na stronie 320

Konfiguracja w systemie iOS

Aby zainstalować i przetestować aplikację dla systemu iOS na urządzeniu oraz aby rozpowszechnić taką aplikację, należy zostać uczestnikiem programu Apple iOS Developer (usługi płatnej). Po zostaniu członkiem programu iOS Developer można uzyskiwać dostęp do witryny iOS Provisioning Portal, gdzie można uzyskać od firmy Apple wymienione poniżej elementy oraz pliki, które są wymagane do zainstalowania aplikacji na telefonie w celu jej przetestowania i późniejszego rozpowszechnienia. Tymi elementami i plikami są między innymi:

- certyfikaty programistów i rozpowszechniania,
- identyfikatory aplikacji,
- pliki informacyjne dotyczące programowania i rozpowszechniania.

Uwagi dotyczące projektowania aplikacji na urządzenia przenośne

Sposób działania i cechy fizyczne urządzeń przenośnych wymagają dokładnego tworzenia kodu oraz projektowania. Na przykład niezwykle istotne jest optymalizowanie kodu, aby był on wykonywany jak najszybciej. Optymalizacja kodu ma oczywiście swoje granice — inteligentnie opracowany projekt działający w ramach ograniczeń narzucanych przez dane urządzenie może również pomóc w uniknięciu zbytniego obciążenia systemu realizującego renderowanie.

Kod

O ile przyspieszenie działania kodu jest zawsze korzystne, o tyle niższa szybkość procesora na większości urządzeń przenośnych oznacza, że korzyści wynikające z poświęcenia czasu na napisanie optymalnego kodu są znaczące. Ponadto urządzenia przenośne działają prawie zawsze przy zasilaniu akumulatorowym. Osiągnięcie tego samego wyniku mniejszym nakładem pracy oznacza mniejsze zużycie baterii.

Projekt

Podczas projektowania sposobu działania aplikacji należy wziąć pod uwagę takie czynniki jak niewielki rozmiar ekranu, tryb współpracy z ekranem dotykowym, czy stale zmieniające się środowisko użytkownika urządzenia przenośnego.

Połączenie kodu i projektu

Jeśli w aplikacji są używane animacje, bardzo ważna jest optymalizacja renderowania. Jednak optymalizacja samego kodu często nie wystarcza. Należy zaprojektować wizualne aspekty aplikacji tak, aby mogły one być skutecznie renderowane przy użyciu kodu.

Ważne techniki optymalizacji zostały omówione w przewodniku [Optymalizowanie materiałów dla platformy Flash](#). Techniki omówione w tym przewodniku odnoszą się do wszystkich materiałów Flash i AIR, ale są kluczowe podczas opracowywania aplikacji działających dobrze na urządzeniach przenośnych.

- [Paul Trani: Wskazówki i porady na temat tworzenia aplikacji Flash dla urządzeń przenośnych](#)
- [roguish: Aplikacja do testowania GPU w środowisku AIR dla urządzeń przenośnych](#)
- [Jonathan Campos: Techniki optymalizacji aplikacji AIR dla systemu Android](#)
- [Charles Schulze: Programowanie gier dla środowiska AIR 2.6 — również dla systemu iOS](#)

Cykl życia aplikacji

Gdy aplikacja przestaje być aktywna (przekazuje punkt skupienia do innej aplikacji), środowisko AIR obniża szybkość odtwarzania do 4 klatek na sekundę i zatrzymuje renderowanie grafiki. Poniżej tej szybkości odtwarzania łatwo dochodzi do przerywania przesyłania strumieniowego za pośrednictwem połączeń sieciowych i gniazd. Jeśli aplikacja nie korzysta z takich połączeń, można jeszcze bardziej obniżyć szybkość odtwarzania.

W stosownych przypadkach należy zatrzymać odtwarzanie dźwięku oraz usunąć detektory czujników przyspieszeniomierza i lokalizacji geograficznej. Obiekt `NativeApplication` środowiska AIR wywołuje zdarzenia aktywacji i dezaktywacji. Za pomocą tych zdarzeń można kontrolować przechodzenie między stanem aktywnym i stanem działania w tle.

Większość systemów operacyjnych urządzeń przenośnych bez ostrzeżenia kończy działanie aplikacji działających w tle. Dzięki częstemu zapisywaniu stanu aplikacji aplikacja powinna być w stanie prawidłowo przywracać swój stan zarówno podczas powracania do stanu aktywnego po działaniu w tle, jak i w przypadku ponownego uruchamiania.

Gęstość informacji

Fizyczna wielkość ekranu urządzeń przenośnych jest dużo mniejsza niż w przypadku komputerów stacjonarnych, ale gęstość pikseli na ich ekranach (ilość pikseli na cal) jest wyższa. Ten sam rozmiar czcionki zaowocuje fizycznie mniejszymi literami na ekranie urządzenia przenośnego niż na komputerze stacjonarnym. W celu zapewnienia czytelności należy używać większej czcionki. Najmniejsza czcionka, która jest dobrze czytelna, ma zazwyczaj rozmiar 14 punktów.

Urządzenia przenośne są często używane w ruchu, a także przy nieodpowiednim oświetleniu. Należy więc zastanowić się, jak wiele informacji można wyświetlić w sposób czytelny na ekranie takiego urządzenia. Może być to mniej niż w przypadku monitora komputera stacjonarnego o tych samych wymiarach w pikselach.

Należy również wziąć pod uwagę to, że gdy użytkownik dotyka ekranu, palec i ręka zasłaniają część wyświetlacza. Gdy użytkownik ma korzystać z elementów interaktywnych dłużej niż przez czas krótkiego dotknięcia, należy umieszczać je po bokach i u dołu ekranu.

Wprowadzanie tekstu

Na wielu urządzeniach tekst jest wprowadzany za pomocą klawiatury wirtualnej. Klawiatury wirtualne zasłaniają część ekranu i często są kłopotliwe w użytkowaniu. Należy unikać polegania na zdarzeniach klawiatury (poza przyciskami programowymi).

Należy rozważyć zastosowanie opcji alternatywnych w stosunku do używania pól tekstu wejściowego. Na przykład do wprowadzenia wartości numerycznej nie jest potrzebne pole tekstowe. Ten sam efekt można osiągnąć za pomocą dwu przycisków, jednego do zwiększania, zaś drugiego do zmniejszania wartości.

Przyciski programowe

Urządzenia przenośne oferują różną liczbę przycisków programowych. Przyciski programowe to przyciski, dla których można zaprogramować różne funkcje. W aplikacji należy przestrzegać konwencji platformy dla tych przycisków.

Zmiany orientacji ekranu

Zawartość wyświetlaną na urządzeniach przenośnych można oglądać w orientacji pionowej lub poziomej. Należy więc rozważyć sposób obsługi zmian orientacji ekranu przez tworzoną aplikację. Więcej informacji zawiera artykuł [Orientacja stołu montażowego](#).

Przyciemnianie ekranu

Środowisko AIR nie zapobiega automatycznie przyciemnianiu ekranu podczas odtwarzania wideo. Do kontrolowania przechodzenia urządzenia do trybu oszczędzania energii można używać właściwości `systemIdleMode` obiektu `NativeApplication` środowiska AIR. (Na niektórych platformach prawidłowe działanie tej funkcji wymaga zażądania odpowiednich uprawnień).

Połączenia przychodzące

Środowisko wykonawcze AIR automatycznie wycisza dźwięki, gdy użytkownik nawiązuje lub odbiera połączenie telefoniczne. Jeśli w systemie Android aplikacja odtwarza dźwięk podczas działania w tle, należy ustawić w deskrypcji aplikacji uprawnienie `READ_PHONE_STATE` systemu Android. W przeciwnym razie system Android uniemożliwi środowisku wykonawczemu wykrywanie połączeń telefonicznych i automatyczne wyciszanie dźwięków. Zobacz „[Uprawnienia w systemie Android](#)” na stronie 82.

Cele dotknięć i kliknięć

Projektując przyciski oraz inne elementy interfejsu użytkownika przeznaczone do stukania przez użytkownika, należy przemyśleć ich rozmiar. Elementy te powinny być odpowiednio duże, tak aby ich aktywacja możliwa była w wygodny sposób za pomocą palca na ekranie dotykowym. Należy również upewnić się, że między miejscami docelowymi jest odpowiednia ilość miejsca. W przypadku typowego ekranu telefonu o dużej rozdzielczości bok obszaru docelowego dotknięcia lub kliknięcia powinien mieć długość od 44 do 57 pikseli.

Rozmiar instalacji pakietu aplikacji

Urządzenia przenośne mają zazwyczaj dużo mniej przestrzeni dyskowej przeznaczonej do instalowania aplikacji i przechowywania danych niż komputery stacjonarne. Rozmiar pakietu należy zminimalizować, usuwając nieużywane zasoby i biblioteki.

W systemie Android pakiet aplikacji nie jest rozpakowywany do oddzielnych plików podczas instalowania aplikacji. Zamiast tego w momencie uzyskiwania dostępu do zasobów są one rozpakowywane do magazynu tymczasowego. Aby zmniejszyć pozostałości wynikające z takiego przechowywania rozpakowywanych zasobów, po całkowitym wczytaniu zasobów należy zamknąć strumień adresu URL i pliku.

Dostęp do systemu plików

Różne systemy operacyjne dla urządzeń przenośnych nakładają na system plików różne ograniczenia, które zazwyczaj są inne niż ograniczenia nakładane przez komputerowe systemy operacyjne. Odpowiednie miejsce do zapisywania plików i danych może więc różnić się w zależności od platformy.

Jedną z konsekwencji różnic między systemami plików jest to, że skróty do typowych katalogów podawanych przez klasę File środowiska AIR nie zawsze są dostępne. Poniższa tabela pokazuje, których skrótów można używać w systemach Android i iOS:

	Android	iOS
File.applicationDirectory	Tylko do odczytu za pomocą adresu URL (nie ścieżki natywnej)	Tylko do odczytu
File.applicationStorageDirectory	Dostępny	Dostępny
File.cacheDirectory	Dostępny	Dostępny
File.desktopDirectory	Katalog główny karty SD	Niedostępny
File.documentsDirectory	Katalog główny karty SD	Dostępny
File.userDirectory	Katalog główny karty SD	Niedostępny
File.createTempDirectory()	Dostępny	Dostępny
File.createTempFile()	Dostępny	Dostępny

Firma Apple opracowała szczegółowe reguły dotyczące umieszczania plików w lokalizacjach magazynu przez aplikacje dla systemu iOS w zależności od sytuacji. Zalecane jest na przykład przechowywanie w katalogu objętym zdalną kopią zapasową tylko tych plików, które zawierają dane wprowadzone przez użytkownika lub z innego powodu są niemożliwe do wygenerowania ani ponownego pobrania. Informacje na temat uzyskiwania zgodności ze wskazówkami firmy Apple dotyczącymi kopii zapasowych i buforowania można znaleźć w rozdziale Sterowanie kopiami zapasowymi i buforowaniem plików.

Elementy interfejsu użytkownika

Firma Adobe opracowała wersję środowiska Flex zoptymalizowaną dla urządzeń przenośnych. Więcej informacji znajduje się w dokumencie [Tworzenie aplikacji przenośnych przy użyciu programów Flex i Flash Builder](#).

Są również dostępne społecznościowe projekty obejmujące składniki przeznaczone dla aplikacji na urządzenia przenośne. Przykłady:

- Josh Tynjala — [Elementy interfejsu użytkownika Feathers dla platformy Starling](#)
- [Skinnable version of Minimal Comps](#) (wersja składników minimalnych z możliwością umieszczania w motywie) Derricka Grigga
- [As3flobile components](#) (składniki as3flobile) Todda Andersona

Renderowanie grafiki na stole montażowym z przyspieszaniem 3D

Począwszy od wersji AIR 3.2, środowisko AIR dla urządzeń przenośnych obsługuje renderowanie grafiki na stole montażowym z przyspieszaniem 3D. Interfejsy API [Stage3D](#) języka ActionScript to zestaw interfejsów API niskiego poziomu, które korzystają z przyspieszania GPU i oferują zaawansowane funkcje 2D i 3D. Za pomocą tych interfejsów API niskiego poziomu programiści mogą elastycznie korzystać z przyspieszania sprzętowego GPU w celu znacznego zwiększenia wydajności. Można również korzystać z mechanizmów gier, które obsługują interfejsy API Stage3D języka ActionScript.

Więcej informacji można znaleźć na stronie [Mechanizmy gier, grafika 3D i stół montażowy 3D](#).

Wyglądanie wideo

W celu zwiększenia wydajności w środowisku AIR jest wyłączona funkcja wyglądzania wideo.

Funkcje natywne

AIR 3.0 i nowsze wersje

Wiele platform dla urządzeń przenośnych oferuje funkcje, które nie są jeszcze dostępne w standardowym interfejsie API środowiska AIR. Od wersji AIR 3 można rozszerzać środowisko AIR przy użyciu własnych, natywnych bibliotek kodu. Natywne biblioteki rozszerzeń mogą używać funkcji dostępnych z poziomu systemu operacyjnego, a nawet specyficznych dla urządzenia. Rozszerzenia natywne można pisać w języku C dla systemu iOS oraz w językach Java i C dla systemu Android. Informacje na temat programowania rozszerzeń natywnych zawiera rozdział Wprowadzanie rozszerzeń natywnych dla środowiska Adobe AIR.

Obieg pracy tworzenia aplikacji AIR dla urządzeń przenośnych

Obieg pracy tworzenia aplikacji AIR dla urządzeń przenośnych (lub innych) jest ogólnie bardzo podobny do obiegu pracy tworzenia aplikacji na komputery stacjonarne. Główne różnice w obiegu pracy występują podczas pakowania, debugowania i instalowania aplikacji. Na przykład aplikacje AIR dla systemu Android korzystają z formatu pakietów Android APK, a nie z formatu pakietów AIR. Z tego powodu używają one również standardowych mechanizmów instalacji i aktualizacji systemu Android.

Środowisko AIR for Android

Poniżej przedstawiono typowe kroki występujące podczas opracowywania aplikacji AIR dla systemu Android:

- Napisz kod ActionScript lub MXML.
- Utwórz plik deskryptora aplikacji AIR (przy użyciu przestrzeni nazw w wersji 2.5 lub nowszej).
- Skompiluj aplikację.
- Spakuj aplikację do postaci pakietu Android (apk).
- Zainstaluj środowisko wykonawcze na urządzeniu lub w symulatorze systemu Android. (Dotyczy używania zewnętrznego środowiska wykonawczego. Od wersji AIR 3.7 środowisko wykonawcze jest zintegrowane z aplikacją).
- Zainstaluj aplikację na urządzeniu (lub w emulatorze systemu Android).
- Uruchom aplikację na urządzeniu.

Do przeprowadzenia tych czynności można użyć programu Adobe Flash Builder lub Adobe Flash Professional CS5 albo narzędzi wiersza poleceń.

Po zakończeniu opracowywania aplikacji AIR i spakowaniu jej do postaci pliku APK można przesłać ją do sklepu Android Market lub rozpowszechnić w inny sposób.

Środowisko AIR for iOS

Poniżej przedstawiono typowe kroki występujące podczas opracowywania aplikacji AIR dla systemu iOS:

- Zainstaluj program iTunes.
- Utwórz wymagane identyfikatory i pliki programisty na stronie iOS Provisioning Portal firmy Apple. Te elementy to między innymi:
 - Certyfikat programisty
 - Identyfikator aplikacji
 - Profil informacyjny

Podczas tworzenia profilu informacyjnego należy podać identyfikatory wszystkich urządzeń testowych, na których ma zostać zainstalowana aplikacja.

- Przekonwertuj certyfikat programowania i klucz prywatny na plik magazynu kluczy P12.
- Napisz kod ActionScript lub MXML aplikacji.
- Skompiluj aplikację za pomocą kompilatora ActionScript lub MXML.
- Utwórz obraz ikony i obraz ekranu początkowego aplikacji.
- Utwórz deskryptor aplikacji (przy użyciu przestrzeni nazw w wersji 2.6 lub nowszej).
- Spakuj plik IPA za pomocą narzędzia ADT.
- Umieść profil informacyjny na urządzeniu testowym za pomocą programu iTunes.
- Zainstaluj i przetestuj aplikację na urządzeniu z systemem iOS. Plik IPA można zainstalować przy użyciu programu iTunes lub narzędzia ADT (środowisko AIR 3.4 lub nowsza wersja).

Po zakończeniu opracowywania aplikacji AIR można ją ponownie spakować przy użyciu certyfikatu rozpowszechniania i profilu informacyjnego. Będzie ona wówczas gotowa do przesłania do sklepu App Store firmy Apple.

Konfigurowanie właściwości aplikacji dla urządzeń przenośnych

Podobnie jak w przypadku innych aplikacji AIR, podstawowe właściwości aplikacji należy ustawić w pliku deskryptora aplikacji. Aplikacje dla urządzeń przenośnych ignorują niektóre właściwości specyficzne dla komputerów stacjonarnych, takie jak rozmiar okna czy przezroczystość. Aplikacje dla urządzeń przenośnych mogą również używać własnych właściwości dla konkretnej platformy. Na przykład w aplikacjach dla systemu Android można korzystać z elementu `android`, a w aplikacjach dla systemu iOS z elementu `iPhone`.

Ustawienia wspólne

Kilka ustawień deskryptora aplikacji jest ważnych dla wszystkich aplikacji dla urządzeń przenośnych.

Wymagana wersja środowiska wykonawczego AIR

Wersję środowiska wykonawczego AIR, która jest wymagana przez aplikację, należy określić przy użyciu przestrzeni nazw pliku deskryptora aplikacji.

Przeźren nazw (przypisana w elemencie `application`) w dużej mierze określa funkcje, których aplikacja może używać. Jeśli na przykład aplikacja korzysta z przestrzeni nazw środowiska AIR 2.7 i użytkownik ma zainstalowaną nowszą wersję, to aplikacja rozpozna zachowania takie jak w środowisku AIR 2.7 (mimo zmiany wersji środowiska na nowszą). Dopiero zmiana przestrzeni nazw i opublikowanie aktualizacji umożliwiają aplikacji uzyskanie dostępu do nowych zachowań i funkcji. Poprawki zabezpieczeń stanowią ważny wyjątek od tej reguły.

Jeśli na urządzeniu korzystającym ze środowiska wykonawczego niezależnie od aplikacji, takim jak urządzenie z systemem Android i środowiskiem AIR 3.6, użytkownik nie ma wymaganej wersji środowiska AIR, jest wyświetlany monit o jej zainstalowanie lub uaktualnienie. Na urządzeniach, w przypadku których środowisko wykonawcze jest osadzone, takich jak telefon iPhone, ta sytuacja nie występuje (ponieważ wymagana wersja jest pakowana z aplikacją).

Uwaga: (AIR 3.7 i nowsze wersje) Domyślnie narzędzie ADT pakuje środowisko wykonawcze wraz z aplikacjami dla systemu Android.

Przeźren nazw należy określić przy użyciu atrybutu `xmlns` głównego elementu `application`. W przypadku aplikacji dla urządzeń przenośnych należy używać następujących przestrzeni nazw (w zależności od tego, na którą platformę dla urządzeń przenośnych jest przeznaczona aplikacja):

```
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">
```

Uwaga: Obsługa urządzeń z systemem iOS 3 jest zapewniana przez zestaw SDK Packager for iPhone, oparty na zestawie SDK środowiska AIR 2.0. Informacje na temat tworzenia aplikacji AIR dla systemu iOS 3 zawiera artykuł [Tworzenie aplikacji na telefon iPhone](#). Zestawy SDK środowiska AIR 2.6 oraz nowszych wersji obsługują system iOS 4 lub nowszy na telefonie iPhone 3G, telefonie iPhone 4 i tablecie iPad.

Więcej tematów Pomocy

„`application`” na stronie 223

Tożsamość aplikacji

Niektóre ustawienia powinny być niepowtarzalne dla każdej publikowanej aplikacji. Takie ustawienia obejmują identyfikator, nazwę aplikacji i nazwę pliku.

Identyfikatory aplikacji w systemie Android

W systemie Android identyfikator jest konwertowany na nazwę pakietu Android przez dodanie przedrostka „air.” do identyfikatora AIR. Jeśli więc identyfikator AIR ma wartość `com.example.MyApp`, wówczas nazwą pakietu Android jest `air.com.example.MyApp`.

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

Ponadto jeśli identyfikator nie jest poprawną nazwą pakietu w systemie operacyjnym Android, jest on konwertowany na poprawną nazwę. Znaki dywizu są zastępowane podkreśleniami, a cyfry wiodące w każdym składniku identyfikatora są poprzedzane wielką literą „A”. Na przykład identyfikator `3-kozy.1-owca` zostanie przekształcony w nazwę pakietu `air.A3_kozy.A1_owca`.

Uwaga: Przedrostek dodawany do identyfikatora aplikacji może służyć do identyfikowania aplikacji AIR w sklepie Android Market. Jeśli aplikacja nie ma być rozpoznawana jako aplikacja AIR z powodu przedrostka, należy rozpakować plik APK, zmienić identyfikator aplikacji i ponownie spakować go zgodnie z opisem w artykule [Rezygnacja z funkcji analizy aplikacji AIR dla systemu Android](#).

Identyfikatory aplikacji w systemie iOS

Należy ustawić identyfikator aplikacji AIR zgodny z identyfikatorem aplikacji utworzonym w witrynie iOS Provisioning Portal firmy Apple.

Identyfikatory aplikacji systemu iOS składają się z identyfikatora wartości początkowej pakietu, po którym następuje identyfikator pakietu. Identyfikator wartości początkowej pakietu to ciąg znaków, taki jak 5RM86Z4DJM, przypisywany przez firmę Apple do identyfikatora aplikacji. Identyfikator pakietu zawiera wybraną przez użytkownika nazwę w odwrotnej notacji domen. Identyfikator pakietu może kończyć się znakiem gwiazdki (*) oznaczającym wieloznaczny identyfikator aplikacji. Jeśli identyfikator pakietu kończy się znakiem wieloznacznym, ten znak można zastąpić dowolnym poprawnym ciągiem.

Na przykład:

- Jeśli identyfikator aplikacji Apple to 5RM86Z4DJM.com.example.witamWszystkich, w deskrytorze aplikacji należy zastosować ciąg com.example.witamWszystkich.
- Jeśli identyfikator aplikacji Apple to 96LPVWEASL.com.example.* (wieloznaczny identyfikator aplikacji), wówczas można użyć identyfikatora com.example.witamWszystkich, com.example.innaAplikacja lub innego identyfikatora zaczynającego się od ciągu com.example.
- Jeśli identyfikator aplikacji Apple składa się wyłącznie z identyfikatora wartości początkowej pakietu i znaku wieloznacznego, na przykład 38JE93KJL.*, wówczas w środowisku AIR można użyć dowolnego identyfikatora aplikacji.

Podczas określania identyfikatora aplikacji nie należy uwzględniać fragmentu identyfikatora aplikacji określającego identyfikator wartości początkowej pakietu.

Więcej tematów Pomocy

[„id”](#) na stronie 239

[„filename”](#) na stronie 234

[„name”](#) na stronie 247

Wersja aplikacji

W środowisku AIR 2.5 lub nowszą wersją aplikacji należy określić w elemencie `versionNumber`. Nie można już stosować elementu `version`. Określając wartość elementu `versionNumber`, należy użyć sekwencji składającej się z maksymalnie trzech liczb rozdzielonych kropkami, na przykład „0.1.2”. Każda z tych liczb w numerze wersji może zawierać do trzech cyfr. Innymi słowy, największym dozwolonym numerem wersji jest „999.999.999”. Numer wersji nie musi zawierać wszystkich trzech liczb. Zarówno „1”, jak i „1.0” stanowią prawidłowe numery wersji.

Przy użyciu elementu `versionLabel` można określić etykietę wersji. Po dodaniu etykiety wersji jest ona wyświetlana zamiast numeru wersji w takich miejscach jak ekran informacyjny aplikacji systemu Android. Aplikacje rozpowszechniane za pomocą sklepu Android Market muszą mieć określoną etykietę wersji. Jeśli w deskrytorze aplikacji AIR nie zostanie określona wartość `versionLabel`, wówczas do pola etykiety wersji systemu Android zostanie przypisana wartość `versionNumber`.

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

W systemie Android element `versionNumber` środowiska AIR jest konwertowany na liczbę całkowitą systemu Android `versionCode` przy użyciu wzoru $a*1000000 + b*1000 + c$, gdzie a, b i c są elementami numeru wersji środowiska AIR: a.b.c.

Więcej tematów Pomocy

„[version](#)” na stronie 255

„[versionLabel](#)” na stronie 256

„[versionNumber](#)” na stronie 256

Główny plik SWF aplikacji

Należy określić główny plik SWF aplikacji w elemencie potomnym `content` elementu `initialWindow`. Jeśli aplikacja jest przeznaczona na urządzenia o profilu urządzenia przenośnego, należy używać pliku SWF. (Aplikacje oparte na standardzie HTML nie są obsługiwane).

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

Plik musi zostać umieszczony w pakiecie aplikacji AIR (przy użyciu narzędzia ADT lub środowiska programistycznego). Samo umieszczenie odniesienia do nazwy w deskrytorze aplikacji nie powoduje automatycznego zawarcia pliku w pakiecie.

Właściwości ekranu głównego

Kilka elementów potomnych elementu `initialWindow` steruje początkowym wyglądem i działaniem głównego ekranu aplikacji.

- **aspectRatio** — określa, czy aplikacja powinna być na początku wyświetlana w formacie *portrait* (pionowym, z wysokością większą niż szerokość), *landscape* (poziomym, z wysokością mniejszą niż szerokość, czy *any* (dowolnym, z orientacją określaną automatycznie przez stół montażowy).

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

- Element **autoOrients** określa, czy orientacja stołu montażowego powinna automatycznie zmieniać się, gdy użytkownik obraca urządzenie lub wykonuje inne gesty związane z orientacją, takie jak otwieranie lub zamykanie wysuwanej klawiatury. Jeśli ten element ma wartość *false*, która jest wartością domyślną, wówczas stół montażowy nie zmienia orientacji razem z urządzeniem.

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

- **depthAndStencil** — Określa, czy ma być używany bufor głębi czy bufor szablonu. Te bufory są zazwyczaj używane podczas pracy z zawartością 3D.

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

- Element **fullScreen** określa, czy aplikacja powinna zajmować cały wyświetlacz, czy powinna współdzielić wyświetlacz z normalną karnacją systemu operacyjnego, na przykład z systemowym paskiem stanu.

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

- Element **renderMode** określa, czy środowisko wykonawcze powinno renderować aplikację za pomocą GPU czy procesora (CPU, central processing unit). Renderowanie za pomocą GPU zazwyczaj zwiększa wydajność, ale niektóre funkcje, takie jak pewne tryby mieszania i filtry `PixelBender`, są niedostępne w trybie GPU. Ponadto różne urządzenia i różne sterowniki urządzeń mają inne możliwości oraz ograniczenia dotyczące GPU. Aplikację należy zawsze testować na możliwie największej liczbie urządzeń, szczególnie w przypadku używania trybu GPU.

Można ustawić tryb renderowania *gpu*, *cpu*, *direct* lub *auto*. Wartością domyślną jest *auto*, co obecnie oznacza wybranie trybu *cpu*.

Uwaga: Aby można było stosować przyspieszanie GPU zawartości Flash w środowisku AIR dla platform przenośnych, firma Adobe zaleca, aby zamiast opcji `renderMode="direct"` (obiektu `Stage3D`) użyć opcji `renderMode="gpu"`. Firma Adobe oficjalnie oferuje pomoc techniczną dla następujących architektur opartych na obiekcie `Stage3D` i zaleca ich stosowanie: *Starling (2D)* i *Away3D (3D)*. Więcej informacji o architekturach `Stage3D` oraz *Starling/Away3D* znajduje się w dokumencie <http://gaming.adobe.com/getstarted/>.

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

Uwaga: W przypadku aplikacji działających w tle nie można używać parametru `renderMode` z wartością „direct”.

Ograniczenia trybu GPU:

- Architektura Flex nie obsługuje trybu renderowania GPU.
- Filtry nie są obsługiwane.
- Mieszanie `PixelBender` i wypełnianie nie są obsługiwane.
- Nie są obsługiwane tryby mieszania: `layer`, `alpha`, `erase`, `overlay`, `hardlight`, `lighten`, `darken`.
- Nie jest zalecane korzystanie z trybu renderowania GPU w aplikacji odtwarzającej wideo.
- W trybie renderowania GPU pola tekstowe nie są prawidłowo przenoszone w widoczne miejsca po otwarciu klawiatury wirtualnej. W celu zapewnienia widoczności pola tekstowego podczas wpisywania tekstu przez użytkownika należy przesunąć pole tekstowe w obręb widocznego obszaru za pomocą właściwości `softKeyboardRect` stołu montażowego oraz zdarzeń klawiatury programowej.
- Jeśli GPU nie może renderować danego obiektu ekranowego, w ogóle nie zostanie on wyświetlony. Jeśli na przykład względem obiektu ekranowego zostanie zastosowany filtr, obiekt nie zostanie wyświetlony.

Uwaga: Implementacja funkcji GPU dla systemu *iOS* w środowisku AIR 2.6+ znacznie różni się od implementacji stosowanej we wcześniejszej wersji środowiska AIR (2.0). Są stosowane inne założenia optymalizacji.

Więcej tematów Pomocy

„[aspectRatio](#)” na stronie 226

„[autoOrients](#)” na stronie 226

„[depthAndStencil](#)” na stronie 230

„[fullScreen](#)” na stronie 238

„[renderMode](#)” na stronie 250

Obsługiwane profile

Można dodać element `supportedProfiles` w celu określenia, które profile urządzeń obsługuje dana aplikacja. Dla urządzeń przenośnych należy używać profilu `mobileDevice`. W przypadku uruchamiania aplikacji za pomocą narzędzia Adobe Debug Launcher (ADL) ten program używa jako profilu aktywnego pierwszego profilu na liście. Podczas uruchamiania narzędzia ADL można użyć flagi `-profile` w celu wybrania konkretnego profilu z listy obsługiwanych. Jeśli aplikacja działa we wszystkich profilach, można całkowicie pominąć element `supportedProfiles`. Narzędzie ADL użyje wówczas profilu komputera stacjonarnego jako domyślnego profilu.

Aby określić, że aplikacja obsługuje profile dla urządzeń przenośnych i komputerów oraz że normalnie ma być testowana w profilu dla urządzeń przenośnych, należy dodać następujący element:

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

Więcej tematów Pomocy

„[supportedProfiles](#)” na stronie 253

„[Profile urządzeń](#)” na stronie 259

„[AIR Debug Launcher \(ADL\)](#)” na stronie 169

Wymagane rozszerzenia natywne

Aplikacje obsługujące profil `mobileDevice` mogą korzystać z rozszerzeń natywnych.

Wszystkie rozszerzenia natywne używane przez aplikację AIR należy zadeklarować w deskrytorze aplikacji. W poniższym przykładzie przedstawiono składnię umożliwiającą określenie dwóch wymaganych rozszerzeń natywnych:

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

Element `extensionID` ma wartość taką samą jak element `id` w pliku deskryptora rozszerzenia. Plik deskryptora rozszerzenia jest plikiem XML o nazwie `extension.xml`. Jest on spakowany w pliku ANE otrzymanym od programisty rozszerzenia natywnego.

Działanie klawiatury wirtualnej

Aby wyłączyć działanie automatycznego panoramowania i zmieniania rozmiaru — funkcji stosowanych przez środowisko wykonawcze w celu zapewnienia widoczności aktywnego pola wpisywania danych po uniesieniu klawiatury wirtualnej — dla elementu `softKeyboardBehavior` należy ustawić wartość `none`. Jeśli zachowanie automatyczne zostanie wyłączone, wówczas to aplikacja będzie odpowiedzialna za gwarantowanie widoczności obszaru wpisywania tekstu lub innej ważnej zawartości po uniesieniu klawiatury. Do wykrywania momentu otwierania klawiatury i zakrywanego przez nią obszaru można używać właściwości `softKeyboardRect` stołu montażowego w połączeniu z obiektem `SoftKeyboardEvent`.

Aby włączyć automatyczne zachowanie, dla tego elementu należy ustawić wartość `pan`.

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

Wartość `pan` jest wartością domyślną, więc pominięcie elementu `softKeyboardBehavior` również powoduje włączenie automatycznego zachowania klawiatury.

Uwaga: Jeśli jest również stosowane renderowanie GPU, nie jest obsługiwane zachowanie panoramowania.

Więcej tematów Pomocy

„[softKeyboardBehavior](#)” na stronie 252

[Stage.softKeyboardRect](#)

[SoftKeyboardEvent](#)

Ustawienia w systemie Android

Na platformie Android w celu dodawania informacji do manifestu aplikacji systemu Android, czyli pliku właściwości aplikacji używanego przez system operacyjny Android, można używać elementu `android` deskryptora aplikacji. Podczas tworzenia pakietu APK narzędzie ADT automatycznie tworzy plik `Manifest.xml` systemu Android. Środowisko AIR ustawia kilka właściwości wartości wymaganych do działania pewnych funkcji. Wszelkie inne właściwości ustawione w sekcji systemu Android deskryptora aplikacji AIR są dodawane do odpowiedniej sekcji pliku `Manifest.xml`.

Uwaga: W przypadku większości aplikacji AIR uprawnienia systemu Android wymagane przez aplikację należy ustawiać w elemencie `android`. Zwykle nie trzeba ustawiać żadnych innych właściwości.

Można ustawiać wyłącznie atrybuty przyjmujące wartości ciągów, liczb całkowitych lub wartości logicznych. Nie jest obsługiwane ustawianie odniesień do zasobów w pakiecie aplikacji.

Uwaga: Środowisko wykonawcze wymaga zestawu SDK nie starszej niż 14. Aby utworzyć aplikację wyłącznie dla nowszych wersji, należy zawrzeć w manifestcie wyrażenie `<uses-sdk android:minSdkVersion=""></uses-sdk>` z uwzględnieniem właściwej wersji.

Zarezerwowane ustawienia manifestu systemu Android

W celu zapewnienia prawidłowego działania aplikacji i środowiska wykonawczego kilka wpisów manifestu w tworzonym dokumencie manifestu systemu Android jest ustawianych przez środowisko AIR. Nie można definiować następujących ustawień:

Element manifest

Nie można ustawiać następujących atrybutów elementu manifest:

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

Element activity

Nie można ustawiać następujących atrybutów dla głównego elementu activity:

- `android:label`
- `android:icon`

Element application

Nie można ustawiać następujących atrybutów elementu application:

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

Uprawnienia w systemie Android

Model bezpieczeństwa w systemie Android wymaga, aby każda aplikacja żądała uprawnień w celu korzystania z funkcji mających następstwa w zakresie bezpieczeństwa lub prywatności. Te uprawnienia muszą zostać określone w momencie pakowania aplikacji i nie można ich zmieniać podczas jej działania. Gdy użytkownik instaluje aplikację, system operacyjny Android informuje, jakich uprawnień żąda ta aplikacja. Jeśli nie nastąpi żądanie uprawnienia wymaganego dla funkcji, system operacyjny Android może zgłosić wyjątek, gdy aplikacja uzyska dostęp do funkcji, ale nie musi to nastąpić. Wyjątki są przekazywane do aplikacji przez środowisko wykonawcze. W przypadku niezgłoszonego niepowodzenia do dziennika systemu Android jest dodawany komunikat o niepowodzeniu dotyczącym uprawnienia.

W środowisku AIR uprawnienia systemu Android są określane w elemencie `android` deskryptora aplikacji. Do dodawania uprawnień służy następujący format (gdzie fragment `PERMISSION_NAME` jest nazwą uprawnienia w systemie Android):

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

Instrukcje `uses-permissions` w elemencie `manifest` są dodawane bezpośrednio do dokumentu manifestu systemu Android.

Do korzystania z różnych funkcji środowiska AIR są wymagane następujące uprawnienia:

ACCESS_COARSE_LOCATION Umożliwia aplikacji uzyskiwanie za pośrednictwem klasy `Geolocation` dostępu do danych o lokalizacji określanych przy użyciu sieci Wi-Fi oraz sieci komórkowych.

ACCESS_FINE_LOCATION Umożliwia aplikacji uzyskiwanie dostępu do danych systemu GPS za pośrednictwem klasy `Geolocation`.

ACCESS_NETWORK_STATE i **ACCESS_WIFI_STATE** Umożliwiają aplikacji uzyskiwanie dostępu do informacji o sieci za pośrednictwem klasy `NetworkInfo`.

CAMERA Umożliwia aplikacji uzyskiwanie dostępu do kamery.

***Uwaga:** W przypadku żądania uprawnienia do korzystania z funkcji kamery system Android zakłada, że aplikacja również wymaga kamery. Jeśli kamera jest opcjonalną funkcją aplikacji, wówczas należy dodać do manifestu element `uses-feature` dotyczący kamery, ustawiając wartość `false` dla atrybutu `required`. Zobacz „[Filtrowanie zgodności w systemie Android](#)” na stronie 84.*

INTERNET Umożliwia aplikacji realizowania żądań dotyczących sieci. Pozwala także na zdalne debugowanie.

READ_PHONE_STATE Umożliwia środowisku AIR wyciszanie dźwięków podczas połączeń telefonicznych. To uprawnienie należy ustawić, jeśli podczas działania w tle aplikacja odtwarza dźwięk.

RECORD_AUDIO Umożliwia aplikacji uzyskiwanie dostępu do mikrofonu.

WAKE_LOCK i **DISABLE_KEYGUARD** Umożliwiają aplikacji blokowanie przejścia urządzenia w tryb uśpienia przy użyciu ustawień klasy `SystemIdleMode`.

WRITE_EXTERNAL_STORAGE Umożliwia aplikacji zapisywanie danych na zewnętrznej karcie pamięci podłączonej do urządzenia.

Na przykład w celu ustawienia uprawnień dla aplikacji wymagającej wszystkich uprawnień można dodać następujący deskryptor aplikacji:

```
<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>
```

Więcej tematów Pomocy

[Zabezpieczenia i uprawnienia w systemie Android](#)

[Klasa Manifest.permission w systemie Android](#)

Własne schematy identyfikatorów URI w systemie Android

Do uruchamiania aplikacji AIR ze strony internetowej lub z natywnej aplikacji systemu Android można używać własnego schematu identyfikatora URI. Obsługa własnych identyfikatorów URI jest oparta na filtrach metody konwersji określonych w manifeście systemu Android, dlatego tej techniki nie można używać na innych platformach.

Aby użyć własnego identyfikatora URI, do deskryptora aplikacji należy dodać w bloku `<android>` filtr metody konwersji. W poniższym przykładzie należy określić oba elementy `intent-filter`. Wyrażenie `<data android:scheme="my-customuri" />` należy edytować w taki sposób, aby odzwierciedlało ono ciąg URI dla danego własnego schematu.


```
<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>
```

Filtr metody konwersji informuje system operacyjny Android, że aplikacja jest dostępna do wykonania danej operacji. W przypadku własnego schematu URI oznacza to, że użytkownik kliknął łącze korzystające z tego schematu URI (a przeglądarka nie wie, jak obsłużyć schemat).

W przypadku wywołania aplikacji przez własny schemat URI obiekt `NativeApplication` wywołuje zdarzenie `invoke`. Adres URL łącza, łącznie z parametrami zapytania, jest umieszczony w tablicy `arguments` obiektu `InvokeEvent`. Można używać dowolnej liczby filtrów metod konwersji.

Uwaga: Łącza w wystąpieniu klasy `StageWebView` nie umożliwiają otwierania adresów URL korzystających z własnego schematu URI.

Więcej tematów Pomocy

[Filtry metod konwersji systemu Android](#)

[Kategorie i operacje w systemie Android](#)

Filtrowanie zgodności w systemie Android

Aby określić, czy dana aplikacja jest zgodna z danym urządzeniem, system operacyjny Android korzysta z szeregu elementów w pliku manifestu tej aplikacji. Dodawanie tych informacji do manifestu jest opcjonalne. Jeśli te elementy nie zostaną dodane, aplikację będzie można instalować na dowolnym urządzeniu z systemem Android. Jednak może ona nie działać prawidłowo na pewnych urządzeniach z systemem Android. Na przykład aplikacja do obsługi kamery nie będzie pomocna na telefonie, który nie jest wyposażony w kamerę.

Niektóre znaczniki manifestu systemu Android, których można używać do filtrowania:

- `supports-screens`
- `uses-configuration`
- `uses-feature`
- `uses-sdk` (w środowisku AIR 3 lub nowszym)

Aplikacje korzystające z kamery

W przypadku żądania dla aplikacji uprawnień dotyczących kamery system Android zakłada, że aplikacja wymaga wszystkich dostępnych funkcji kamery, łącznie z automatycznym ustawianiem ostrości i lampą błyskową. Jeśli aplikacja nie wymaga wszystkich funkcji kamery lub jeśli kamera jest funkcją opcjonalną, należy ustawić dla kamery poszczególne elementy `uses-feature` w celu określenia, że są one opcjonalne. W przeciwnym razie użytkownicy urządzeń, które nie mają jednej funkcji lub które w ogóle nie mają kamery, nie będą mogli znaleźć danej aplikacji w sklepie Android Market.

Poniższy przykład ilustruje, w jaki sposób zażądać uprawnień dotyczącego kamery i uczynić wszystkie funkcje aparatu opcjonalnymi.

```
<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>
```

Aplikacje do nagrywania dźwięku

W przypadku żądania uprawnień do nagrywania dźwięku system Android zakłada również, że aplikacja wymaga mikrofonu. Jeśli nagrywanie dźwięku jest opcjonalną funkcją aplikacji, można określić, że mikrofon nie jest konieczny, dodając znacznik `uses-feature`. W przeciwnym razie użytkownicy urządzeń, które nie mają mikrofonu, nie będą mogli znaleźć danej aplikacji w sklepie Android Market.

Poniższy przykład ilustruje sposób żądania uprawnień do używania mikrofonu, a jednocześnie ustawienia obsługi sprzętowej mikrofonu jako opcjonalnej.

```
<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>
```

Więcej tematów Pomocy

[Programiści aplikacji dla systemu Android: Zgodność w systemie Android](#)

[Programiści aplikacji dla systemu Android: Stałe nazw funkcji w systemie Android](#)

Lokalizacja instalacji

Ustawiając dla atrybutu `installLocation` elementu `manifest` systemu Android wartość `auto` lub `preferExternal`, można zezwolić na instalację lub przeniesienie aplikacji na zewnętrzną kartę pamięci.

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

System operacyjny Android nie gwarantuje, że aplikacja zostanie zainstalowana w pamięci zewnętrznej. Użytkownik może również przenosić aplikację między pamięcią wewnętrzną i zewnętrzną za pomocą aplikacji ustawień systemowych.

Nawet w przypadku zainstalowania w pamięci zewnętrznej bufor aplikacji oraz dane użytkownika, takie jak zawartość katalogu magazynu aplikacji, obiekty udostępnione i pliki tymczasowe, są nadal przechowywane w pamięci wewnętrznej. W celu uniknięcia zbyt dużego użycia pamięci wewnętrznej należy rozsądnie wybierać dane zapisywane w katalogu magazynu aplikacji. Duże ilości danych należy zapisywać na karcie SD, korzystając z lokalizacji `File.userDirectory` lub `File.documentsDirectory` (które w systemie Android są odwzorowywane na katalog główny karty SD).

Włączanie programu Flash Player i innych dodatków plug-in w obiekcie StageWebView

W systemie Android 3.0+ aplikacja musi włączyć przyspieszanie sprzętowe w elemencie aplikacji Android, aby zawartość dodatku plug-in była wyświetlana w obiekcie StageWebView. Aby włączyć renderowanie dodatków plug-in, należy ustawić dla atrybutu `android:hardwareAccelerated` elementu `application` wartość `true`.

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

Głębina koloru

AIR 3 i nowsze wersje

Środowisko wykonawcze AIR 3 i nowsze wersje ustawiają renderowanie kolorów 32-bitowych. We wcześniejszych wersjach środowiska wykonawczego AIR stosowane są kolory 16-bitowe. Aby określić, że środowisko wykonawcze ma używać kolorów 16-bitowych, można użyć elementu `<colorDepth>` w deskrytorze aplikacji:

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

Stosowanie 16-bitowej głębi koloru może zwiększyć wydajność renderowania, lecz obniża jakość kolorów.

Ustawienia w systemie iOS

Ustawienia odnoszące się wyłącznie do urządzeń z systemem iOS są umieszczane w elemencie `<iPhone>` w deskrypcyjach aplikacji. Element `iPhone` może mieć elementy potomne `InfoAdditions`, `requestedDisplayResolution`, `Entitlements`, `externalSwfs` i `forceCpuRenderModeForDevices`.

Element `InfoAdditions` umożliwia określanie par klucz-wartość, które są dodawane do pliku ustawień `Info.plist` aplikacji. Na przykład poniższe wartości powodują ustawienie stylu paska stanu aplikacji oraz określają, że aplikacja nie wymaga stałego dostępu do sieci Wi-Fi.

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

Ustawienia `InfoAdditions` są umieszczone w znaczniku `CDATA`.

Element `Entitlements` umożliwia określenie par klucz-wartość dodanych do pliku ustawień `Entitlements.plist` aplikacji. Plik ustawień `Entitlements.plist` udostępnia aplikacjom niektóre funkcje systemu iOS, takie jak powiadomienia w trybie push.

Więcej informacji na temat plików ustawień `Info.plist` i `Entitlements.plist` zawiera dokumentacja dla programistów udostępniana przez firmę Apple.

Zadania pomocnicze działające w tle w systemie iOS

AIR 3.3

Od wersji 3.3 środowisko Adobe AIR obsługuje mechanizm wielozadaniowy systemu iOS. W związku z tym są dostępne pewne opcje związane z działaniem w tle:

- Dźwięk
- Aktualizacje lokalizacji
- Operacje sieciowe
- Wyłączenie wykonywania aplikacji w tle

Uwaga: W przypadku wersji 21 i starszych formatu SWF środowisko AIR nie obsługuje wykonywania w tle w systemach iOS i Android, gdy jest ustawiony bezpośredni tryb renderowania. Z powodu tego ograniczenia aplikacje używające mechanizmu Stage3D nie mogą realizować zadań w tle, takich jak odtwarzanie dźwięku, aktualizacja lokalizacji czy wysyłanie lub pobieranie danych przez sieć. System iOS nie umożliwia wywoływania procedur interfejsu OpenGL ES ani realizacji innych funkcji wyświetlania obrazu w tle. Próby wykonania takich operacji są przerywane przez system iOS. Takie ograniczenie nie występuje w systemie Android, który umożliwia wywoływanie w tle procedur interfejsu OpenGL ES lub innych zadań, na przykład odtwarzania dźwięku. W przypadku wersji 22 i starszych formatu SWF aplikacje AIR dla urządzeń przenośnych mogą działać w tle, gdy jest ustawiony bezpośredni tryb renderowania. Środowisko wykonawcze AIR w systemie iOS powoduje błąd języka (3768 — nie można używać interfejsu API obiektu Stage3D podczas działania w tle) w przypadku wywołań OpenGL ES w tle. W systemie Android nie występują takie błędy, ponieważ jego aplikacje natywne mogą używać wywołań OpenGL ES podczas działania w tle. Aby optymalnie używać zasobu na urządzeniu przenośnym, nie należy stosować wywołań renderowania, gdy aplikacja działa w tle.

Dźwięk w tle

Aby włączyć odtwarzanie i nagrywanie dźwięków w tle, należy dołączyć następującą parę kluczy do elementu

InfoAdditions:

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

Aktualizacje lokalizacji w tle

Aby włączyć aktualizacje lokalizacji w tle, należy dołączyć następującą parę kluczy do elementu InfoAdditions:

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

Uwaga: *Tej funkcji należy używać tylko wtedy, gdy jest potrzebna, ponieważ stosowanie interfejsów API lokalizacji powoduje zużycie dużej ilości energii akumulatora.*

Operacje sieciowe w tle

Aby wykonywać w tle krótkie zadania, aplikacja powinna ustawić właściwość

`NativeApplication.nativeApplication.executeInBackground` na `true`.

Użytkownik może na przykład przenieść aplikację do tła (przejdź do innej aplikacji) po rozpoczęciu operacji wysyłania pliku. Gdy aplikacja odbierze zdarzenie ukończenia wysyłania, może przestawić właściwość

`NativeApplication.nativeApplication.executeInBackground` na `false`.

Ustawienie wartości `true` dla właściwości `NativeApplication.nativeApplication.executeInBackground` nie gwarantuje, że aplikacja będzie działać dowolnie długo. System iOS przydziela limit czasowy dla zadań działających w tle. Gdy system iOS zatrzymuje przetwarzanie w tle, środowisko AIR wywołuje zdarzenie

`NativeApplication.suspend`.

Wyłączanie wykonywania operacji w tle

Aplikacja może jawnie zrezygnować z działania w tle. W tym celu należy dołączyć następującą parę kluczy do elementu

InfoAdditions:

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

Zarezerwowane ustawienia InfoAdditions systemu iOS

Środowisko AIR ustawia w tworzonym pliku Info.plist kilka pozycji w celu zapewnienia prawidłowego działania funkcji środowiska wykonawczego i aplikacji. Nie można definiować następujących ustawień:

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

Uwaga: Można zdefiniować parametr `MinimumOSVersion`. Definicja `MinimumOSVersion` funkcjonuje w środowisku AIR 3.3 lub nowszym.

Obsługa różnych modeli urządzeń z systemem iOS

W celu obsługi tabletu iPad należy podać prawidłowe ustawienia klucz-wartość dla ustawienia `UIDeviceFamily` w elemencie `InfoAdditions`. `UIDeviceFamily` jest tablicą ciągów znaków. Każdy ciąg definiuje obsługiwane urządzenia. Ustawienie `<string>1</string>` oznacza obsługę urządzeń iPhone i iPod Touch. Ustawienie `<string>2</string>` oznacza obsługę tabletu iPad. Ustawienie `<string>3</string>` oznacza obsługę platformy tvOS. W wypadku określenia tylko jednego z tych ustawień, obsługiwana będzie wyłącznie wskazana w ten sposób rodzina urządzeń. Na przykład poniższe ustawienie oznacza, że obsługiwany będzie tylko iPad:

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

Następujące ustawienie obsługuje obie rodziny urządzeń (iPhone/iPod Touch oraz iPad):

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

Ponadto w środowisku AIR 3.7 lub nowszym można za pomocą znacznika `forceCpuRenderModeForDevices` wymusić tryb renderowania przy użyciu procesora dla określonego zbioru urządzeń, a także włączyć tryb renderowania oparty na GPU dla pozostałych urządzeń z systemem iOS.

Ten znacznik można dodać jako obiekt potomny znacznika `iPhone`, wskazując rozdzieloną spacjami listę nazw modeli urządzeń. Listę prawidłowych nazw modeli urządzeń można znaleźć w opisie elementu „`forceCpuRenderModeForDevices`” na stronie 237.

Aby na przykład użyć trybu procesora w przypadku starszych odtwarzaczy iPod, telefonów iPhone i tabletów iPad, a jednocześnie włączyć tryb GPU dla wszystkich pozostałych urządzeń, w deskrypcji aplikacji należy określić następujące ustawienia:

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

Wyświetlacze o wysokiej rozdzielczości

Element `requestedDisplayResolution` określa, czy na urządzeniach z systemem iOS wyposażonych w wyświetlacz o wysokiej rozdzielczości aplikacja powinna korzystać z trybu rozdzielczości *standard* czy *high*.

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

W trybie wysokiej rozdzielczości można oddzielnie odwoływać się do każdego piksela na wyświetlaczu o wysokiej rozdzielczości. W trybie standardowym wyświetlacz urządzenia będzie dla aplikacji wyświetlaczem o standardowej rozdzielczości. W tym trybie narysowanie pojedynczego piksela spowoduje ustawienie koloru czterech pikseli na wyświetlaczu o wysokiej rozdzielczości.

Ustawienie domyślne to `standard`. W aplikacjach dla urządzeń z systemem iOS element `requestedDisplayResolution` jest elementem potomnym elementu `iPhone` (nie elementu `InfoAdditions` ani `initialWindow`).

Aby używać różnych ustawień na różnych urządzeniach, należy określić wartość domyślną jako wartość elementu `requestedDisplayResolution`. Korzystając z atrybutu `excludeDevices`, można wskazać urządzenia, na których ma być używana wartość odwrotna. Poniższy przykładowy kod stosuje tryb wysokiej rozdzielczości na obsługujących go urządzeniach oprócz tabletów iPad 3. generacji, na których jest używany tryb standardowy.

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

Atrybut `excludeDevices` jest dostępny w środowisku AIR 3.6 lub nowszym.

Więcej tematów Pomocy

„`requestedDisplayResolution`” na stronie 250

[Renaun Erickson: Programowanie dla urządzeń z systemem iOS z ekranami Retina i bez nich przy użyciu środowiska AIR 2.6](#)

Własne schematy identyfikatorów URI w systemie iOS

Rejestrując własny schemat identyfikatora URI, można pozwolić na wywoływanie aplikacji przez łącze na stronie internetowej lub przez inną aplikację natywną na urządzeniu. Aby zarejestrować schemat identyfikatora URI, należy dodać do elementu `InfoAdditions` klucz `CFBundleURLTypes`. W poniższym przykładzie jest rejestrowany schemat identyfikatora URI o nazwie `com.example.app`, dzięki czemu aplikacja może być wywoływana przez adresy URL w postaci `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>
```

W przypadku wywołania aplikacji przez własny schemat URI obiekt `NativeApplication` wywołuje zdarzenie `invoke`. Adres URL łączy, łącznie z parametrami zapytania, jest umieszczany w tablicy `arguments` obiektu `InvokeEvent`. Można korzystać z dowolnej liczby własnych schematów identyfikatora URI.

Uwaga: Łączy w wystąpieniu klasy `StageWebView` nie umożliwiają otwierania adresów URL korzystających z własnego schematu URI.

Uwaga: Jeśli dany schemat identyfikatora URI został już zarejestrowany dla innej aplikacji, nie ma możliwości zarejestrowania go dla własnej aplikacji przez zastąpienie poprzedniej.

Filtrowanie zgodności w systemie iOS

Jeśli dana aplikacja powinna być używana wyłącznie na urządzeniach o konkretnych funkcjach sprzętowych lub programowych, należy dodać odpowiednie wpisy do tablicy `UIRequiredDeviceCapabilities` w elemencie `InfoAdditions`. Na przykład poniższy wpis wskazuje, że aplikacja wymaga aparatu fotograficznego i mikrofonu.

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

Jeśli urządzenie nie jest wyposażone w odpowiednie funkcje, nie można zainstalować aplikacji. Niektóre ustawienia funkcji odnoszące się do aplikacji AIR:

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

Środowisko AIR 2.6+ automatycznie dodaje do listy wymaganych funkcji pozycje `armv7` i `opengles-2`.

Uwaga: Nie ma potrzeby dodawania tych funkcji w deskrytorze aplikacji, aby aplikacja mogła ich używać. Ustawień `UIRequiredDeviceCapabilities` należy używać wyłącznie po to, aby uniemożliwić użytkownikom instalowanie aplikacji na urządzeniach, na których nie może ona prawidłowo działać.

Kończenie zamiast wstrzymywania

Gdy użytkownik przełącza się z aplikacji AIR, przechodzi ona do trybu działania w tle i jest wstrzymywana. Jeśli zamiast wstrzymania danej aplikacji ma nastąpić jej zakończenie, dla właściwości `UIApplicationExitsOnSuspend` należy ustawić wartość `YES`:


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

Minimalizowanie ilości pobieranych danych przez wczytywanie zewnętrznych plików SWF zawierających tylko zasoby

AIR 3.7

Aby zminimalizować wstępny rozmiar pobieranej aplikacji, można spakować podzbiór plików SWF używanych przez aplikację, a do wczytywania pozostałych zewnętrznych plików SWF (zawierających tylko zasoby) stosować metodę `Loader.load()` w czasie wykonywania. W celu skorzystania z tej funkcji trzeba spakować aplikację w taki sposób, aby narzędzie ADT przeniosło cały kod bajtowy ActionScript (ABC) z zewnętrznych plików SWF do głównego pliku SWF aplikacji, co pozwoli uzyskać plik SWF zawierający tylko zasoby. Sposób ten zapewnia zgodność z obowiązującą w sklepie Apple Store regułą, która zabrania pobierania jakiegokolwiek kodu do zainstalowanej aplikacji.

Aby umożliwić obsługę wczytywania zewnętrznych plików SWF (zwanych również uproszczonymi plikami SWF), narzędzie ADT wykonuje następujące czynności:

- Odczytuje plik tekstowy określony w elemencie `<iPhone>` przy użyciu elementu potomnego `<externalSwfs>` w celu uzyskania dostępu do rozdzielonej znakami końca wiersza listy plików SWF, które mają być wczytywane w czasie wykonywania:

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

- Przenosi kod ABC z wszystkich zewnętrznie wczytywanych plików SWF do głównego pliku wykonywalnego.
- Pomija zewnętrznie wczytywane pliki SWF w pliku IPA.
- Kopiuje uproszczone pliki SWF do katalogu `.remoteStrippedSWFs`. Programista udostępnia te pliki SWF na serwerze internetowym. W razie potrzeby aplikacja wczytuje je w czasie wykonywania.

Programista wskazuje pliki SWF, które mają być wczytywane w czasie wykonywania, podając ich nazwy w oddzielnych wierszach pliku tekstowego, tak jak pokazano w następującym przykładzie:

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

Ścieżka do pliku jest podana względem pliku deskryptora aplikacji. Ponadto w poleceniu `adt` należy wskazać te pliki SWF jako zasoby.

Uwaga: Ta funkcja dotyczy tylko pakowania standardowego. W przypadku pakowania szybkiego (na przykład z użyciem interpretera, symulatora lub debugera) narzędzie ADT nie tworzy uproszczonych plików SWF.



Więcej informacji o tej funkcji oraz przykładowy kod zawiera wpis na blogu autorstwa Abhinava Dhandha z firmy Adobe: [Przechowywanie dodatkowych plików SWF aplikacji AIR dla systemu iOS na serwerze zewnętrznym](#).

Obsługa sprawdzania lokalizacji geograficznej

W celu korzystania z lokalizacji geograficznej dodaj jedną z następujących par klucz-wartość do elementu `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>
```

Ikony aplikacji

Poniższa tabela zawiera listę rozmiarów ikon stosowanych na poszczególnych platformach urządzeń przenośnych.

Rozmiar ikon	Platforma
29 x 29	iOS
36 x 36	Android
40 x 40	iOS
48 x 48	Android, iOS
50x50	iOS
57 x 57	iOS
58x58	iOS
60 x 60	iOS
72 x 72	Android, iOS
75x75	iOS
76x76	iOS
80x80	iOS
87x87	iOS
96x96	Android
100x100	iOS
114x114	iOS
120 x 120	iOS
144x144	Android, iOS
152x152	iOS
167 x 167	iOS
180x180	iOS
192x192	Android
512x512	Android, iOS
1024x1024	iOS

Ścieżkę do plików ikon należy określić w elemencie icon pliku deskryptora aplikacji.

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

Jeśli nie zostanie udostępniona ikona w danym rozmiarze, zostanie użyta większa ikona o najbliższym rozmiarze, która zostanie przeskalowana w celu dopasowania do wymaganej wielkości.

Ikony w systemie Android

W systemie Android ikony określone w deskrypcji aplikacji są używane jako ikony uruchamiające aplikacje. Ikona modułu uruchamiania aplikacji powinna zostać udostępniona w postaci obrazów PNG o rozmiarach 36×36, 48×48, 72×72, 96×96, 144×144 i 192×192 pikseli. Te rozmiary ikon są używane odpowiednio dla wyświetlaczy o niskiej, średniej i wysokiej gęstości.

Programiści muszą przesłać ikonę o rozmiarze 512×512 pikseli w momencie przesłania aplikacji do sklepu Google Play.

Ikony w systemie iOS

W przypadku aplikacji dla systemu iOS ikony zdefiniowane w deskrypcji aplikacji są używane w następujących miejscach:

- Ikona 29 na 29 pikseli — ikona wyszukiwania Spotlight dla telefonów iPhone/odtwarzaczy iPod o niższej rozdzielczości oraz ikona ustawień dla tabletu iPad o niższej rozdzielczości.
- Ikona 40 na 40 pikseli — ikona wyszukiwania Spotlight dla tabletów iPad o niższej rozdzielczości.
- Ikona 48 na 48 pikseli — środowisko AIR dodaje obramowanie do tego obrazu i używa go jako ikony wyszukiwania Spotlight o rozmiarze 50 x 50 na tabletach iPad o niższej rozdzielczości.
- Ikona 50 na 50 pikseli — wyszukiwanie Spotlight dla tabletów iPad o niższej rozdzielczości.
- Ikona 57 na 57 pikseli — wyszukiwanie Spotlight dla telefonów iPhone/odtwarzaczy iPod o niższej rozdzielczości.
- Ikona 58 na 58 pikseli — ikona wyszukiwania Spotlight dla telefonów iPhone/odtwarzaczy iPod z wyświetlaczem Retina oraz ikona ustawień dla tabletów iPad z wyświetlaczem Retina.
- Ikona 60 na 60 pikseli — ikona aplikacji dla telefonów iPhone/odtwarzaczy iPod o niższej rozdzielczości.
- Ikona 72 na 72 piksele (opcjonalna) — ikona aplikacji dla tabletów iPad o niższej rozdzielczości.
- Ikona 76 na 76 piksele (opcjonalna) — ikona aplikacji dla tabletów iPad o niższej rozdzielczości.
- Ikona 80 na 80 pikseli — ikona wyszukiwania Spotlight dla telefonów iPhone/odtwarzaczy iPod/tabletów iPad.
- Ikona 100 na 100 pikseli — wyszukiwanie Spotlight dla tabletów iPad z wyświetlaczem Retina.
- Ikona 114 na 114 pikseli — ikona aplikacji dla telefonów iPhone/odtwarzaczy iPod z wyświetlaczem Retina.
- Ikona 120 na 120 pikseli — wyszukiwanie Spotlight dla telefonów iPhone/odtwarzaczy iPod o wyższej rozdzielczości.
- Ikona 152 na 152 piksele — ikona aplikacji dla tabletów iPad o wyższej rozdzielczości.
- Ikona 167 na 167 pikseli — wyszukiwanie Spotlight dla tabletów iPad Pro o wyższej rozdzielczości.
- Ikona 512 na 512 pikseli — ikona aplikacji dla urządzeń (iPhone/iPod/iPad) o niższej rozdzielczości. Ta ikona jest wyświetlana w programie iTunes. 512-pikselowy plik PNG używany jest tylko do testowania wersji roboczych aplikacji. Przesyłając ostateczną wersję aplikacji do serwisu Apple App Store, obraz 512-pikselowy dołącza się osobno jako plik JPG. Nie jest on uwzględniany w pliku IPA.
- Ikona 1024 na 1024 piksele — ikona aplikacji dla urządzeń (iPhone/iPod/iPad) z wyświetlaczem Retina.

System iOS wzbogaci ikonę o efekt poświaty. Nie ma potrzeby stosowania efektu względem obrazu źródłowego. W celu eliminacji tego efektu domyślnego należy do elementu `InfoAdditions` w pliku deskryptora aplikacji dodać następujący kod.

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

Uwaga: W systemie iOS metadane aplikacji są wstawiane do ikon aplikacji jako metadane plików PNG. W ten sposób firma Adobe może śledzić liczbę aplikacji AIR dostępnych w sklepie iOS App Store firmy Apple. Jeśli aplikacja nie ma być rozpoznawana jako aplikacja AIR na podstawie tych metadanych ikony, należy rozpakować plik IPA, usunąć metadane z ikony i ponownie spakować plik. Odpowiednią procedurę opisano w artykule [Rezygnowanie z analityki aplikacji AIR w systemie iOS](#).

Więcej tematów Pomocy

„icon” na stronie 239

„imageNxN” na stronie 240

[Programiści aplikacji dla systemu Android: Wytyczne dotyczące projektowania ikon](#)

[Wytyczne dotyczące interfejsu użytkownika w systemie iOS: Wytyczne dotyczące tworzenia własnych ikon i obrazów](#)

Obrazy uruchamiania w systemie iOS

Oprócz ikon aplikacji należy także zapewnić co najmniej jeden obraz uruchamiania: *Default.png*. Opcjonalnie można dołączyć oddzielne obrazy uruchamiania dla różnych orientacji uruchamiania, rozdzielczości (w tym dla wyświetlacza Retina o wysokiej rozdzielczości i dla proporcji 16:9) oraz urządzeń. Można również dołączyć różne obrazy uruchamiania przeznaczone do użytku w sytuacji, gdy aplikacja jest wywoływana za pośrednictwem adresu URL.

Łącza do plików obrazów uruchamiania nie są umieszczane w deskrytorze aplikacji — te pliki należy umieścić w katalogu głównym aplikacji. (Nie należy umieszczać tych plików w podkatalogu).

Schemat nazywania plików

Obraz należy nazwać zgodnie z następującym schematem:

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

Fragment *basename* jest jedyną wymaganą częścią nazwy pliku. Ma on postać *Default* (z wielką literą D) lub jest nazwą określoną za pomocą klucza `UILaunchImageFile` w elemencie `InfoAdditions` w deskrytorze aplikacji.

Fragment *screen size modifier* wskazuje rozmiar ekranu w sytuacji, gdy nie jest stosowany jeden ze standardowych rozmiarów. Ten modyfikator dotyczy tylko modeli telefonu iPhone i odtwarzacza iPod touch z ekranami o proporcjach 16:9, takich jak telefon iPhone 5 i odtwarzacz iPod touch (5. generacji). Jedyną obsługiwaną wartością tego modyfikatora jest `-568h`. Te urządzenia obsługują wyświetlacze o wysokiej rozdzielczości (Retina), dlatego modyfikator rozmiaru ekranu jest zawsze używany z obrazem uwzględniającym modyfikator skali `@2x`. Cała domyślna nazwa obrazu uruchamiania dla tych urządzeń ma postać `Default-568h@2x.png`.

Fragment *urischeme* jest ciągiem służącym do określenia schematu identyfikatora URI. Ten fragment dotyczy tylko aplikacji obsługujących co najmniej jeden własny schemat adresów URL. Jeśli na przykład aplikację można wywołać za pośrednictwem łącza takiego jak `example://foo`, wówczas jako fragmentu schematu nazwy pliku obrazu uruchamiania należy użyć wartości `-example`.

Fragment *orientation* umożliwia ustawienie kilku obrazów uruchamiania dostosowanych do różnych orientacji urządzenia. Ten fragment dotyczy tylko obrazów w aplikacjach dla tabletu iPad. Może on mieć jedną z poniższych wartości oznaczających orientację urządzenia w momencie uruchamiania aplikacji:

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

Fragment *scale* ma wartość @2x (dla telefonu iPhone 4, 5 lub 6) lub @3x (dla telefonu iPhone 6 plus) w przypadku obrazów uruchamiania używanych na wyświetlaczach o wysokiej rozdzielczości (Retina). W przypadku obrazów używanych na wyświetlaczach o standardowej rozdzielczości należy całkowicie pominąć fragment *scale*. Jeśli obraz uruchamiania jest przeznaczony dla wyższych urządzeń, takich jak telefon iPhone 5 i odtwarzacz iPod touch (5. generacji), należy również ustawić modyfikator rozmiaru ekranu -528h po fragmencie *basename* i przed kolejnymi fragmentami.

Fragment *device* służy do określania obrazów uruchamiania dla urządzeń przenośnych i telefonów. Należy go użyć, jeśli aplikacja jest uniwersalna — za pomocą jednego pliku binarnego obsługuje zarówno małe urządzenia, jak i tablety. Dostępne wartości to ~ipad i ~iphone (zarówno dla telefonu iPhone, jak i dla odtwarzacza iPod Touch).

W przypadku telefonu iPhone można stosować wyłącznie obrazy o proporcjach pionowych. W przypadku telefonu iPhone 6 plus można dodawać obrazy poziome. Dla urządzeń o rozdzielczości standardowej należy stosować obrazy o wymiarach 320 x 480 pikseli. Dla urządzeń o wysokiej rozdzielczości należy używać obrazów o wymiarach 640 x 960 pikseli. W przypadku urządzeń o proporcjach 16:9, takich jak telefon iPhone 5 i odtwarzacz iPod touch (5. generacji) należy używać obrazy o wymiarach 640 x 1136 pikseli.

W przypadku tabletu iPad obrazy można dołączyć w następujący sposób:

- AIR 3.3 i starsze wersje, obrazy niepełnoekranowe: Można dołączyć obrazy o orientacji poziomej (1024 x 748 dla normalnej i 2048 x 1496 dla wysokiej rozdzielczości) oraz pionowej (768 x 1004 dla normalnej i 1536 x 2008 dla wysokiej rozdzielczości).
- AIR 3.4 i nowsze wersje, obrazy pełnoekranowe: Można dołączyć obrazy o orientacji poziomej (1024 x 768 dla normalnej i 2048 x 1536 dla wysokiej rozdzielczości) oraz pionowej (768 x 1024 dla normalnej i 1536 x 2048 dla wysokiej rozdzielczości). Uwaga: W przypadku spakowania obrazu pełnoekranowego dla aplikacji niepełnoekranowej górne 20 pikseli (lub 40 pikseli w wysokiej rozdzielczości) zostanie zajęte przez pasek stanu. Należy unikać wyświetlania informacji w tym obszarze.

Przykłady

Poniższa tabela przedstawia przykładowy zestaw obrazów uruchamiania możliwych do dołączenia do hipotetycznej aplikacji obsługującej jak najszerszy zakres urządzeń i orientacji, którą można uruchamiać za pośrednictwem adresów URL przy użyciu schematu `example://`.

Nazwa pliku	Rozmiar obrazu	Zastosowanie
Default.png	320 x 480	Telefon iPhone, standardowa rozdzielczość
Default@2x.png	640 x 960	Telefon iPhone, wysoka rozdzielczość
Default-568h@2x.png	640 x 1136	iPhone, wysoka rozdzielczość, proporcje 16:9

Nazwa pliku	Rozmiar obrazu	Zastosowanie
Default-Portrait.png	768 x 1004 (środowisko AIR 3.3 i starsze wersje) 768 x 1024 (środowisko AIR 3.4 i nowsze wersje)	Tablet iPad, orientacja pionowa
Default-Portrait@2x.png	1536 x 2008 (środowisko AIR 3.3 i starsze wersje) 1536 x 2048 (środowisko AIR 3.4 i nowsze wersje)	iPad, wysoka rozdzielczość, orientacja pionowa
Default-PortraitUpsideDown.png	768 x 1004 (środowisko AIR 3.3 i starsze wersje) 768 x 1024 (środowisko AIR 3.4 i nowsze wersje)	Tablet iPad, orientacja pionowa do góry nogami
Default-PortraitUpsideDown@2x.png	1536 x 2008 (środowisko AIR 3.3 i starsze wersje) 1536 x 2048 (środowisko AIR 3.4 i nowsze wersje)	iPad, wysoka rozdzielczość, odwrócona orientacja pionowa
Default-Landscape.png	1024 x 768	Tablet iPad, orientacja pozioma lewa
Default-LandscapeLeft@2x.png	1536 x 2048	iPad, wysoka rozdzielczość, orientacja pozioma lewostronna
Default-LandscapeRight.png	1024 x 768	Tablet iPad, orientacja pozioma prawa
Default-LandscapeRight@2x.png	1536 x 2048	iPad, wysoka rozdzielczość, orientacja pozioma prawostronna
Default-example.png	320 x 480	Adres URL example:// na standardowym telefonie iPhone
Default-example@2x.png	640 x 960	Adres URL example:// na telefonie iPhone o wysokiej rozdzielczości
Default-example~ipad.png	768 x 1004	Adres URL example:// na tablecie iPad z orientacją pionową
Default-example-Landscape.png	1024 x 768	Adres URL example:// na tablecie iPad z orientacją poziomą

Ten przykład przedstawia tylko jedno rozwiązanie. Można na przykład zastosować obraz `Default.png` dla tabletu iPad i określić odpowiednie obrazy uruchamiania dla telefonu iPhone i odtwarzacza iPod, używając nazw `Default~iphone.png` i `Default@2x~iphone.png`.

Zobacz także

[Przewodnik dotyczący programowania aplikacji dla systemu iOS: Obrazy uruchamiania aplikacji](#)

Obrazy uruchamiania do pakowania dla urządzeń z systemem iOS

Urządzenia	Rozdzielczość (piksele)	Nazwa obrazu uruchamiania	Orientacja
Telefony iPhone			
iPhone 4 (nie Retina)	640 x 960	Default~iphone.png	Pionowa
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Pionowa
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	Pionowa

iPhone 6, iPhone 7	750 x 1334	Default-375w-667h@2x~iphone.png	Pionowa
iPhone 6+, iPhone 7+	1242 x 2208	Default-414w-736h@3x~iphone.png	Pionowa
iPhone 6+, iPhone 7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	Pozioma
Tablety iPad			
iPad 2	768 x 1024	Default-Portrait~ipad.png	Pionowa
iPad 2	768 x 1024	Default-PortraitUpsideDown~ipad.png	Pionowa do góry nogami
iPad 2	1024 x 768	Default-Landscape~ipad.png	Pozioma z lewej strony
iPad 2	1024 x 768	Default-LandscapeRight~ipad.png	Pozioma z prawej strony
iPad 3, Air	1536 x 2048	Default-Portrait@2x~ipad.png	Pionowa
iPad 3, Air	1536 x 2048	Default-PortraitUpsideDown@2x~ipad.png	Pionowa do góry nogami
iPad 3, Air	1536 x 2048	Default-LandscapeLeft@2x~ipad.png	Pozioma z lewej strony
iPad 3, Air	1536 x 2048	Default-LandscapeRight@2x~ipad.png	Pozioma z prawej strony
iPad Pro	2732 x 2048	Default-Portrait@2x.png	Pionowa
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Pozioma

Wytyczne dotyczące grafiki

Możliwe jest utworzenie dowolnej grafiki do zapisania w obrazie uruchamiania. Jedynym warunkiem są prawidłowe wymiary. Zalecane jest jednak, aby obraz odpowiadał stanowi początkowemu aplikacji. Taki obraz uruchamiania można utworzyć, wykonując zrzut ekranu startowego aplikacji.

- 1 Otwórz aplikację na urządzeniu z systemem iOS. Po pojawieniu się pierwszego ekranu interfejsu użytkownika naciśnij i przytrzymaj przycisk Home (pod ekranem). Przytrzymując przycisk Home, naciśnij przycisk Power/Sleep (w górnej części urządzenia). Zrzut ekranu zostanie przesłany do Rolki z aparatu.
- 2 Prześlij obraz na komputer używany do programowania, wysyłając zdjęcia z programu iPhoto lub innej aplikacji do transferu zdjęć.

W obrazie uruchamiania nie należy umieszczać tekstów, jeśli aplikacja ma być lokalizowana na kilka języków. Obraz uruchamiania jest statyczny i w takiej sytuacji jego treść byłaby nieodpowiednia dla innych wersji językowych.

Zobacz także

[Wytyczne dotyczące interfejsu użytkownika w systemie iOS: obrazy uruchamiania](#)

Ustawienia ignorowane

Aplikacje dla urządzeń przenośnych ignorują ustawienia aplikacji, które odnoszą się do natywnych okien lub funkcji komputerowego systemu operacyjnego. Ignorowane są następujące ustawienia:

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder

- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

Pakowanie aplikacji AIR dla urządzeń przenośnych

W przypadku aplikacji AIR dla urządzeń przenośnych do tworzenia pakietu aplikacji należy używać polecenia -package narzędzia ADT. Parametr -target określa platformę dla urządzeń przenośnych, dla której jest tworzony pakiet.

Pakiety systemu Android

Aplikacje AIR w systemie Android korzystają z formatu pakietu systemu Android (APK), a nie z formatu pakietu środowiska AIR.

Pakiety uzyskane za pomocą narzędzia ADT przy użyciu typu docelowego *APK* mają format, który pozwala na ich przesłanie do sklepu Android Market. Sklep Android Market stawia określone wymagania, które muszą spełniać przesyłane do niego aplikacje. Przed utworzeniem ostatecznego pakietu należy zapoznać się z najnowszymi wymaganiami. Zobacz [Programiści aplikacji dla systemu Android: Publikowanie w sklepie Android Market](#).

W przeciwieństwie do aplikacji dla systemu iOS do podpisania aplikacji dla systemu Android można użyć normalnego certyfikatu podpisywania kodu środowiska AIR. Aby przesłać aplikację do sklepu Android Market, certyfikat musi być jednak zgodny z zasadami tego sklepu, które wymagają certyfikatu ważnego co najmniej do roku 2033. Taki certyfikat można utworzyć za pomocą polecenia -certificate narzędzia ADT.

W celu przesłania aplikacji do innego sklepu, który nie zezwala, aby aplikacja wymagała pobrania środowiska AIR ze sklepu firmy Google, można określić alternatywny adres URL pobierania przy użyciu parametru -airDownloadURL narzędzia ADT. Gdy aplikację uruchamia użytkownik, który nie ma wymaganej wersji środowiska wykonawczego AIR, następuje przekierowanie do określonego adresu URL. Więcej informacji zawiera sekcja „[Polecenie package narzędzia ADT](#)” na stronie 176.

Domyślnie ADT pakuje aplikację Android ze współdzielonym środowiskiem wykonawczym. Dlatego w celu uruchomienia aplikacji użytkownik powinien zainstalować na urządzeniu odrębne środowisko wykonawcze AIR.

Uwaga: Aby wymusić utworzenie w programie ADT pliku APK używającego dołączonego środowiska wykonawczego, należy użyć elementu docelowego *apk-captive-runtime*.

Pakiety systemu iOS

Aplikacje AIR w systemie iOS korzystają z formatu pakietów systemu iOS (IPA), a nie z natywnego formatu środowiska AIR.

Pakiety uzyskane za pomocą narzędzia ADT przy użyciu typu docelowego *ipa-app-store*, poprawnego certyfikatu podpisywania kodu oraz profilu informacyjnego są w formacie, który pozwala przesłać je do sklepu App Store firmy Apple. Do pakowania aplikacji w celu szybkiego rozpowszechniania należy używać typu docelowego *ipa-ad-hoc*.

Do podpisywania aplikacji należy używać prawidłowego certyfikatu programisty wydanego przez firmę Apple. Inne certyfikaty są używane do tworzenia wersji testowych, a inne do ostatecznego pakowania przed przesłaniem aplikacji.

Przykład pakowania aplikacji dla systemu iOS za pomocą skryptu Ant — [Piotr Walczyszyn: Pakowanie aplikacji AIR dla urządzeń z systemem iOS przy użyciu polecenia ADT i skryptów ANT](#)

Pakowanie przy użyciu narzędzia ADT

Zestawy SDK środowiska AIR 2.6 i nowszych wersji obsługują pakowanie zarówno dla systemu iOS, jak i Android. Przed przystąpieniem do pakowania należy skompilować kod ActionScript, MXML oraz kod wszystkich rozszerzeń. Jest również potrzebny certyfikat podpisywania kodu.

Szczegółową dokumentację poleceń i opcji narzędzia ADT zawiera sekcja „[Narzędzie ADT](#)” na stronie 175.

Pakiety APK systemu Android

Tworzenie pakietu APK

Do utworzenia pakietu APK należy użyć polecenia `package` narzędzia ADT, ustawiając dla typu docelowego wartość *apk* w przypadku wersji do rozpowszechniania, *apk-debug* w przypadku wersji do debugowania i *apk-emulator* w przypadku kompilacji do rozpowszechniania, które będą uruchamiane w emulatorze.

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

Całe polecenie należy wpisać w jednym wierszu. Znaki podziału wiersza umieszczono w powyższym przykładzie wyłącznie w celu poprawienia czytelności. W powyższym przykładzie założono również, że ścieżka narzędzia ADT znajduje się w definicji ścieżek powłoki wiersza poleceń. (W celu uzyskania pomocy zapoznaj się z tematem „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Pliki aplikacji użyte w tym przykładzie to `myApp-app.xml` (plik deskryptora aplikacji), `myApp.swf` i katalog ikon.

Po uruchomieniu polecenia w przedstawiony sposób narzędzie ADT wyświetli monit o podanie hasła magazynu kluczy. (Znaki hasła nie są wyświetlane. Po zakończeniu wpisywania należy nacisnąć klawisz Enter).

Uwaga: Domyślnie wszystkie aplikacje AIR for Android mają przedrostek pakietu `air..` Aby wyłączyć to zachowanie, ustaw zmienną środowiskową `AIR_NOANDROIDFLAIR` na wartość `true` na komputerze.

Tworzenie pakietu APK dla aplikacji korzystającej z rozszerzeń natywnych

Aby utworzyć pakiet APK dla aplikacji korzystającej z rozszerzeń natywnych, należy dodać opcję `-extdir` oprócz normalnych opcji pakowania. W przypadku wielu plików ANE korzystających ze wspólnych zasobów/bibliotek program ADT wybiera tylko jeden zasób/jedną bibliotekę i ignoruje powielone wpisy, generując ostrzeżenie. Ta opcja określa katalog zawierający pliki ANE używane przez aplikację. Na przykład:

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

Tworzenie pakietu APK zawierającego własną wersję środowiska wykonawczego AIR

Aby utworzyć pakiet APK zawierający aplikację i dołączoną wersję środowiska wykonawczego AIR, należy użyć typu docelowego `apk-captive-runtime`. Ta opcja określa katalog zawierający pliki ANE używane przez aplikację. Na przykład:

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

Potencjalne wady tej techniki:

- Krytyczne poprawki zabezpieczeń nie są automatycznie udostępniane użytkownikom po opublikowaniu przez firmę Adobe.
- Aplikacja wymaga więcej pamięci RAM.

Uwaga: W przypadku dołączenia środowiska wykonawczego narzędzie ADT dodaje uprawnienia `INTERNET` i `BROADCAST_STICKY` do aplikacji. Te uprawnienia są wymagane przez środowisko wykonawcze AIR.

Tworzenie pakietu debugowania APK

Aby utworzyć wersję aplikacji, której można używać z debugerem, należy użyć wartości `apk-debug` dla typu docelowego i określić opcje połączenia.

```
adt -package
                                     -target apk-debug
                                     -connect 192.168.43.45
                                     -storetype pkcs12 -keystore ../codesign.p12
myApp.apk
myApp-app.xml
myApp.swf icons
```

Flaga `-connect` powiadamia środowisko wykonawcze AIR na urządzeniu o tym, gdzie należy nawiązać przez sieć połączenie ze zdalnym debugerem. Aby przeprowadzić debugowanie za pośrednictwem połączenia USB, należy zamiast tego ustawić flagę `-listen`, określając port TCP na potrzeby połączenia z debugerem.

```
adt -package
                                     -target apk-debug
                                     -listen 7936
                                     -storetype pkcs12 -keystore ../codesign.p12
myApp.apk
myApp-app.xml
myApp.swf icons
```

Aby działała większość funkcji debugowania, należy skompilować pliki SWF i SWC przy włączonym debugowaniu. Pełny opis flag `-connect` i `-listen` zawiera rozdział „[Opcje połączenia debugera](#)” na stronie 194.

Uwaga: Domyślnie program ADT integruje środowisko wykonawcze AIR podczas pakowania aplikacji dla systemu Android przy użyciu celu `apk-debug`. Aby wymusić utworzenie w programie ADT pliku APK używającego zewnętrznego środowiska wykonawczego, należy ustawić zmienną środowiskową `AIR_ANDROID_SHARED_RUNTIME` na `true`.

Aby aplikacja w systemie Android mogła nawiązać połączenie z komputerem przeprowadzającym debugowanie przez sieć, musi ona mieć również uprawnienie do uzyskiwania dostępu do Internetu. Zobacz „[Uprawnienia w systemie Android](#)” na stronie 82.

Tworzenie pakietu APK do użytku w emulatorze systemu Android

W emulatorze systemu Android można używać pakietu debugowania APK, ale nie pakietu w trybie do rozpowszechniania. Aby utworzyć pakiet APK przeznaczony do użytku w emulatorze, należy użyć polecenia `package` programu ADT, ustawiając dla typu docelowego wartość `apk-emulator`.

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

W tym przykładzie przyjęto, że ścieżka do narzędzia ADT znajduje się w definicji ścieżki powłoki wiersza poleceń. (W celu uzyskania pomocy zapoznaj się z tematem „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Tworzenie pakietu APK z pliku AIR lub AIRI

Pakiet APK można utworzyć bezpośrednio z istniejącego pliku AIR lub AIRI.

```
adt -target apk -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp.air
```

Plik AIR musi używać przestrzeni nazw AIR 2.5 (lub nowszej) w pliku deskryptora aplikacji.

Tworzenie pakietu APK dla platformy Android x86

Od wersji środowiska AIR 14 argument `-arch` pozwala spakować plik APK dla platformy Android x86. Na przykład:

```
adt -package -target apk-debug -listen 7936 -arch x86 -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp-app.xml myApp.swf icons
```

Pakiety systemu iOS

W przypadku systemu iOS narzędzie ADT konwertuje kod bajtowy pliku SWF i inne pliki źródłowe na natywną aplikację systemu iOS.

- 1 Utwórz plik SWF przy użyciu programu Flash Builder lub Flash Professional albo kompilatora wiersza polecenia.
- 2 Otwórz warstwę poleceń lub terminal i przejdź do folderu projektu aplikacji na telefon iPhone.
- 3 Następnie utwórz plik IPA za pomocą narzędzia ADT, korzystając z poniższej składni.

```
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
```

Zmień odniesienie do aplikacji `adt` w celu uwzględnienia pełnej ścieżki do aplikacji `adt`. Aplikacja ADT zostanie zainstalowana w podkatalogu `bin` zestawu SDK środowiska AIR.

Należy wybrać opcję `-target` odpowiadającą typowi aplikacji na telefon iPhone, którą chce się utworzyć:

- `-target ipa-test` — tę opcję należy wybrać, aby szybko skompilować wersję aplikacji na potrzeby testowania na telefonie iPhone, którym dysponuje programista. Można też użyć opcji `ipa-test-interpreter`, która przyspiesza kompilację, lub opcji `ipa-test-interpreter-simulator`, która pozwala uruchomić symulator systemu iOS.
- `-target ipa-debug` — tę opcję należy wybrać, aby skompilować wersję debugowanej aplikacji na potrzeby testowania na telefonie iPhone, którym dysponuje programista. Ta opcja umożliwia odbieranie wyników instrukcji `trace()` z aplikacji na telefon iPhone w sesji debugowania.

Możliwe jest podanie jednej z następujących opcji `-connect` (OPCJE_POŁĄCZENIA) w celu określenia adresu IP komputera programistycznego, na którym działa debugger:

- `-connect` — aplikacja będzie próbowała nawiązać (przez sieć Wi-Fi) połączenie z sesją debugowania na komputerze używanym do programowania, na którym została skompilowana.
- `-connect ADRES_IP` — aplikacja będzie próbowała nawiązać połączenie (przez sieć Wi-Fi) z sesją debugowania na komputerze o określonym adresie IP. Przykład:
`-target ipa-debug -connect 192.0.32.10`
- `-connect NAZWA_HOSTA` — aplikacja będzie próbowała nawiązać połączenie (przez sieć Wi-Fi) z sesją debugowania na komputerze o określonej nazwie hosta. Na przykład:
`-target ipa-debug -connect bobroberts-mac.example.com`

Opcja `-connect` nie jest wymagana. Jeśli nie zostanie określona, uzyskana aplikacja przeznaczona do debugowania nie będzie próbowała łączyć się z debugerem na hoście. Można także użyć opcji `-listen` zamiast `-connect`, aby włączyć debugowanie przez połączenie USB opisane w rozdziale „Zdalne debugowanie za pomocą programu FDB i połączenia USB” na stronie 113.

Jeśli próba połączenia z sesją debugowania nie powiedzie się, aplikacja wyświetli okno dialogowe z prośbą o wpisanie adresu IP komputera z debugerem. Próba połączenia nie powiedzie się, jeśli urządzenie nie jest podłączone do sieci Wi-Fi. Problem z połączeniem może również wystąpić, jeśli urządzenie jest podłączone, ale nie znajduje się za zaporą sieciową hosta z debugerem.

Można też użyć opcji `ipa-debug-interpreter`, która przyspiesza kompilację, lub opcji `ipa-debug-interpreter-simulator`, która pozwala uruchomić symulator systemu iOS.

Więcej informacji zawiera sekcja „[Debugowanie aplikacji AIR dla urządzeń przenośnych](#)” na stronie 106.

- `-target ipa-ad-hoc` — tę opcję należy wybrać w celu utworzenia aplikacji na potrzeby instalacji ad hoc. Więcej informacji można znaleźć w witrynie Apple Dev Center firmy Apple.
- `-target ipa-app-store` — tę opcję należy wybrać w celu utworzenia ostatecznej wersji pliku IPA z myślą o udostępnieniu w serwisie Apple App Store.

Zastąp element `PROFILE_PATH` ścieżką do pliku profilu informacyjnego dla tworzonej aplikacji. Więcej informacji na temat profili informacyjnych zawiera sekcja „[Konfiguracja w systemie iOS](#)” na stronie 70.

Za pomocą opcji `-platformsdk` można wskazać zestaw SDK symulatora iOS w przypadku tworzenia aplikacji przeznaczonej do uruchamiania w symulatorze systemu iOS.

Zastąp element `SIGNING_OPTIONS` odwołaniem do certyfikatu i hasła programisty aplikacji na telefon iPhone. Należy użyć poniższej składni:

```
-storetype pkcs12 -keystore P12_FILE_PATH -storepass PASSWORD
```

Zastąp element `P12_FILE_PATH` ścieżką do pliku certyfikatu P12. Zastąp element `PASSWORD` hasłem dla certyfikatu. (Patrz przykład poniżej.) Więcej informacji na temat pliku certyfikatu P12 zawiera sekcja „[Konwertowanie certyfikatu programisty na plik magazynu kluczy P12](#)” na stronie 209.

Uwaga: Podczas pakowania dla symulatora systemu iOS można użyć certyfikatu podpisanego automatycznie.

Zastąp element `APP_DESCRIPTOR` odwołaniem do pliku deskryptora aplikacji.

Zastąp element `SOURCE_FILES` odwołaniem do głównego pliku SWF projektu wraz z ewentualnymi innymi zasobami, które mają zostać uwzględnione. Uwzględnij ścieżki do wszystkich plików ikon zdefiniowanych w oknie dialogowym ustawień aplikacji w programie Flash Professional lub we własnym pliku deskryptora aplikacji. Następnie dodaj plik kompozycji ekranu początkowego, `Default.png`.

Opcja `-extdir extension-directory` pozwala określić katalog zawierający pliki ANE (rozszerzenia natywne) używane przez aplikację. Jeśli aplikacja nie korzysta z rozszerzeń natywnych, nie należy dołączać tej opcji.

Ważne: W katalogu aplikacji nie należy tworzyć podkatalogu o nazwie `Resources`. Środowisko wykonawcze automatycznie tworzy folder o tej nazwie, aby zachować zgodność ze strukturą pakietu IPA. Utworzenie własnego folderu `Resources` spowoduje krytyczny konflikt.

Tworzenie przeznaczonego do debugowania pakietu dla systemu iOS

Aby utworzyć pakiet dla systemu iOS przeznaczony do zainstalowania na urządzeniach testowych, należy użyć polecenia `package` narzędzia ADT, ustawiając dla typu docelowego wartość `ios-debug`. Przed wywołaniem tego polecenia należy uzyskać programistyczny certyfikat podpisywania kodu i profil informacyjny od firmy 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
```

Uwaga: Można też użyć opcji `ipa-debug-interpreter`, która przyspiesza kompilację, lub opcji `ipa-debug-interpreter-simulator`, która pozwala uruchomić symulator systemu iOS Simulator.

Całe polecenie należy wpisać w jednym wierszu. Znaki podziału wiersza umieszczono w powyższym przykładzie wyłącznie w celu poprawienia czytelności. W powyższym przykładzie założono również, że ścieżka narzędzia ADT znajduje się w definicji ścieżek powłoki wiersza poleceń. (W celu uzyskania pomocy zapoznaj się z tematem „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Plikami aplikacji w tym przykładzie są myApp-app.xml (plik deskryptora aplikacji), myApp.swf, katalog ikon i plik Default.png.

Do podpisania aplikacji należy użyć poprawnego certyfikatu rozpowszechniania wydanego przez firmę Apple. Nie można używać innych certyfikatów podpisywania.

W przypadku debugowania przez sieć Wi-Fi należy użyć opcji `-connect`. Aplikacja próbuje zainicjować sesję debugowania z programem Flash Debugger (FDB) uruchomionym pod specjalnym adresem IP lub na komputerze o specjalnej nazwie. W przypadku debugowania przez połączenie USB należy użyć opcji `-listen`. Należy najpierw zainstalować aplikację, a następnie uruchomić narzędzie FDB, które zainicjuje sesję debugowania uruchomionej aplikacji. Więcej informacji zawiera sekcja „[Nawiązywanie połączenia z debugerem programu Flash](#)” na stronie 110.

Tworzenie pakietu dla systemu iOS w celu przesłania do sklepu App Store firmy Apple

Aby utworzyć pakiet systemu iOS w celu przesłania do sklepu App Store firmy Apple, należy użyć polecenia package narzędzia ADT, ustawiając dla typu docelowego wartość `ios-app-store`. Przed wywołaniem tego polecenia należy uzyskać certyfikat podpisywania kodu do rozpowszechniania i profil informacyjny od firmy 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
```

Całe polecenie należy wpisać w jednym wierszu. Znaki podziału wiersza umieszczono w powyższym przykładzie wyłącznie w celu poprawienia czytelności. W powyższym przykładzie założono również, że ścieżka narzędzia ADT znajduje się w definicji ścieżek powłoki wiersza poleceń. (W celu uzyskania pomocy zapoznaj się z tematem „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Plikami aplikacji w tym przykładzie są myApp-app.xml (plik deskryptora aplikacji), myApp.swf, katalog ikon i plik Default.png.

Do podpisania aplikacji należy użyć poprawnego certyfikatu rozpowszechniania wydanego przez firmę Apple. Nie można używać innych certyfikatów podpisywania.

Ważne: Firma Apple wymaga używania programu Apple Application Loader do przesyłania programów do sklepu App Store. Firma Apple publikuje program Application Loader wyłącznie dla systemu Mac OS X. Opracowując aplikację AIR na telefon iPhone na komputerze z systemem Windows, w celu przesłania aplikacji do sklepu App Store należy więc mieć dostęp do komputera z systemem OS X (w wersji 10.5.3 lub nowszej). Program Application Loader można uzyskać w witrynie iOS Developer Center firmy Apple.

Tworzenie pakietu dla systemu iOS w celu szybkiego rozpowszechniania

Aby utworzyć pakiet dla systemu iOS przeznaczony do szybkiego rozpowszechniania, należy użyć polecenia package narzędzia ADT, ustawiając dla typu docelowego wartość `ios-ad-hoc`. Przed wywołaniem tego polecenia należy uzyskać odpowiedni certyfikat podpisywania kodu do szybkiego rozpowszechniania i profil informacyjny od firmy 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
```

Całe polecenie należy wpisać w jednym wierszu. Znaki podziału wiersza umieszczono w powyższym przykładzie wyłącznie w celu poprawienia czytelności. W powyższym przykładzie założono również, że ścieżka narzędzia ADT znajduje się w definicji ścieżek powłoki wiersza poleceń. (W celu uzyskania pomocy zapoznaj się z tematem „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Plikami aplikacji w tym przykładzie są myApp-app.xml (plik deskryptora aplikacji), myApp.swf, katalog ikon i plik Default.png.

Do podpisania aplikacji należy użyć poprawnego certyfikatu rozpowszechniania wydanego przez firmę Apple. Nie można używać innych certyfikatów podpisywania.

Tworzenie pakietu systemu iOS dla aplikacji korzystającej z rozszerzeń natywnych

Aby utworzyć pakiet systemu iOS dla aplikacji korzystającej z rozszerzeń natywnych, należy użyć polecenia package narzędzia ADT z opcją `-extdir`. Polecenie narzędzia ADT powinno zostać użyte w sposób odpowiedni dla danego typu docelowego (`ipa-app-store`, `ipa-debug`, `ipa-ad-hoc`, `ipa-test`). Na przykład:

```
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
```

Całe polecenie należy wpisać w jednym wierszu. Znaki podziału wiersza umieszczono w powyższym przykładzie wyłącznie w celu poprawienia czytelności.

Odnośnie rozszerzeń natywnych w przykładzie przyjęto założenie, że katalog o nazwie `extensionsDir` znajduje się wewnątrz katalogu, w którym jest uruchamiane polecenie. Katalog `extensionsDir` zawiera pliki ANE używane przez aplikację.

Debugowanie aplikacji AIR dla urządzeń przenośnych

Aplikację AIR dla urządzeń przenośnych można debugować na kilka sposobów. Najprostszym sposobem wykrycia problemów w logice aplikacji jest debugowanie za pomocą narzędzia ADL na komputerze używanym do programowania lub w symulatorze systemu iOS. Aplikację można również zainstalować na urządzeniu i debugować zdalnie za pomocą debugera Flash działającego na komputerze zdalnym.

Symulowanie urządzenia przy użyciu narzędzia ADL

Najszybszym i najprostszym sposobem testowania i debugowania większości funkcji aplikacji dla urządzeń przenośnych jest uruchomienie aplikacji na komputerze używanym do programowania za pomocą narzędzia Adobe Debug Launcher (ADL). Narzędzie ADL używa elementu `supportedProfiles` z deskryptora aplikacji w celu określenia, który profil ma być stosowany. Jeśli na liście jest więcej niż jeden profil, narzędzie ADL używa pierwszej pozycji na liście. Można również wybrać jeden z innych profili na liście `supportedProfiles` za pomocą parametru `-profile` narzędzia ADL. (Jeśli w deskrytorze aplikacji nie zostanie umieszczony element `supportedProfiles`, wówczas w argumencie `-profile` można określić dowolny profil). Aby na przykład uruchomić aplikację w celu symulacji profilu urządzenia przenośnego, można użyć następującego polecenia:

```
adl -profile mobileDevice myApp-app.xml
```


Gdy profil urządzenia przenośnego jest symulowany na komputerze stacjonarnym w taki sposób, aplikacja działa w środowisku w dużym stopniu odpowiadającym docelowemu urządzeniu przenośnemu. Interfejsy API języka ActionScript, które nie należą do profilu urządzenia przenośnego, nie są dostępne. Narzędzie ADL nie odróżnia jednak możliwości poszczególnych urządzeń przenośnych. Można na przykład wysyłać do aplikacji symulowane naciśnięcia klawiszy programowych nawet wtedy, gdy rzeczywiste urządzenie docelowe nie oferuje funkcji klawiszy programowych.

Narzędzie ADL obsługuje symulacje zmian orientacji urządzenia i danych wejściowych pochodzących z klawiszy programowych przez polecenia menu. W przypadku uruchamiania narzędzia ADL w profilu urządzenia przenośnego w narzędziu jest wyświetlane menu (w oknie aplikacji lub na pasku menu komputera) umożliwiające wpisywanie obrotu urządzenia lub symulowanych danych wejściowych klawiszy programowych.

Dane wejściowe klawiszy programowych

Narzędzie ADL symuluje klawisze programowe odpowiadające przyciskom Wstecz, Menu i Wyszukiwanie na urządzeniu przenośnym. Po uruchomieniu narzędzia ADL przy użyciu profilu urządzenia przenośnego zostaje wyświetlone menu, za pomocą którego można wysyłać impulsy tych klawiszy do symulowanego urządzenia.

Obracanie urządzenia

Po uruchomieniu narzędzia ADL przy użyciu profilu urządzenia przenośnego jest wyświetlane menu, za pośrednictwem którego narzędzie umożliwia symulowanie obracania urządzenia. Symulowane urządzenie można obracać w prawo lub w lewo.

Symulacja obracania ma wpływ wyłącznie na aplikacje obsługujące automatyczną orientację. Tę funkcję można włączyć, ustawiając dla elementu `autoOrients` wartość `true` w deskrypcorze aplikacji.

Rozmiar ekranu

Ustawiając parametr `-screensize` narzędzia ADL, można przetestować aplikację na ekranach o różnych rozmiarach. Można skorzystać z kodu jednego z predefiniowanych typów ekranów lub z ciągu czterech wartości przedstawiających wymiary normalnego i zmaksymalizowanego ekranu.

Zawsze należy określać wymiary w pikselach układu pionowego — z wartością szerokości mniejszą niż wartość wysokości. Na przykład poniższe polecenie spowodowałoby otwarcie narzędzia ADL symulującego ekran używany na telefonie Motorola Droid.

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Listę predefiniowanych typów ekranów można znaleźć w sekcji „[Korzystanie z programu ADL](#)” na stronie 169.

Ograniczenia

Niektórych interfejsów, które nie są obsługiwane w profilu komputera stacjonarnego, nie można symulować za pomocą narzędzia ADL. Niektóre interfejsy API, które nie są symulowane:

- Accelerometer
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- Gesty wielodotykowe i gesty w systemach operacyjnych komputerów stacjonarnych, które nie obsługują tych funkcji
- SystemIdleMode

Jeśli dana aplikacja korzysta z tych klas, należy przetestować te funkcje na rzeczywistym urządzeniu lub w emulatorze.

Istnieją również interfejsy API działające podczas pracy w narzędziu ADL na komputerze, które nie działają na urządzeniach przenośnych pewnych typów. Przykłady:

- Kodek audio Speex i AAC
- Obsługa dostępności i czytnika ekranu
- RTMPE
- Wczytywanie plików SWF zawierających kod bajtowy ActionScript
- Moduły cieniujące PixelBender

Należy pamiętać o przetestowaniu aplikacji korzystających z tych funkcji na urządzeniach docelowych, gdyż narzędzie ADL nie symuluje całkowicie środowiska wykonawczego.

Symulowanie urządzeń przy użyciu symulatora systemu iOS

Symulator systemu iOS (tylko dla komputerów Mac) pozwala szybko uruchamiać i debugować aplikacje przeznaczone dla systemu iOS. Podczas testowania przy użyciu symulatora systemu iOS nie jest potrzebny certyfikat programisty ani profil informacyjny. Nadal trzeba utworzyć certyfikat p12, ale może on mieć podpis automatyczny.

Domyślnie program ADT zawsze uruchamia symulator telefonu iPhone. Aby zmienić urządzenie do symulacji, wykonaj następujące czynności:

- Za pomocą poniższego polecenia wyświetl dostępne symulatory.

```
xcrun simctl list devices
```

Uzyskany wynik będzie podobny do zaprezentowanego poniżej.

```
== 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)
....
```

- Aby wybrać konkretny symulator, ustaw zmienną środowiskową `AIR_IOS_SIMULATOR_DEVICE` w następujący sposób:

```
export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
```

Po ustawieniu zmiennej środowiskowej ponownie uruchom proces i uruchom aplikację na wybranym symulowanym urządzeniu.

Uwaga: W przypadku używania narzędzia ADT z symulatorem systemu iOS należy zawsze dołączyć opcję `-platformsdk`, aby podać ścieżkę zestawu SDK symulatora systemu iOS.

Aby uruchomić aplikację w symulatorze systemu iOS:

- 1 Użyj polecenia `adt -package` z opcją `-target ipa-test-interpreter-simulator` lub `-target ipa-debug-interpreter-simulator` zgodnie z poniższym przykładem.

```
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
```

Uwaga: W przypadku symulatorów nie są już wymagane opcje podpisów. We fladze `-keystore` można podać dowolną wartość, gdyż nie jest ona uwzględniana przez program ADT.

- 2 Zainstaluj aplikację w symulatorze systemu iOS za pomocą polecenia `adt -installApp` zgodnie z poniższym przykładem.

```
adt -installApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -package sample_ipa_name.ipa
```

- 3 Uruchom aplikację w symulatorze systemu iOS za pomocą polecenia `adt -launchApp` zgodnie z poniższym przykładem.

Uwaga: Domyślnie polecenie `adt -launchApp` uruchamia aplikację w symulatorze iPhone'a. Aby uruchomić aplikację w symulatorze iPada, wyeksportuj zmienną środowiskową `AIR_IOS_SIMULATOR_DEVICE = "iPad"`, a następnie użyj polecenia `adt -launchApp`.

```
adt -launchApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -appid sample_ipa_name
```

Aby przetestować rozszerzenie natywne w symulatorze systemu iOS, użyj nazwy platformy `iPhone-x86` w pliku `extension.xml` i podaj wartość `library.a` (bibliotekę statyczną) w elemencie `nativeLibrary` analogicznie do poniższego przykładu pliku `extension.xml`.

```
<extension xmlns="http://ns.adobe.com/air/extension/3.1">
  <id>com.cnative.extensions</id>
  <versionNumber>1</versionNumber>
  <platforms>
    <platform name="iPhone-x86">
      <applicationDeployment>
        <nativeLibrary>library.a</nativeLibrary>
        <initializer>TestNativeExtensionsInitializer </initializer>
        <finalizer>TestNativeExtensionsFinalizer </finalizer>
      </applicationDeployment>
    </platform>
  </platforms>
</extension>
```

Uwaga: W przypadku testowania rozszerzenia natywnego w symulatorze systemu iOS nie należy używać biblioteki statycznej (pliku z rozszerzeniem) skompilowanej dla urządzenia. Zamiast tego należy użyć biblioteki statycznej skompilowanej dla symulatora.

Instrukcje śledzenia

W przypadku uruchamiania aplikacji dla urządzeń przenośnych na komputerze stacjonarnym wyniki działania funkcji trace (śledzenia) są wysyłane do debugera lub do okna terminala użytego do uruchomienia narzędzia ADL. Gdy aplikacja jest uruchamiana na urządzeniu lub w emulatorze, można skonfigurować sesję zdalnego debugowania w celu wyświetlenia wyników działania funkcji śledzenia. Jeśli ta funkcja jest obsługiwana, wyniki działania funkcji śledzenia można również wyświetlać za pomocą narzędzi do programowania dostarczanych przez producenta urządzenia lub systemu operacyjnego.

We wszystkich przypadkach pliki SWF aplikacji muszą zostać skompilowane przy włączonym debugowaniu, aby środowisko wykonawcze mogło wysłać wyniki instrukcji śledzenia.

Zdalne instrukcje śledzenia w systemie Android

W przypadku uruchamiania na urządzeniu z systemem Android lub w emulatorze takiego urządzenia można wyświetlać wyniki instrukcji trace (śledzenia) w dzienniku systemu Android za pomocą narzędzia Android Debug Bridge (ADB) wchodzącego w skład zestawu SDK. Aby wyświetlić wyniki działania aplikacji, należy wykonać poniższe polecenie, korzystając z wiersza poleceń lub z okna terminala na komputerze, na którym opracowano tę aplikację.

```
tools/adb logcat air.MyApp:I *:S
```

Wartość *MyApp* jest identyfikatorem danej aplikacji AIR. Argument **:S* zatrzymuje uzyskiwanie wyników z wszystkich innych procesów. Aby oprócz wyników funkcji śledzenia wyświetlić informacje systemowe o aplikacji, w specyfikacji filtra logcat można umieścić polecenie `ActivityManager`.

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

W tych przykładach poleceń założono, że narzędzie ADB jest uruchamiane w folderze zestawu SDK systemu Android lub folder `SDK` został dodany do zmiennej środowiskowej ścieżek.

Uwaga: W środowisku AIR 2.6 (lub nowszym) narzędzie ADB jest częścią zestawu SDK środowiska AIR i można je znaleźć w folderze `lib/android/bin`.

Zdalne instrukcje śledzenia w systemie iOS

Aby wyświetlić wyniki działania funkcji trace (śledzenia) aplikacji działającej w systemie iOS, należy ustawić zdalną sesję debugowania za pomocą programu Flash Debugger (FDB).

Więcej tematów Pomocy

[Android Debug Bridge: Włączanie logowania logcat](#)

„Zmienne środowiskowe ścieżek” na stronie 320

Nawiązywanie połączenia z debugerem programu Flash

Aby przeprowadzić debugowanie aplikacji działającej na urządzeniu przenośnym, należy uruchomić debuger programu Flash na komputerze używanym do debugowania i podłączyć go do sieci. Aby włączyć zdalne debugowanie, należy wykonać następujące czynności:

- W systemie Android określ w deskrytorze aplikacji uprawnienie `android.permission.INTERNET`.
- Skompiluj pliki SWF aplikacji przy włączonym debugowaniu.
- Aplikację można spakować przy użyciu opcji `-target apk-debug` (dla systemu Android) lub `-target ipa-debug` (dla systemu iOS) i flagi `-connect` (debugowanie przez sieć Wi-Fi) lub `-listen` (debugowanie przez połączenie USB).

W przypadku zdalnego debugowania przez sieć Wi-Fi urządzenie musi być w stanie uzyskać dostęp do portu TCP 7935 komputera z działającym debugerem programu Flash na podstawie adresu IP lub w pełni kwalifikowanej nazwy domeny. W przypadku zdalnego debugowania przez połączenie USB urządzenie musi być w stanie uzyskać dostęp do portu TCP 7936 lub portu określonego we flagie `-listen`.

W przypadku systemu iOS należy też określić opcję `-target ipa-debug-interpreter` lub `-target ipa-debug-interpreter-simulator`.

Zdalne debugowanie za pomocą programu Flash Professional

Gdy aplikacja będzie gotowa do debugowania, a uprawnienia zostaną ustawione w deskrypcji aplikacji, należy wykonać następujące czynności:

- 1 Otwórz okno dialogowe Settings (Ustawienia) w środowisku AIR for Android.
- 2 Na karcie Deployment (Wdrażanie):
 - Dla typu wdrażania wybierz opcję Device debugging (Debugowanie urządzenia).
 - W sekcji After publishing (Po publikacji) zaznacz opcję Install application on the connected Android device (Zainstaluj aplikację na podłączonym urządzeniu z systemem Android).
 - W sekcji After publishing (Po publikacji) usuń zaznaczenie opcji Launch application on the connected Android device (Uruchom aplikację na podłączonym urządzeniu z systemem Android).
 - W razie konieczności ustaw ścieżkę do zestawu SDK systemu Android.
- 3 Kliknij przycisk Publikuj.
Aplikacja zostanie zainstalowana i uruchomiona na urządzeniu.
- 4 Zamknij okno dialogowe Settings (Ustawienia) środowiska AIR for Android.
- 5 W menu programu Flash Professional wybierz opcje Debuguj > Rozpocznij zdalną sesję debugowania > ActionScript 3.
Na panelu Wyjście programu Flash Professional zostanie wyświetlony komunikat „Oczekiwanie na połączenie odtwarzacza”.
- 6 Uruchom aplikację na urządzeniu.
- 7 W oknie dialogowym połączenia środowiska Adobe AIR wpisz adres IP lub nazwę hosta komputera, na którym jest uruchomiony debuger Flash, a następnie kliknij przycisk OK.

Zdalne debugowanie za pomocą programu FDB i połączenia sieciowego

Aby debugować aplikację działającą na urządzeniu z programem Flash Debugger (FDB) uruchomionym w wierszu polecenia, należy najpierw uruchomić debugera na komputerze używanym do programowania, a następnie uruchomić aplikację na urządzeniu. W poniższej procedurze do kompilowania, pakowania i debugowania aplikacji na urządzeniu są używane narzędzia AMXMLC, FDB i ADT. W przykładach założono, że jest używane połączenie programu Flex i zestawu SDK środowiska AIR, a katalog bin jest uwzględniony w zmiennej środowiskowej ścieżek. (To założenie ma na celu wyłącznie uproszczenie przykładowych poleceń).

- 1 Otwórz terminal lub okno wiersza polecenia i przejdź do katalogu zawierającego kod źródłowy aplikacji.
- 2 Skompiluj aplikację przy użyciu narzędzia amxmlc z włączonym debugowaniem.

```
amxmlc -debug DebugExample.as
```
- 3 Spakuj aplikację, stosując wartość typu docelowego `apk-debug` lub `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
```

Jeśli do debugowania jest zawsze używana ta sama nazwa hosta lub ten sam adres IP, można podać tę wartość po flagdze `-connect`. Aplikacja automatycznie spróbuje nawiązać połączenie z tym adresem IP lub daną nazwą hosta. W przeciwnym razie podczas każdego rozpoczynania debugowania będzie trzeba wpisywać te informacje na urządzeniu.

4 Zainstaluj aplikację.

W systemie Android można użyć polecenia `-installApp` narzędzia ADT.

```
adt -installApp -platform android -package DebugExample.apk
```

W systemie iOS aplikację można zainstalować przy użyciu polecenia `-installApp` narzędzia ADT lub programu iTunes.

5 Otwórz program FDB w drugim terminalu lub oknie wiersza poleceń.

```
fdb
```

6 W oknie narzędzia FDB wpisz polecenie 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 Uruchom aplikację na urządzeniu.

8 Po uruchomieniu aplikacji na urządzeniu lub w emulatorze zostanie otwarte okno dialogowe połączenia ze środowiskiem Adobe AIR. (Jeśli podczas pakowania aplikacji określono nazwę hosta lub adres IP za pomocą opcji `-connect`, nastąpi próba automatycznego połączenia z tym adresem). Wpisz odpowiedni adres i dotknij przycisku OK.

Aby w tym trybie było możliwe nawiązanie połączenia z debugerem, urządzenie musi być w stanie rozpoznać adres lub nazwę hosta i łączyć się z portem TCP 7935. Jest wymagane połączenie sieciowe.

9 Po nawiązaniu połączenia między zdalnym środowiskiem wykonawczym i debugerem można ustawić punkty przerwania przy użyciu polecenia `break` narzędzia FDB i rozpocząć wykonywanie za pomocą polecenia `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
```

Zdalne debugowanie za pomocą programu FDB i połączenia USB

AIR 2.6 (Android), AIR 3.3 (iOS)

Aby debugować aplikację za pośrednictwem połączenia USB, należy spakować aplikację za pomocą opcji `-listen`, a nie opcji `-connect`. Gdy zostanie określona opcja `-listen`, po uruchomieniu aplikacji środowisko wykonawcze wykryje połączenie z narzędziem FDB na porcie TCP 7936. Następnie należy uruchomić narzędzie FDB z opcją `-p`. Narzędzie FDB zainicjuje połączenie.

Procedura debugowania przez połączenie USB dla systemu Android

Aby debugger Flash działający na komputerze stacjonarnym połączył się ze środowiskiem wykonawczym AIR działającym na urządzeniu lub w emulatorze, należy za pomocą narzędzia Android Debug Bridge (ADB) wchodzącego w skład zestawu SDK systemu Android lub iOS Debug Bridge (IDB) wchodzącego w skład zestawu SDK środowiska AIR przekazać sygnał z portu urządzenia do portu komputera.

1 Otwórz terminal lub okno wiersza poleceń i przejdź do katalogu zawierającego kod źródłowy aplikacji.

2 Skompiluj aplikację przy użyciu narzędzia `amxmlc` z włączonym debugowaniem.

```
amxmlc -debug DebugExample.as
```

3 Aplikację należy spakować przy użyciu odpowiedniego docelowego formatu debugowania (takiego jak `apk-debug`), określając też opcję `-listen`:

```
adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12  
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

4 Podłącz urządzenie za pomocą kabla USB do komputera używanego do debugowania. (Używając tej procedury, można również debugować aplikację działającą w emulatorze. Wówczas połączenie USB nie jest konieczne, ani nawet możliwe).

5 Zainstaluj aplikację.

W tym celu można użyć polecenia `-installApp` narzędzia ADT.

```
adt -installApp -platform android -package DebugExample.apk
```

6 Przekaż sygnał z portu TCP 7936 z urządzenia lub z emulatora do komputera stacjonarnego za pomocą narzędzia ADB systemu Android.

```
adb forward tcp:7936 tcp:7936
```

7 Uruchom aplikację na urządzeniu.

8 W terminalu lub w oknie poleceń uruchom program FDB, korzystając z opcji `-p`.

```
fdb -p 7936
```

9 W oknie narzędzia FDB wpisz polecenie `run`.

```
Adobe fdb (Flash Player Debugger) [build 14159]  
Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.  
(fdb) run
```

10 Program FDB spróbuje połączyć się z aplikacją.

11 Po nawiązaniu zdalnego połączenia można ustawiać punkty przerywania za pomocą polecenia `break` programu FDB. W celu rozpoczęcia wykonywania należy użyć polecenia `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
```

Uwaga: Port o numerze 7936 jest używany jako domyślny port podczas debugowania za pośrednictwem połączenia USB zarówno przez środowisko wykonawcze AIR, jak i przez program FDB. Można określić inne porty, korzystając z parametru portu `-listen` narzędzia ADT i parametru portu `-p` programu FDB. W takim przypadku należy użyć narzędzia Android Debug Bridge do przekazania numeru portu określonego w narzędziu ADT do portu określonego w programie FDB, stosując następujące polecenia: `adb forward tcp:port_nasluchwania_adt# tcp:port_fdb#`.

Procedura debugowania przez połączenie USB dla systemu iOS

Aby debugger Flash działający na komputerze stacjonarnym połączył się ze środowiskiem wykonawczym AIR działającym na urządzeniu lub w emulatorze, należy za pomocą narzędzia iOS Debug Bridge (IDB) przekazać sygnał z portu urządzenia do portu komputera.

- 1 Otwórz terminal lub okno wiersza poleceń i przejdź do katalogu zawierającego kod źródłowy aplikacji.
- 2 Skompiluj aplikację przy użyciu narzędzia `amxmlc` z włączonym debugowaniem.

```
amxmlc -debug DebugExample.as
```

- 3 Aplikację należy spakować przy użyciu odpowiedniego docelowego formatu debugowania (takiego jak `ipa-debug` lub `ipa-debug-interpreter`), określając też opcję `-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 Podłącz urządzenie za pomocą kabla USB do komputera używanego do debugowania. (Używając tej procedury, można również debugować aplikację działającą w emulatorze. Wówczas połączenie USB nie jest konieczne, ani nawet możliwe).
- 5 Zainstaluj i uruchom aplikację na urządzeniu z systemem iOS. W środowisku AIR 3.4 lub nowszym można zainstalować aplikację przez połączenie USB, używając opcji `adt -installApp`.
- 6 Określ uchwyt urządzenia za pomocą polecenia `idb -devices`. (Narzędzie IDB znajduje się w katalogu `air_sdk_root/lib/aot/bin/iOSBin/idb`).

```
./idb -devices
                                     List of attached devices
Handle    UUID
      1    91770d8381d12644df91fbcee1c5bbdacb735500
```

Uwaga: (Środowisko AIR 3.4 i nowsze wersje) Używając opcji `adt -devices` zamiast opcji `idb -devices`, można określić uchwyt urządzenia.

- 7 Przekaż port na komputerze do portu określonego w parametrze `adt -listen` (w tym przypadku 16 000, domyślnie 7936) za pomocą narzędzia IDB i identyfikatora urządzenia uzyskanego w poprzednim kroku.

```
idb -forward 7936 16000 1
```

W tym przykładzie port komputera ma numer 7936, port, na którym nasłuchuje podłączone urządzenie, ma numer 16 000, a identyfikator tego urządzenia ma wartość 1.

- 8 W terminalu lub w oknie poleceń uruchom program FDB, korzystając z opcji -p.

```
fdb -p 7936
```

- 9 W oknie narzędzia FDB wpisz polecenie run.

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                     Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                     (fdb) run
```

- 10 Program FDB spróbuje połączyć się z aplikacją.

- 11 Po nawiązaniu zdalnego połączenia można ustawiać punkty przerywania za pomocą polecenia break programu FDB. W celu rozpoczęcia wykonywania należy użyć polecenia continue.

Uwaga: Port o numerze 7936 jest używany jako domyślny port podczas debugowania za pośrednictwem połączenia USB zarówno przez środowisko wykonawcze AIR, jak i przez program FDB. Można określić inne porty, korzystając z parametru portu -listen narzędzia IDB i parametru portu -p programu FDB.

Instalowanie środowiska AIR i aplikacji AIR na urządzeniach przenośnych

Użytkownicy końcowi aplikacji mogą instalować środowisko wykonawcze AIR i aplikacje AIR, korzystając z normalnych mechanizmów dotyczących aplikacji i rozpowszechniania na swoich urządzeniach.

Na przykład w systemie Android użytkownicy mogą instalować aplikacje ze sklepu Android Market. Jeśli użytkownicy zezwolili w ustawieniach aplikacji na instalowanie aplikacji z nieznanymi źródłami, mogą oni instalować aplikację, klikając łącze na stronie internetowej lub kopiując pakiet aplikacji na urządzenie i rozpakowując go. Jeśli użytkownik spróbuje zainstalować aplikację dla systemu Android, ale nie zostało jeszcze zainstalowane środowisko wykonawcze AIR, nastąpi automatyczne przekierowanie do sklepu Android Market, gdzie można zainstalować środowisko wykonawcze.

W systemie iOS istnieją dwa sposoby rozpowszechniania aplikacji do użytkowników końcowych. Podstawowym kanałem rozpowszechniania jest sklep App Store firmy Apple. Można również skorzystać z opcji szybkiego rozpowszechniania, aby umożliwić ograniczonej liczbie użytkowników instalację aplikacji bez korzystania ze sklepu App Store.

Instalowanie środowiska wykonawczego AIR i aplikacji na potrzeby programowania

Aplikacje AIR na urządzeniach przenośnych są instalowane jako pakiety natywne, można więc skorzystać z normalnych funkcji platformy w celu zainstalowania aplikacji do testowania. Jeśli narzędzie ADT jest obsługiwane, można instalować środowisko wykonawcze AIR i aplikacje AIR za pomocą poleceń tego narzędzia. Obecnie taka możliwość jest obsługiwana w systemie Android.

W systemie iOS można instalować aplikacje do testowania za pomocą programu iTunes. Aplikacje testowe muszą być podpisane za pomocą certyfikatu podpisywania kodu wystawionego przez firmę Apple konkretnie w celu opracowywania aplikacji, a także muszą być spakowane przy użyciu programistycznego profilu informacyjnego. Aplikacja AIR w systemie iOS jest samodzielnym pakietem. Nie jest używane oddzielne środowisko wykonawcze.

Instalowanie aplikacji AIR za pomocą narzędzia ADT

Podczas programowania aplikacji AIR można używać narzędzia ADT do instalowania i odinstalowywania zarówno środowiska wykonawczego, jak i aplikacji. (Te polecenia mogą być również zintegrowane ze środowiskiem programistycznym, aby nie trzeba było samodzielnie uruchamiać narzędzia ADT).

Środowisko wykonawcze AIR można zainstalować na urządzeniu lub w emulatorze za pomocą narzędzia ADT dla środowiska AIR. Należy zainstalować pakiet SDK dla danego urządzenia. Użyj polecenia `-installRuntime`.

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

Jeśli nie zostanie określony parametr `-package`, pakiet środowiska wykonawczego odpowiedni dla urządzenia lub emulatora zostanie wybrany z pakietów dostępnych w zainstalowanym zestawie SDK środowiska AIR.

Aby zainstalować aplikację AIR w systemie Android lub iOS (środowisko AIR 3.4 lub nowsze), użyj podobnego polecenia `-installApp`.

```
adt -installApp -platform android -device deviceID -package path-to-app
```

Wartość ustawiona dla argumentu `-platform` powinna być zgodna z urządzeniem, na którym jest przeprowadzana instalacja.

Uwaga: Przed ponowną instalacją należy usunąć istniejące wersje środowiska wykonawczego AIR lub aplikacji AIR.

Instalowanie aplikacji AIR na urządzeniach z systemem iOS przy użyciu programu iTunes

Aby zainstalować aplikację AIR na urządzeniu z systemem iOS w celu testowania:

- 1 Otwórz aplikację iTunes.
- 2 Jeśli wcześniej nie wykonano tej czynności, dodaj plik informacyjny na potrzeby tej aplikacji do programu iTunes. W programie iTunes wybierz kolejno opcje Plik > Dodaj do biblioteki. Następnie wybierz plik profilu informacyjnego (dla którego jako typ pliku wybrano `mobileprovision`).
- 3 Niektóre wersje iTunes nie dopuszczają do zastępowania aplikacji w sytuacji, gdy ta sama jej wersja jest już zainstalowana. W takim przypadku należy usunąć aplikację z urządzenia oraz z listy aplikacji w iTunes.
- 4 Kliknij dwukrotnie plik IPA dla swojej aplikacji. Powinien on pojawić się na liście aplikacji w programie iTunes.
- 5 Podłącz urządzenie do portu USB komputera.
- 6 W iTunes sprawdź kartę Aplikacja dla urządzenia i upewnij się, że aplikacja została wybrana z listy aplikacji do zainstalowania.
- 7 Wybierz urządzenie z listy po lewej stronie aplikacji iTunes. Następnie kliknij przycisk Synchronizacja. Po zakończeniu synchronizacji aplikacja Hello World pojawi się na telefonie iPhone.

Jeśli nowa wersja nie została zainstalowana, usuń ją z urządzenia i z listy aplikacji w programie iTunes, a następnie wykonaj tę procedurę ponownie. Może mieć to miejsce, jeśli obecnie zainstalowana wersja ma ten sam identyfikator aplikacji i numer wersji.

Więcej tematów Pomocy

„[Polecenie `installRuntime` narzędzia ADT](#)” na stronie 188

„[Polecenie `installApp` narzędzia ADT](#)” na stronie 185

Uruchamianie aplikacji AIR na urządzeniu

Zainstalowane aplikacje AIR można uruchamiać za pomocą interfejsu użytkownika urządzenia. Jeśli ta funkcja jest obsługiwana, można również uruchamiać aplikacje zdalnie za pomocą narzędzia ADT dla środowiska AIR.

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

Wartość argumentu `-appid` musi być identyfikatorem tej aplikacji AIR, która jest uruchamiana. Należy użyć wartości określonej w deskrypcji aplikacji AIR (bez przedrostka *air.* dodanego podczas pakowania).

Jeśli podłączono i działa tylko jedno urządzenie lub jeden emulator, wówczas można pominąć flagę `-device`. Wartość ustawiona dla argumentu `-platform` powinna być zgodna z urządzeniem, na którym jest przeprowadzana instalacja. Obecnie jedyną obsługiwaną wartością jest *android*.

Usuwanie aplikacji i środowiska wykonawczego AIR

Do usuwania aplikacji można używać normalnych sposobów oferowanych przez system operacyjny urządzenia. Jeśli ta funkcja jest obsługiwana, aplikacje i środowisko wykonawcze AIR można również usuwać za pomocą narzędzia ADT dla środowiska AIR. Aby usunąć środowisko wykonawcze, należy użyć polecenia `-uninstallRuntime`.

```
adt -uninstallRuntime -platform android -device deviceID
```

Aby odinstalować aplikację, należy użyć polecenia `-uninstallApp`.

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Jeśli podłączono i działa tylko jedno urządzenie lub jeden emulator, wówczas można pominąć flagę `-device`. Wartość ustawiona dla argumentu `-platform` powinna być zgodna z urządzeniem, na którym jest przeprowadzana instalacja. Obecnie jedyną obsługiwaną wartością jest *android*.

Konfigurowanie emulatora

Aby uruchomić aplikację w emulatorze urządzenia, należy zazwyczaj skorzystać z zestawu SDK dla danego urządzenia w celu utworzenia i uruchomienia wystąpienia emulatora na komputerze używanym do programowania. Następnie można zainstalować wersję środowiska AIR oferującą emulator i zainstalować aplikację AIR w emulatorze. Aplikacje działają zazwyczaj znacznie wolniej w emulatorze niż na rzeczywistym urządzeniu.

Tworzenie emulatora systemu Android

- 1 Uruchom zestaw SDK systemu Android i aplikację AVD Manager.
 - W systemie Windows uruchom plik SDK Setup.exe znajdujący się w katalogu głównym zestawu SDK systemu Android.
 - W systemie Mac OS uruchom aplikację android znajdującą się w podkatalogu tools katalogu zestawu SDK systemu Android.
- 2 Zaznacz opcje Settings (Ustawienia) i Force https:// (Wymuś https://).
- 3 Zaznacz opcję Available Packages (Dostępne pakiety). Powinna zostać wyświetlona lista dostępnych zestawów SDK systemu Android.
- 4 Wybierz zgodny zestaw SDK systemu Android (dla systemu Android w wersji 2.3 lub nowszej) i kliknij przycisk Install Selected (Instaluj wybrane).
- 5 Zaznacz opcję Virtual Devices (Urządzenia wirtualne) i kliknij przycisk New (Nowe).
- 6 Wybierz następujące ustawienia:
 - Nazwa urządzenia wirtualnego
 - Docelowy interfejs API, taki jak Android 2.3, API level 8
 - Rozmiar karty SD (taki jak 1024)
 - Karnacja (taką jak Default HVGA)

7 Kliknij przycisk Create AVD (Utwórz urządzenie AVD).

Należy pamiętać, że w zależności od konfiguracji systemu utworzenie urządzenia wirtualnego może zająć dłuższy czas.

Teraz można uruchomić nowe urządzenie wirtualne.

- 1 Wybierz urządzenie Virtual Device (Urządzenie wirtualne) w aplikacji AVD Manager. Utworzone powyżej urządzenie wirtualne powinno być widoczne.
- 2 Wybierz opcję Virtual Device (Urządzenie wirtualne) i kliknij przycisk Start (Uruchom).
- 3 Na następnym ekranie kliknij przycisk Launch (Uruchom).

Na pulpicie powinno zostać wyświetlone okno emulatora. Może to potrwać kilka sekund. Zainicjalizowanie systemu operacyjnego Android może również potrwać dłuższy czas. W emulatorze można instalować aplikacje spakowane za pomocą wartości *apk-debug* i *apk-emulator*. Aplikacje spakowane za pomocą wartości *apk* nie działają w emulatorze.

Więcej tematów Pomocy

<http://developer.android.com/guide/developing/tools/othertools.html#android>

<http://developer.android.com/guide/developing/tools/emulator.html>

Aktualizowanie aplikacji AIR dla urządzeń przenośnych

Aplikacje AIR dla urządzeń przenośnych są rozpowszechniane jako pakiety macierzyste, dlatego korzystają ze standardowych mechanizmów aktualizacji aplikacji oferowanych przez poszczególne platformy. Zazwyczaj wymaga to przesłania aktualizacji do tego samego sklepu, w którym była rozpowszechniana oryginalna aplikacja.

Aplikacje AIR dla urządzeń przenośnych nie mogą używać platformy ani klasy Updater środowiska AIR.

Aktualizowanie aplikacji AIR w systemie Android

W przypadku aplikacji rozpowszechnianych za pośrednictwem sklepu Android Market aplikację można aktualizować, umieszczając jej nową wersję w tym sklepie, o ile są spełnione następujące warunki (narzucone przez sklep Android Market, a nie przez środowisko AIR):

- Pakiet APK jest podpisany za pomocą tego samego certyfikatu.
- Identyfikator AIR jest taki sam.
- Wartość `versionNumber` w deskrypcji aplikacji jest większa. (Należy również zwiększyć wartość `versionLabel`, jeśli jest używana).

Użytkownicy, którzy pobrali aplikację ze sklepu Android Market są powiadamiani o dostępności aktualizacji za pośrednictwem oprogramowania urządzenia.

Więcej tematów Pomocy

[Programiści aplikacji dla systemu Android: Publikowanie aktualizacji w sklepie Android Market](#)

Aktualizowanie aplikacji AIR w systemie iOS

W przypadku aplikacji AIR rozpowszechnianych za pośrednictwem sklepu aplikacji iTunes aplikację można zaktualizować, wysyłając aktualizację do sklepu, o ile są spełnione wszystkie z następujących warunków (zasad narzucanych przez sklep aplikacji firmy Apple, a nie przez środowisko AIR):

- Certyfikat podpisywania kodu i profile informacyjne są wystawione dla tego samego identyfikatora Apple.
- Pakiet IPA korzysta z tego samego identyfikatora pakietu Apple ID.
- Aktualizacja nie zmniejsza liczby obsługiwanych urządzeń. Jeśli oryginalna aplikacja obsługuje urządzenia z systemem iOS 3, wówczas nie można utworzyć aktualizacji kończącej obsługę systemu iOS 3.

***Ważne:** Zestaw SDK środowiska AIR 2.6 lub nowszego nie obsługuje systemu iOS 3, a środowisko AIR 2 go obsługuje, nie można więc aktualizować opublikowanej aplikacji dla systemu iOS opracowanej w środowisku AIR 2 za pomocą aktualizacji opracowanej w środowisku AIR 2.6+.*

Używanie powiadomień w trybie push

Powiadomienia w trybie push są powiadomieniami zdalnie wysyłanymi do aplikacji uruchomionych na urządzeniach przenośnych. Środowisko AIR 3.4 obsługuje powiadomienia w trybie push dla urządzeń z systemem iOS, używając do tego usługi APN (Apple Push Notification).

***Uwaga:** Aby włączyć powiadomienia w trybie push w środowisku AIR for Android, należy użyć rozszerzenia natywnego, na przykład opracowanego przez specjalistę ds. produktów Adobe Piotra Walczyszyna rozszerzenia [as3c2dm](#).*

Pozostała część tej sekcji określa sposób włączania powiadomień push w środowisku AIR for iOS.

***Uwaga:** W omówieniu przyjęto założenie, że użytkownik ma identyfikator programisty firmy Apple, zna mechanizmy programowania dla systemu iOS i wdrożył przynajmniej jedną aplikację dla urządzeń z systemem iOS.*

Omówienie powiadomień w trybie push

Usługa APN (Apple Push Notification) pozwala dostawcom powiadomień zdalnych na przesyłanie powiadomień do aplikacji uruchomionych na urządzeniach z systemem iOS. Usługa APN obsługuje następujące typy powiadomień:

- Komunikaty
- Ikony
- Dźwięki

Pełne informacje o usłudze APN znajdują się w witrynie developer.apple.com.

Używanie w aplikacji powiadomień w trybie push wymaga skonfigurowania kilku elementów:

- **Aplikacja kliencka** rejestruje się do powiadomień w trybie push, komunikuje się ze zdalnymi dostawcami powiadomień i odbiera powiadomienia w trybie push.
- **System iOS** zarządza interakcjami między aplikacją kliencką i usługą APN.
- **Usługa APN** zapewnia identyfikator tokenID podczas rejestracji klienta i przekazuje powiadomienia od zdalnego dostawcy powiadomień do systemu iOS.
- **Zdalny dostawca powiadomień** — przechowuje informację tokenId-client z aplikacji i przesyła powiadomienia do usługi APN.

Obieg pracy podczas rejestracji

Obieg pracy podczas rejestracji powiadomień w trybie push z usługą uruchomioną na serwerze jest następujący:

- 1 Aplikacja kliencka przesyła żądanie włączenia powiadomień w trybie push w systemie iOS.
- 2 System iOS przesyła żądanie do usługi APN.
- 3 Serwer APN zwraca wartość tokenId do systemu iOS.
- 4 System iOS zwraca wartość tokenId do aplikacji klienckiej.
- 5 Aplikacja kliencka (za pomocą mechanizmu specyficznego dla niej) przekazuje wartość tokenId do zdalnego dostawcy powiadomień, który ją zapisuje na potrzeby obsługi powiadomień w trybie push.

Obieg pracy w zakresie obsługi powiadomień

Obieg pracy w zakresie obsługi powiadomień wygląda następująco:

- 1 Zdalny dostawca powiadomień generuje powiadomienie i przesyła jego treść do usługi APN razem z wartością tokenId.
- 2 Usługa APN przekazuje powiadomienie do systemu iOS w urządzeniu.
- 3 System iOS przesyła zawartość powiadomienia do aplikacji.

Interfejs API powiadomień w trybie push

W środowisku AIR 3.4 został wprowadzony zbiór interfejsów API służących do obsługi powiadomień w trybie push w systemie iOS. Interfejsy te są dostępne w pakiecie `flash.notifications` i obejmują następujące klasy:

- `NotificationStyle` definiuje stałe wartości dla poszczególnych typów powiadomień: `ALERT`, `BADGE` i `SOUND.C`
- `RemoteNotifier` pozwala na subskrypcję powiadomień w trybie push oraz rezygnację z nich.
- `RemoteNotifierSubscribeOptions` pozwala wybrać typ powiadomień do odebrania. Za pomocą właściwości `notificationStyles` można zarejestrować wektor ciągów rejestrujących wiele typów powiadomień.

Środowisko AIR 3.4 zawiera także zdarzenie `flash.events.RemoteNotificationEvent` wywoływane przez klasę `RemoteNotifier` w następujący sposób:

- Gdy aplikacja pomyślnie zasubskrybuje powiadomienia, usługa APN tworzy nową wartość tokenId i przesyła ją do aplikacji.
- Po odebraniu nowego powiadomienia zdalnego.

Jeśli w trakcie procesu subskrypcji wystąpi błąd, klasa `RemoteNotifier` wywoła zdarzenie `flash.events.StatusEvent`.

Zarządzanie w aplikacji powiadomieniami w trybie push

Aby zarejestrować w aplikacji powiadomienia w trybie push, należy wykonać następujące kroki:

- Utwórz kod, który wykona subskrypcję powiadomień w trybie push w aplikacji.
- Włącz powiadomienia w trybie push w pliku XML aplikacji.
- Utwórz profil informacyjny oraz certyfikat umożliwiające włączenie usługi powiadomień w trybie push w systemie iOS.

Przedstawiony poniżej przykładowy kod z odpowiednimi komentarzami pozwala zasubskrybować powiadomienia w trybie push i obsługuje zdarzenia, które są z nimi związane:

```
package

{
import flash.display.Sprite;
import flash.display.StageAlign;
import flash.display.StageScaleMode;
import flash.events.*;
import flash.events.Event;
import flash.events.IOErrorEvent;
import flash.events.MouseEvent;
import flash.net.*;
import flash.text.TextField;
import flash.text.TextFormat;
import flash.ui.Multitouch;
import flash.ui.MultitouchInputMode;
// Required packages for push notifications
import flash.notifications.NotificationStyle;
import flash.notifications.RemoteNotifier;
import flash.notifications.RemoteNotifierSubscribeOptions;
import flash.events.RemoteNotificationEvent;
import flash.events.StatusEvent;
[SWF(width="1280", height="752", frameRate="60")]
public class TestPushNotifications extends Sprite
{
private var notiStyles:Vector.<String> = new Vector.<String>;;
private var tt:TextField = new TextField();
private var tf:TextFormat = new TextFormat();
// Contains the notification styles that your app wants to receive
private var preferredStyles:Vector.<String> = new Vector.<String>();
private var subscribeOptions:RemoteNotifierSubscribeOptions = new
RemoteNotifierSubscribeOptions();
private var remoteNot:RemoteNotifier = new RemoteNotifier();
private var subsButton:CustomButton = new CustomButton("Subscribe");
private var unSubsButton:CustomButton = new
CustomButton("UnSubscribe");
private var clearButton:CustomButton = new CustomButton("clearText");
private var urlreq:URLRequest;
private var urlLoad:URLLoader = new URLLoader();
private var urlString:String;
public function TestPushNotifications()
{
super();
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
tf.size = 20;
tf.bold = true;
tt.x=0;
tt.y =150;
tt.height = stage.stageHeight;
tt.width = stage.stageWidth;
tt.border = true;
tt.defaultTextFormat = tf;
addChild(tt);
subsButton.x = 150;
subsButton.y=10;
subsButton.addEventListener(MouseEvent.CLICK, subsButtonHandler);
stage.addChild(subsButton);
```

```
unSubsButton.x = 300;
unSubsButton.y=10;
unSubsButton.addEventListener(MouseEvent.CLICK,unSubsButtonHandler);
stage.addChild(unSubsButton);
clearButton.x = 450;
clearButton.y=10;
clearButton.addEventListener(MouseEvent.CLICK,clearButtonHandler);
stage.addChild(clearButton);
//
tt.text += "\n SupportedNotification Styles: " +
RemoteNotifier.supportedNotificationStyles.toString() + "\n";
tt.text += "\n Before Preferred notificationStyles: " +
subscribeOptions.notificationStyles.toString() + "\n";
// Subscribe to all three styles of push notifications:
// ALERT, BADGE, and SOUND.
preferredStyles.push(NotificationStyle.ALERT
,NotificationStyle.BADGE,NotificationStyle.SOUND );
subscribeOptions.notificationStyles= preferredStyles;
tt.text += "\n After Preferred notificationStyles:" +
subscribeOptions.notificationStyles.toString() + "\n";

remoteNot.addEventListener(RemoteNotificationEvent.TOKEN,tokenHandler);

remoteNot.addEventListener(RemoteNotificationEvent.NOTIFICATION,notificationHandler);
remoteNot.addEventListener(StatusEvent.STATUS,statusHandler);
this.stage.addEventListener(Event.ACTIVATE,activateHandler);
}
// Apple recommends that each time an app activates, it subscribe for
// push notifications.
public function activateHandler(e:Event):void{
// Before subscribing to push notifications, ensure the device supports
it.

// supportedNotificationStyles returns the types of notifications
// that the OS platform supports
if(RemoteNotifier.supportedNotificationStyles.toString() != "")
{
remoteNot.subscribe(subscribeOptions);
}
else{
tt.appendText("\n Remote Notifications not supported on this Platform
!");
}
}
public function subsButtonHandler(e:MouseEvent):void{
remoteNot.subscribe(subscribeOptions);
}
// Optionally unsubscribe from push notifications at runtime.
public function unSubsButtonHandler(e:MouseEvent):void{
remoteNot.unsubscribe();
tt.text += "\n UNSUBSCRIBED";
}
public function clearButtonHandler(e:MouseEvent):void{
tt.text = " ";
}
// Receive notification payload data and use it in your app
public function notificationHandler(e:RemoteNotificationEvent):void{
tt.appendText("\nRemoteNotificationEvent type: " + e.type +
```

```
        "\nbubbles: "+ e.bubbles + "\ncancelable " +e.cancelable);
    for (var x:String in e.data) {
        tt.text += "\n"+ x + ": " + e.data[x];
    }
}
// If the subscribe() request succeeds, a RemoteNotificationEvent of
// type TOKEN is received, from which you retrieve e.tokenId,
// which you use to register with the server provider (urbanairship, in
// this example.
public function tokenHandler(e:RemoteNotificationEvent):void
{
    tt.appendText("\nRemoteNotificationEvent type: "+e.type +"\nBubbles:
"+ e.bubbles + "\ncancelable " +e.cancelable +"\ntokenID:\n"+ e.tokenId +"\n");
    urlString = new
String("https://go.urbanairship.com/api/device_tokens/" +
    e.tokenId);
    urlreq = new URLRequest(urlString);
    urlreq.authenticate = true;
    urlreq.method = URLRequestMethod.PUT;
    URLRequestDefaults.setLoginCredentialsForHost
("go.urbanairship.com",
"1ssB2iV_RL6_UBLiYMQVfg", "t-kZlzXGQ6-yU8T3iHiSyQ");
    urlLoad.load(urlreq);
    urlLoad.addEventListener(IOErrorEvent.IO_ERROR,iohandler);
    urlLoad.addEventListener(Event.COMPLETE,compHandler);
    urlLoad.addEventListener(HTTPStatusEvent.HTTP_STATUS,httpHandler);
}
private function iohandler(e:IOErrorEvent):void
{
    tt.appendText("\n In IOError handler" + e.errorID + " " +e.type);
}
private function compHandler(e:Event):void{
    tt.appendText("\n In Complete handler,"+"status: " +e.type + "\n");
}
private function httpHandler(e:HTTPStatusEvent):void{
    tt.appendText("\n in httpstatus handler,"+ "Status: " + e.status);
}
// If the subscription request fails, StatusEvent is dispatched with
// error level and code.
public function statusHandler(e:StatusEvent):void{
    tt.appendText("\n statusHandler");
    tt.appendText("event Level" + e.level +"\nevent code " +
e.code + "\ne.currentTarget: " + e.currentTarget.toString());
}
}
}
```

Włączanie powiadomień w trybie push w pliku XML aplikacji

Aby użyć w aplikacji powiadomień w trybie push, należy w znaczniku `Entitlements` wprowadzić następujące informacje (w znaczniku `iphone`).


```
<iphone>
    ...
    <Entitlements>
    <![CDATA[
        <key>aps-environment</key>
        <string>development</string>
    ]]>
    </Entitlements>
</iphone>
```

Gdy aplikacja jest już gotowa do przesłania do sklepu App Store, należy zmienić zawartość ciągu `<string>` z „development” na „production”.

```
<string>production</string>
```

Jeśli aplikacja obsługuje zlokalizowane ciągi, należy podać języki w znaczniku `supportedLanguages`, pod znacznikiem `initialWindow`, tak jak w poniższym przykładzie.

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

Tworzenie profilu informacyjnego oraz certyfikatu umożliwiającego włączenie usługi obsługującej powiadomienia w trybie push w systemie iOS

Aby umożliwić komunikację między aplikacją i usługą APN, należy spakować aplikację razem z profilem i certyfikatem informacji, zgodnie z poniższym opisem. Pozwoli to na włączenie usługi powiadomień w trybie push w systemie iOS.

- 1 Zaloguj się na konto programisty firmy Apple.
- 2 Przejdź do witryny Provisioning Portal (Portal informacji).
- 3 Kliknij zakładkę App IDs (Identyfikatory aplikacji).
- 4 Kliknij przycisk New App ID (Nowy identyfikator aplikacji).
- 5 Wprowadź opis i identyfikator paczki pakietu. (W identyfikatorze nie należy używać znaku *).
- 6 Kliknij przycisk Submit (Wyślij). Witryna Provisioning Portal (Portal informacji) wygeneruje identyfikator App ID i wyświetli ponownie stronę identyfikatorów.
- 7 Kliknij przycisk Configure (Konfiguruj) umieszczony obok otrzymanego identyfikatora App ID. Pojawi się strona Configure App ID (Konfiguracja identyfikatora aplikacji)
- 8 Zaznacz pole wyboru Enable for Apple Push Notification service (Włącz w powiadomieniach Apple w trybie push) Uwaga: Dostępne są dwa rodzaje certyfikatów SSL do obsługi powiadomień w trybie push — jeden do programowania i testowania oraz drugi do użytku produkcyjnego.
- 9 Kliknij przycisk Configure (Konfiguruj) umieszczony obok nazwy Development Push SSL Certificate (Certyfikat SSL trybu push do programowania). Zostanie wyświetlona strona generowania żądania podpisania certyfikatu (Certificate Signing Request, CSR).
- 10 Zgodnie z informacjami podanymi na tej stronie wygeneruj żądanie CSR, używając narzędzia Dostęp do pęku kluczy.
- 11 Wygeneruj certyfikat SSL.
- 12 Pobierz i zainstaluj certyfikat SSL.
- 13 (Opcjonalnie) Powtórz kroki od 9 do 12 dla certyfikatu SSL przeznaczonego do obsługi powiadomień w trybie push w użytku produkcyjnym.
- 14 Kliknij polecenie Done (Gotowe). Pojawi się strona Configure App ID (Konfiguracja identyfikatora aplikacji)

- 15 Kliknij polecenie Done (Gotowe). Pojawi się strona App IDs (Identyfikatory aplikacji). Zwróć uwagę na zielony okrąg obok nazwy Push Notifications (Powiadomienia w trybie push) umieszczony przy otrzymanym identyfikatorze App ID.
- 16 Zapisuj otrzymywane certyfikaty SSL. Są one później używane do komunikacji między aplikacją i dostawcą.
- 17 Kliknij kartę Provisioning (Informacje). Zostanie wyświetlona strona Provisioning Profiles (Profile informacji).
- 18 Utwórz profil informacyjny dla nowego identyfikatora App ID i pobierz go.
- 19 Kliknij kartę Certificates (Certyfikaty) i pobierz nowy certyfikat dla nowego profilu informacyjnego.

Używanie dźwięku dla powiadomień w trybie push

Aby włączyć w aplikacji powiadomienia dźwiękowe, pliki dźwiękowe należy dołączyć tak samo jak inne zasoby, jednak muszą się one znajdować w tym samym katalogu co pliki SWF i app.xml. Na przykład:

```
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
```

Firma Apple obsługuje następujące formaty danych dźwiękowych (w plikach AIFF, WAV lub CAF):

- Linear PCM
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Używanie zlokalizowanych powiadomień z komunikatami

Aby użyć w aplikacji zlokalizowanych powiadomień z komunikatami, zlokalizowane ciągi należy dołączyć w postaci folderów lproj. Aby na przykład wprowadzić obsługę komunikatów w języku hiszpańskim, należy wykonać następujące czynności:

- 1 Utwórz w projekcie folder es.lproj na tym samym poziomie, na którym znajduje się plik app.xml.
- 2 W folderze es.lproj utwórz plik tekstowy o nazwie Localizable.Strings.
- 3 Otwórz plik Localizable.Strings w edytorze tekstowym i wpisz w nim klucze komunikatów oraz odpowiednie zlokalizowane ciągi. Na przykład:

```
"PokeMessageFormat" = "La notificación de alertas en español."
```
- 4 Zapisz plik.
- 5 Gdy aplikacja otrzyma powiadomienie z komunikatem o danej wartości klucza, a językiem urządzenia jest hiszpański, pojawi się zlokalizowany tekst komunikatu.

Konfigurowanie zdalnego dostawcy powiadomień

Do przesyłania do aplikacji powiadomień w trybie push potrzebny jest zdalny dostawca powiadomień. Aplikacja serwera działa jako dostawca, odbierając informacje przesyłane w trybie push oraz przekazując powiadomienia i ich dane do usługi APN. Ta usługa przesyła do aplikacji klienckiej powiadomienie w trybie push.

Szczegółowe informacje o powiadomieniach w trybie push od zdalnego dostawcy powiadomień znajdują się w dokumencie [Komunikacja dostawcy z usługą powiadomień w trybie push firmy Apple](#) w bibliotece dokumentacji dla programistów firmy Apple.

Opcje obsługi zdalnego dostawcy powiadomień

Dostępne są następujące opcje obsługi zdalnego dostawcy powiadomień:

- Można utworzyć własnego dostawcę opartego na serwerze open source APNS-php. Serwer PHP można skonfigurować, używając projektu <http://code.google.com/p/apns-php/>. Ten należący do usługi Google Code projekt pozwala na utworzenie interfejsu pasującego do określonych wymagań.
- Można użyć dostawcy usług. Na przykład witryna <http://urbanairship.com/> oferuje gotowego dostawcę APN. Po zarejestrowaniu usługi można ją uruchomić, podając token urządzenia za pomocą kodu podobnego do poniższego.

```
private var urlreq:URLRequest;

private var urlLoad:URLLoader = new URLLoader();
private var urlString:String;
//When subscription is successful then only call the
following code
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);
}
```

Następnie można przetestować obsługę powiadomień, używając narzędzi Urban Airship.

Certyfikaty zdalnego dostawcy powiadomień

Wygenerowany wcześniej certyfikat SSL i klucz prywatny należy skopiować do odpowiedniego miejsca na serwerze zdalnego dostawcy powiadomień. Te dwa pliki można przeważnie połączyć w jednym pliku PEM. Aby to zrobić, wykonaj następujące czynności:

1 Otwórz okno terminalu.

2 Utwórz plik PEM z certyfikatu SSL, wpisując następujące polecenie:

```
openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
```

3 Utwórz plik PEM z pliku klucza prywatnego (.p12), wpisując następujące polecenie:

```
openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
```

4 Połącz dwa uzyskane pliki PEM w jeden, wpisując następujące polecenie:

```
cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
```

5 Podaj utworzony plik PEM podczas tworzenia aplikacji obsługującej powiadomienia w trybie push po stronie serwera.

Więcej informacji znajduje się w artykule [Instalowanie certyfikatu i klucza SSL na serwerze](#) dostępnym w dokumencie Podręcznik firmy Apple: Programowanie powiadomień lokalnych i w trybie push.

Obsługa w aplikacji powiadomień w trybie push

Obsługa w aplikacji powiadomień w trybie push obejmuje następujące czynności:

- Globalna konfiguracja użytkownika i przyjmowanie powiadomień w trybie push
- Przyjmowanie przez użytkownika poszczególnych powiadomień w trybie push
- Obsługa powiadomień w trybie push oraz danych, które są w nich przesyłane

Konfigurowanie i przyjmowanie powiadomień w trybie push

Gdy użytkownik pierwszy raz uruchomi aplikację obsługującą powiadomienia w trybie push, system iOS wyświetli komunikat „*appname*” chce wysłać Ci powiadomienia Push. Dostępne są wówczas dwa przyciski: Nie pozwalaj i OK. Jeśli użytkownik wybierze przycisk OK, aplikacja będzie mogła odbierać wszystkie rodzaje powiadomień, które subskrybuje. Jeśli użytkownik wybierze polecenie Nie pozwalaj, nie będą przesyłane żadne powiadomienia.

Uwaga: Rodzaje powiadomień, które aplikacja może otrzymywać, można także skonfigurować, wybierając polecenie Ustawienia > Powiadomienia.

Firma Apple zaleca, aby aplikacja zawsze subskrybowała powiadomienia podczas uaktywniania. Wywołanie przez aplikację metody `RemoteNotifier.subscribe()` powoduje otrzymanie parametru `RemoteNotificationEvent` typu `token` z 32-bajtową niepowtarzalną wartością liczbową `tokenId`, która pozwala na jednoznaczny identyfikację aplikacji na danym urządzeniu.

Gdy urządzenie odbierze powiadomienie w trybie push, wyświetli okno podręczne z przyciskami Zamknij i Uruchom. Gdy użytkownik dotknie przycisku Zamknij, system iOS wywoła aplikację, która odbierze zdarzenie `flash.events.RemoteNotificationEvent` opisanego poniżej typu `notification`.

Obsługa powiadomień w trybie push oraz danych, które są w nich przesyłane

Gdy zdalny dostawca powiadomień przesyła powiadomienie do urządzenia (za pomocą wartości `tokenID`), aplikacja odbiera zdarzenie `flash.events.RemoteNotificationEvent` typu `notification` niezależnie od tego, czy jest uruchomiona. Następnie wykonywane są wewnętrzne procesy aplikacji związane z obsługą powiadomienia. Jeśli aplikacja obsługuje dane powiadomień, można uzyskać do nich dostęp za pomocą właściwości `RemoteNotificationEvent.data` w formacie JSON.

Rozdział 8: Programowanie aplikacji AIR dla urządzeń telewizyjnych

Funkcje środowiska AIR dotyczące obsługi telewizorów

Aplikacje Adobe® AIR® można tworzyć dla urządzeń telewizyjnych, takich jak telewizory, cyfrowe kamery wideo i odtwarzacze Blu-ray, jeśli na urządzeniach docelowych jest dostępne środowisko Adobe AIR. Środowisko AIR dla urządzeń telewizyjnych jest zoptymalizowane w celu używania między innymi funkcji przyspieszenia sprzętowego urządzeń, co pozwala uzyskać wysoką jakość wideo i grafiki.

Aplikacje AIR dla urządzeń telewizyjnych są zachowywane w plikach SWF (nie HTML). Aplikacja AIR dla urządzeń telewizyjnych może korzystać z przyspieszania sprzętowego i innych funkcji środowiska AIR dedykowanych do zastosowań domowych.

Profile urządzeń

Profile używane w środowisku AIR umożliwiają zdefiniowanie docelowego zestawu urządzeń o podobnych funkcjach. W przypadku aplikacji AIR dla urządzeń telewizyjnych należy korzystać z następujących profili:

- Profil `tv`. Tego profilu należy używać w aplikacjach AIR przeznaczonych na urządzenia telewizyjne ze środowiskiem AIR.
- Profil `extendedTV`. Tego profilu należy używać, jeśli aplikacja AIR dla urządzeń telewizyjnych korzysta z rozszerzeń natywnych.

Funkcje języka ActionScript definiowane przez te profile opisano w temacie „[Profil urządzeń](#)” na stronie 259. Różnice specyficzne dla funkcji języka ActionScript w aplikacjach AIR dla urządzeń telewizyjnych opisano w dokumencie [Dokumentacja języka ActionScript 3.0 dla platformy Adobe Flash](#).

Szczegółowe informacje o profilach środowiska AIR dla urządzeń telewizyjnych można znaleźć w sekcji „[Obsługiwane profile](#)” na stronie 151.

Przyspieszanie sprzętowe

Urządzenia telewizyjne są wyposażone w akceleratory sprzętowe, które znacznie zwiększają wydajność grafiki i wideo w aplikacji AIR. Więcej informacji na temat korzystania z tych akceleratorów zawiera sekcja „[Uwagi dotyczące projektowania aplikacji AIR dla urządzeń telewizyjnych](#)” na stronie 131.

Ochrona zawartości

Środowisko AIR dla urządzeń telewizyjnych pozwala tworzyć rozbudowane, kompleksowe produkty związane z materiałami wideo o wysokiej jakości, od hollywoodzkich hitów przez filmy niezależne po seriale telewizyjne. Dostawcy takich materiałów mogą tworzyć aplikacje interaktywne za pomocą narzędzi firmy Adobe. Mogą oni włączać do swojej infrastruktury rozpowszechniania materiałów produkty serwerowe firmy Adobe albo pracować z jednym z partnerów firmy Adobe.

Jednym z głównych wymagań związanych z rozpowszechnianiem materiałów wideo o wysokiej jakości jest ochrona zawartości. Środowisko AIR dla urządzeń telewizyjnych obsługuje oprogramowanie Adobe® Flash® Access™ przeznaczone do ochrony zawartości i stosowania rozwiązań finansowych, które spełnia surowe wymagania w zakresie zabezpieczeń stawiane przez właścicieli tworzonych materiałów, w tym przez duże studia filmowe.

Oprogramowanie Flash Access obsługuje następujące funkcje:

- Pobieranie i przesyłanie strumieniowe wideo
- Różne modele biznesowe, w tym utrzymywanie z reklam, subskrypcję, wypożyczanie i sprzedaż elektroniczną
- Różne technologie dostarczania zawartości, takie jak dynamiczne przesyłanie strumieniowe HTTP, przesyłanie strumieniowe za pośrednictwem protokołu RTMP (Real Time Media Protocol) przy użyciu serwera Flash® Media Server i stopniowe pobieranie za pomocą protokołu HTTP

Środowisko AIR dla urządzeń telewizyjnych oferuje również wbudowaną obsługę protokołu RTMPE, który jest szyfrowaną wersją protokołu RTMP przeznaczoną dla istniejących rozwiązań w zakresie przesyłania strumieniowego o niższych wymaganiach bezpieczeństwa. Protokół RTMPE i powiązane z nim technologie weryfikacji plików SWF są obsługiwane w oprogramowaniu Flash Media Server.

Więcej informacji można uzyskać na stronie produktu [Adobe Flash Access](#).

Dźwięk wielokanałowy

Począwszy od wersji AIR 3, środowisko AIR dla urządzeń telewizyjnych obsługuje dźwięk wielokanałowy w materiałach wideo, które są stopniowo pobierane z serwera HTTP. Ta obsługa obejmuje następujące kodeki:

- AC-3 (Dolby Digital)
- E-AC-3 (Enhanced Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

***Uwaga:** Nie jest jeszcze dostępna obsługa dźwięku wielokanałowego w materiałach wideo przesyłanych strumieniowo z programu Adobe Flash Media Server.*

Urządzenia wejściowe do obsługi gier

Począwszy od wersji AIR 3, środowisko AIR dla urządzeń telewizyjnych obsługuje interfejsy API języka ActionScript umożliwiające aplikacjom komunikowanie się z dołączonymi urządzeniami wejściowymi do obsługi gier, takimi jak joysticki, kontrolery do gier i kontrolery ruchu. Choć są one nazywane urządzeniami wejściowymi do obsługi gier, mogą z nich korzystać nie tylko gry, lecz wszystkie aplikacje AIR dla urządzeń telewizyjnych.

Dostępny jest szeroki wybór urządzeń wejściowych do obsługi gier o różnych możliwościach. Dlatego urządzenia te są obsługiwane ogólnie w interfejsie API, tak aby aplikacja mogła poprawnie współpracować z różnymi (również z nieznanymi) typami urządzeń wejściowych do obsługi gier.

Klasa `GameInput` jest punktem wejścia interfejsów API języka ActionScript dla urządzeń wejściowych do obsługi gier. Więcej informacji zawiera opis klasy [GameInput](#).

Przyspieszone renderowanie grafiki na stole montażowym 3D

Począwszy od wersji AIR 3, środowisko AIR dla urządzeń telewizyjnych obsługuje przyspieszone renderowanie grafiki na stole montażowym 3D. Interfejsy API [Stage3D](#) języka ActionScript to zestaw interfejsów API niskiego poziomu, które korzystają z przyspieszania GPU i oferują zaawansowane funkcje 2D i 3D. Za pomocą tych interfejsów API niskiego poziomu programiści mogą elastycznie korzystać z przyspieszania sprzętowego GPU w celu znacznego zwiększenia wydajności. Można również korzystać z mechanizmów gier, które obsługują interfejsy API Stage3D języka ActionScript.

Więcej informacji można znaleźć na stronie [Mechanizmy gier, grafika 3D i stół montażowy 3D](#).

Rozszerzenia natywne

Gdy aplikacja jest przeznaczona do używania profilu `extendedTV`, może korzystać z pakietów rozszerzeń natywnych środowiska AIR (ANE, AIR Native Extension).

Zwykle producenci urządzeń udostępniają pakiety ANE zapewniające dostęp do funkcji urządzeń, które bez takich pakietów nie są obsługiwane przez środowisko AIR. Są to na przykład rozszerzenia natywne umożliwiające zmienianie kanałów na telewizorze lub wstrzymywanie odtwarzania na odtwarzaczu wideo.

W przypadku pakowania aplikacji AIR dla urządzeń telewizyjnych, która korzysta z pakietów ANE, aplikacja jest pakowana do pliku AIRN, a nie AIR.

Rozszerzenia natywne środowiska AIR dla urządzeń telewizyjnych są zawsze rozszerzeniami *dostępnyymi jako część pakietu urządzenia*. Dostępność jako część pakietu urządzenia oznacza, że biblioteki rozszerzenia są instalowane na urządzeniu telewizyjnym ze środowiskiem AIR. Pakiet ANE dołączany do pakietu aplikacji *nigdy* nie zawiera bibliotek natywnych rozszerzenia. Czasami zawiera on tylko wersję rozszerzenia natywnego opartą wyłącznie na kodzie ActionScript. Ta wersja jest symulatorem rozszerzenia opartym wyłącznie na kodzie ActionScript. Producent urządzenia instaluje na urządzeniu właściwe rozszerzenie, w tym biblioteki natywne.

Podczas opracowywania rozszerzeń natywnych należy zwrócić uwagę na poniższe kwestie:

- Opracowując rozszerzenie natywne środowiska AIR dla urządzeń telewizyjnych przeznaczone dla określonych urządzeń, zawsze należy konsultować się z ich producentem.
- W przypadku niektórych urządzeń telewizyjnych ze środowiskiem AIR tylko producent urządzenia tworzy rozszerzenia natywne.
- W przypadku wszystkich urządzeń telewizyjnych ze środowiskiem AIR producent urządzenia decyduje o tym, jakie rozszerzenia natywne są instalowane.
- Narzędzia do programowania rozszerzeń natywnych środowiska AIR dla urządzeń telewizyjnych różnią się w zależności od producenta.

Więcej informacji o korzystaniu z rozszerzeń natywnych w aplikacji AIR można znaleźć w sekcji „[Korzystanie z rozszerzeń natywnych dla środowiska Adobe AIR](#)” na stronie 159.

Informacje na temat opracowywania rozszerzeń natywnych zawiera strona [Programowanie rozszerzeń natywnych dla środowiska Adobe AIR](#).

Uwagi dotyczące projektowania aplikacji AIR dla urządzeń telewizyjnych

Uwagi dotyczące wideo

Wytyczne dotyczące kodowania wideo

W przypadku przesyłania strumieniowego wideo na urządzenie telewizyjne firma Adobe zaleca przestrzeganie następujących wytycznych w zakresie kodowania:

Kodek wideo:	H.264, profil Main (Główny) lub High (Wysoka jakość), kodowanie progresywne
Rozdzielczość:	720i, 720p, 1080i lub 1080p
Szybkość odtwarzania:	24 klatki na sekundę lub 30 klatek na sekundę
Kodek audio:	AAC-LC lub AC-3 (44,1 kHz, stereo) albo następujące kodeki dźwięku wielokanałowego: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio lub DTS-HD Master Audio
Łączna szybkość odtwarzania:	Do 8 Mb/s (w zależności od dostępnej przepustowości)
Szybkość odtwarzania audio:	Do 192 Kb/s
Proporcje pikseli:	1 × 1

Firma Adobe zaleca używanie kodeka H.264 dla materiałów wideo przeznaczonych na urządzenia telewizyjne ze środowiskiem AIR.

Uwaga: Środowisko AIR dla urządzeń telewizyjnych obsługuje również materiały wideo zakodowane za pomocą kodeków Sorenson Spark i On2 VP6. Sprzęt nie obsługuje jednak dekodowania ani renderowania tych kodeków. Zamiast tego środowisko wykonawcze stosuje dla tych kodeków dekodowanie i renderowanie programowe, więc materiały wideo są odtwarzane znacznie wolniej. Dlatego należy używać kodeka H.264, jeśli tylko jest to możliwe.

Klasa StageVideo

Środowisko AIR dla urządzeń telewizyjnych obsługuje renderowanie i dekodowanie sprzętowe zawartości wideo zakodowanej w formacie H.264. Do używania tej funkcji służy klasa StageVideo.

Sekcja [Używanie klasy StageVideo do renderowania z przyspieszaniem sprzętowym](#) w dokumencie *ActionScript 3.0 — Podręcznik programistów* zawiera szczegółowe informacje na następujące tematy:

- Interfejs API klasy StageVideo i klas z nią związanych
- Ograniczenia stosowania klasy StageVideo

Aby zapewnić najlepszą obsługę istniejącym aplikacjom AIR używającym obiektu Video w przypadku wideo zakodowanego w formacie H.264, środowisko AIR dla urządzeń telewizyjnych *wewnętrznie* korzysta z obiektu StageVideo. Oznacza to, że podczas odtwarzania wideo jest stosowane renderowanie i dekodowanie sprzętowe. Obiekt Video podlega jednak tym samym ograniczeniom co obiekt StageVideo. Jeśli na przykład aplikacja próbuje obrócić obraz wideo, obraz nie jest obracany, ponieważ zawartość wideo jest renderowana sprzętowo, a nie przez środowisko wykonawcze.

Pisząc nowe aplikacje, w przypadku wideo zakodowanego w formacie H.264 należy jednak używać obiektu StageVideo.

Przykład stosowania klasy StageVideo można znaleźć w temacie [Dostarczanie wideo i materiałów dla platformy Flash na urządzeniach telewizyjnych](#).

Wytyczne dotyczące dostarczania wideo

Na urządzeniu telewizyjnym ze środowiskiem AIR podczas odtwarzania wideo może zmieniać się dostępna przepustowość sieci. Zmiany mogą występować na przykład wtedy, gdy z tego samego połączenia internetowego zaczyna korzystać inny użytkownik.

Dlatego firma Adobe zaleca, aby system dostarczania wideo korzystał z funkcji adaptacyjnej szybkości odtwarzania. Po stronie serwera adaptacyjną szybkość odtwarzania obsługuje na przykład oprogramowanie Flash Media Server. Po stronie klienta można używać platformy OSMF (Open Source Media Framework).

Zawartość wideo można dostarczać przez sieć do aplikacji AIR dla urządzeń telewizyjnych za pośrednictwem następujących protokołów:

- Dynamiczne przesyłanie strumieniowe HTTP i HTTPS (format F4F)
- Przesyłanie strumieniowe RTMP, RTMPE, RTMFP, RTMPT i RTMPTE
- Stopniowe pobieranie HTTP i HTTPS

Więcej informacji:

- [Podręcznik programisty produktu Adobe Flash Media Server](#)
- [Platforma Open Source Media Framework](#)

Uwagi dotyczące dźwięków

Kod ActionScript przeznaczony do odtwarzania dźwięku w aplikacjach AIR dla urządzeń telewizyjnych jest taki sam jak w innych aplikacjach AIR. Więcej informacji zawiera sekcja [Praca z dźwiękiem](#) w dokumencie *ActionScript 3.0 — Podręcznik programistów*.

W przypadku obsługi dźwięku wielokanałowego w środowisku AIR dla urządzeń telewizyjnych należy wziąć pod uwagę następujące kwestie:

- Środowisko AIR dla urządzeń telewizyjnych obsługuje dźwięk wielokanałowy w materiałach wideo, które są stopniowo pobierane z serwera HTTP. Nie jest jeszcze dostępna obsługa dźwięku wielokanałowego w materiałach wideo przesyłanych strumieniowo z programu Adobe Flash Media Server.
- Środowisko AIR dla urządzeń telewizyjnych obsługuje wiele kodeków audio, lecz nie wszystkie *urządzenia* telewizyjne ze środowiskiem AIR obsługują cały ten zestaw. Klasa `flash.system.Capabilities` zawiera metodę `hasMultiChannelAudio()`, za pomocą której należy sprawdzić, czy dane urządzenie telewizyjne ze środowiskiem AIR obsługuje określony kodek dźwięku wielokanałowego, taki jak AC-3.

Założmy na przykład, że aplikacja stopniowo pobiera plik wideo z serwera. Na serwerze znajdują się różne pliki wideo H.264, które obsługują odmienne kodeki dźwięku wielokanałowego. Za pomocą metody `hasMultiChannelAudio()` aplikacja może ustalić, którego pliku wideo należy zażądać od serwera. Aplikacja może też wysłać do serwera ciąg znaków zawarty we właściwości `Capabilities.serverString`. Ten ciąg znaków wskazuje, które kodeki dźwięku wielokanałowego są dostępne, dzięki czemu serwer może wybrać odpowiedni plik wideo.

- Podczas używania jednego z kodeków dźwięku DTS istnieje możliwość, że metoda `hasMultiChannelAudio()` zwróci wartość `true`, chociaż dźwięk w formacie DTS nie będzie odtwarzany.

Za przykład może posłużyć odtwarzacz Blu-ray z wyjściem S/PDIF podłączony do starego wzmacniacza. Stary wzmacniacz może nie obsługiwać formatu DTS, ale wyjście S/PDIF nie oferuje protokołu do powiadamiania o tym odtwarzacza Blu-ray. Jeżeli odtwarzacz Blu-ray będzie wysyłać strumień dźwięku DTS do starego wzmacniacza, użytkownik nic nie usłyszy. Dlatego w przypadku dźwięku DTS najlepiej jest zapewnić interfejs użytkownika, za pomocą którego użytkownik może wskazać, że dźwięk nie jest odtwarzany. Pozwoli to aplikacji zmienić kodek na inny.

Poniższa tabela zawiera podsumowanie określające, kiedy należy używać poszczególnych kodeków audio w aplikacjach AIR dla urządzeń telewizyjnych. W tabeli wskazano również, kiedy są używane akceleratory sprzętowe podczas dekodowania za pomocą kodeka audio na urządzeniach telewizyjnych ze środowiskiem AIR. Dekodowanie sprzętowe zwiększa wydajność i odciąża procesor.

Kodek audio	Dostępność na urządzeniu telewizyjnym ze środowiskiem AIR	Dekodowanie sprzętowe	Kiedy używać tego kodeku audio	Więcej informacji
AAC	Zawsze	Zawsze	W plikach wideo zakodowanych w formacie H.264. W przypadku przesyłania strumieniowego dźwięków, na przykład w internetowej usłudze przesyłania strumieniowego muzyki.	Podczas przesyłania strumieniowego wyłącznie dźwięków w formacie AAC dźwięki przesyłane strumieniowo należy umieścić w kontenerze MP4.
mp3	Zawsze	Nie	Dla dźwięków w plikach SWF aplikacji. W materiałach wideo zakodowanych za pomocą kodeka Sorenson Spark lub On2 VP6.	Materiały wideo H.264 z dźwiękiem mp3 nie są odtwarzane na urządzeniach telewizyjnych ze środowiskiem AIR.

Kodek audio	Dostępność na urządzeniu telewizyjnym ze środowiskiem AIR	Dekodowane sprzętowe	Kiedy używać tego kodeku audio	Więcej informacji
AC-3 (Dolby Digital) E-AC-3 (Enhanced Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Sprawdź	Tak	W plikach wideo zakodowanych w formacie H.264.	Zazwyczaj środowisko AIR dla urządzeń telewizyjnych przekazuje strumień dźwięku wielokanałowego do zewnętrznego odbiornika audio/wideo, który dekoduje i odtwarza dźwięk.
Speex	Zawsze	Nie	Odbieranie strumienia głosowego na żywo.	Materiały wideo H.264 korzystające z kodeka dźwięku Speex nie są odtwarzane na urządzeniach telewizyjnych ze środowiskiem AIR. Kodeka Speex należy używać tylko w materiałach wideo zakodowanych przy użyciu kodeka Sorenson Spark lub On2 VP6.
NellyMoser	Zawsze	Nie	Odbieranie strumienia głosowego na żywo.	Materiały wideo H.264 korzystające z kodeka dźwięku NellyMoser nie są odtwarzane na urządzeniach telewizyjnych ze środowiskiem AIR. Kodeka NellyMoser należy używać tylko w materiałach wideo zakodowanych przy użyciu kodeka Sorenson Spark lub On2 VP6.

Uwaga: Niektóre pliki wideo zawierają dwa strumienie audio. Na przykład plik wideo może zawierać zarówno strumień AAC, jak i strumień AC3. Środowisko AIR dla urządzeń telewizyjnych nie obsługuje takich plików wideo. Użycie takiego pliku może spowodować brak dźwięku związanego z wideo.

Sprzętowe przyspieszanie grafiki

Używanie sprzętowego przyspieszania grafiki

Urządzenia telewizyjne ze środowiskiem AIR zapewniają przyspieszanie sprzętowe dla operacji na grafice 2D. Sprzętowe akceleratory graficzne urządzenia odciążają procesor w zakresie wykonywania następujących działań:

- Renderowanie bitmap
- Skalowanie bitmap
- Mieszanie bitmap
- Wypełnianie pełnych prostokątów

Takie sprzętowe przyspieszanie grafiki oznacza, że wiele operacji graficznych w aplikacji AIR dla urządzeń telewizyjnych można realizować z wysoką wydajnością. Przykładami tych operacji mogą być:

- Przejścia z przesuwaniem
- Przejścia ze skalowaniem
- Pojawianie się i znikanie obiektów

- Składanie wielu obrazów przy użyciu kanału alfa

Aby w operacjach tego typu uzyskać korzyści wynikające z używania sprzętowego przyspieszania grafiki, należy zastosować jedną z następujących technik:

- Dla właściwości `cacheAsBitmap` w obiektach `MovieClip` i w innych obiektach ekranowych o głównie niezmiennej zawartości ustaw wartość `true`. Następnie wykonaj względem tych obiektów przejścia przesuwane, przejścia skalowane i mieszanie alfa.
- Względem obiektów ekranowych, które mają zostać poddane skalowaniu lub translacji (zmianom współrzędnych x i y), użyj właściwości `cacheAsBitmapMatrix`.

Zastosowanie operacji klasy `Matrix` do skalowania i translacji sprawia, że te operacje są wykonywane przez akceleratory sprzętowe urządzenia. Załóżmy na przykład, że są zmieniane wymiary obiektu ekranowego, dla którego właściwości `cacheAsBitmap` ustawiono wartość `true`. Gdy te wymiary są zmieniane, oprogramowanie środowiska wykonawczego ponownie rysuje bitmapę. Ponowne rysowanie za pomocą oprogramowania oferuje niższą wydajność niż skalowanie za pomocą przyspieszania sprzętowego przy użyciu operacji `Matrix`.

Załóżmy, że pewna aplikacja wyświetla obraz, który jest powiększany po wybraniu przez użytkownika końcowego. Wielokrotne użycie operacji skalowania `Matrix` stwarza wrażenie powiększania obrazu. Jakość obrazu końcowego może jednak być nie do przyjęcia, w zależności od rozmiaru oryginalnego obrazu i obrazu końcowego. Z tego powodu po zakończeniu operacji powiększania należy wyzerować wymiary obiektu ekranowego. Wartość właściwości `cacheAsBitmap` wynosi `true`, więc środowisko wykonawcze ponownie rysuje obiekt ekranowy, ale tylko raz, po czym jest renderowany obraz o wysokiej jakości.

Uwaga: Zazwyczaj urządzenia telewizyjne ze środowiskiem AIR nie obsługują obracania ani pochylania przyspieszanego sprzętowo. Z tego powodu w przypadku określenia w klasie `Matrix` obracania i pochylania środowisko AIR dla urządzeń telewizyjnych wykonuje wszystkie operacje `Matrix` w oprogramowaniu. Te operacje programowe mogą obniżyć wydajność.

- W celu zrealizowania własnego mechanizmu buforowania bitmapy należy użyć klasy `BitmapData`.

Więcej informacji o buforowaniu bitmap zawierają następujące strony:

- [Buforowanie obiektów ekranowych](#)
- [Buforowanie bitmap](#)
- [Ręczne buforowanie bitmap](#)

Zarządzanie pamięcią graficzną

W celu wykonywania przyspieszanych operacji graficznych akceleratory sprzętowe korzystają ze specjalnej pamięci graficznej. Aplikacja używająca całej pamięci graficznej działa wolniej, gdyż środowisko AIR dla urządzeń telewizyjnych powraca do używania oprogramowania podczas operacji graficznych.

Aby zarządzać używaniem pamięci graficznej przez aplikację:

- Po zakończeniu używania obrazu lub innych danych bitmapowych należy zwolnić odpowiednią pamięć graficzną. Aby to zrobić, należy wywołać metodę `dispose()` właściwości `bitmapData` obiektu `Bitmap`. Na przykład:

```
myBitmap.bitmapData.dispose();
```

Uwaga: Zwolnienie odniesienia do obiektu `BitmapData` nie powoduje natychmiastowego zwolnienia pamięci graficznej. Proces czyszczenia pamięci środowiska wykonawczego spowoduje w końcu zwolnienie pamięci graficznej, ale wywołanie metody `dispose()` pozwala na dokładne sterowanie sytuacją przez aplikację.

- Oferowana przez firmę Adobe aplikacja AIR — PerfMaster Deluxe — pozwala lepiej zrozumieć mechanizm przyspieszania sprzętowego na urządzeniu docelowym. Ta aplikacja pokazuje liczbę klatek na sekundę podczas wykonywania różnych operacji. Korzystając z aplikacji PerfMaster Deluxe, można porównywać różne sposoby implementacji tej samej operacji. Można na przykład porównać przesuwanie obrazu bitmapowego z przesuwaniami obrazu wektorowego. Aplikacja PerfMaster Deluxe jest dostępna na stronie [Platforma Flash dla urządzeń telewizyjnych](#).

Zarządzanie listą wyświetlania

Aby uczynić obiekt ekranowy niewidzialnym, należy ustawić dla jego właściwości `visible` wartość `false`. Obiekt pozostanie na liście wyświetlania, ale nie będzie renderowany ani wyświetlany przez środowisko AIR dla urządzeń telewizyjnych. Ta technika jest przydatna w przypadku obiektów często pojawiających się i znikających, gdyż wymaga niewielu dodatkowych obliczeń. Ustawienie dla właściwości `visible` wartości `false` nie zwalnia jednak żadnych zasobów obiektu. Dlatego jeśli obiekt ekranowy nie jest już używany — lub nie będzie używany przez dłuższy czas — należy usunąć go z listy wyświetlania. Należy też ustawić wartość `null` dla wszystkich odniesień do tego obiektu. Pozwoli to procesowi czyszczenia pamięci zwolnić zasoby obiektu.

Używanie obrazów PNG i JPEG

Dwoma typowymi formatami obrazów w aplikacjach są PNG i JPEG. Rozważając używanie tych formatów w środowisku AIR dla urządzeń telewizyjnych, należy wziąć pod uwagę następujące czynniki:

- Środowisko AIR dla urządzeń telewizyjnych używa zazwyczaj przyspieszania sprzętowego w celu dekodowania plików JPEG.
- Środowisko AIR dla urządzeń telewizyjnych zazwyczaj dekoduje pliki PNG programowo. Dekodowanie plików PNG w oprogramowaniu przebiega szybko.
- Format PNG jest jedynym formatem bitmapowym współpracującym z wieloma platformami, który obsługuje przezroczystość (kanał alfa).

Z tego powodu w aplikacjach należy używać tych formatów obrazów w następujący sposób:

- Aby w przypadku zdjęć mogło być używane dekodowanie przyspieszane sprzętowo, należy używać plików JPEG.
- Plików PNG należy używać w elementach interfejsu użytkownika. Elementy interfejsu użytkownika mogą mieć ustawienie wartości alfa, a dekodowanie programowe zapewnia wystarczająco wysoką wydajność dla elementów interfejsu użytkownika.

Stół montażowy w aplikacjach AIR dla urządzeń telewizyjnych

Podczas tworzenia aplikacji AIR dla urządzeń telewizyjnych należy wziąć pod uwagę następujące aspekty związane z pracą klasą Stage:

- Rozdzielczość ekranu
- Bezpieczny obszar wyświetlania
- Tryb skalowania stołu montażowego
- Wyrównanie stołu montażowego
- Stan wyświetlania stołu montażowego
- Projektowanie na potrzeby wielu rozmiarów ekranu
- Ustawienia jakości stołu montażowego

Rozdzielczość ekranu

Obecnie urządzenia telewizyjne mają jedną z następujących rozdzielczości ekranu: 540p, 720p i 1080p. Te rozdzielczości ekranu dają zazwyczaj następujące wartości klasy `Capabilities` języka ActionScript:

Rozdzielczość ekranu	<code>Capabilities.screenResolutionX</code>	<code>Capabilities.screenResolutionY</code>
540p	960	540
720p	1280	720
1080p	1920	1080

Tworząc pełnoekranową aplikację AIR dla urządzeń telewizyjnych przeznaczoną na konkretne urządzenie, należy na stałe nadać właściwościom `Stage.stageWidth` i `Stage.stageHeight` wartości rozdzielczości ekranu urządzenia. Tworząc aplikację pełnoekranową działającą na wielu urządzeniach, do ustawiania wymiarów stołu montażowego należy używać właściwości `Capabilities.screenResolutionX` i `Capabilities.screenResolutionY`.

Na przykład:

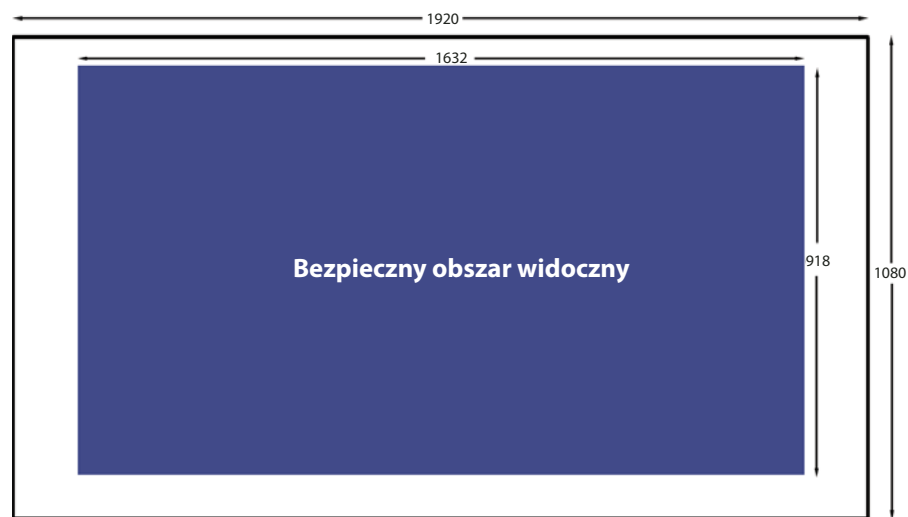
```
stage.stageWidth = Capabilities.screenResolutionX;  
stage.stageHeight = Capabilities.screenResolutionY;
```

Bezpieczny obszar wyświetlania

Bezpieczny obszar wyświetlania na telewizorze jest obszarem ekranu odsuniętym od krawędzi ekranu. Ten obszar jest wystarczająco odsunięty, aby użytkownik końcowy widział go w całości (bez zasłaniania fragmentów przez maskownicę telewizora). Maskownica, która jest fizyczną ramką dookoła ekranu, różni się w zależności od producenta, więc stosowane odsunięcie też musi się zmieniać. Bezpieczny obszar wyświetlania to próba zagwarantowania, że obszar ekranu będzie widoczny. Bezpieczny obszar wyświetlania jest również określany jako *bezpieczny obszar tytułowy*.

Nadmiarowość ekranu jest obszarem ekranu, który nie jest widoczny, ponieważ znajduje się poza maskownicą.

Firma Adobe zaleca stosowanie wstawki o wartości 7,5% dla każdej krawędzi ekranu. Na przykład:



Bezpieczny obszar wyświetlania w przypadku rozdzielczości ekranu wynoszącej 1920x1080

Tworząc pełnoekranową aplikację AIR dla urządzeń telewizyjnych należy zawsze pamiętać o stosowaniu bezpiecznego obszaru wyświetlania.

- Pracując z tłem, na przykład ustawiając obraz lub kolor tła, należy używać całego ekranu.
- Bezpiecznego obszaru wyświetlania należy używać wyłącznie w krytycznych elementach aplikacji, takich jak tekst, grafika, wideo i elementy interfejsu użytkownika (na przykład przyciski).

W poniższej tabeli przedstawiono wymiary bezpiecznego obszaru wyświetlania dla typowych rozdzielczości ekranu ze wstawkami o wartości 7,5%.

Rozdzielczość ekranu	Szerokość i wysokość bezpiecznego obszaru wyświetlania	Szerokość lewej i prawej wstawki	Wysokość lewej i prawej wstawki
960x540	816x460	72	40
1280x720	1088x612	96	54
1920x1080	1632x918	144	81

Najlepszą praktyką jest jednak zawsze dynamiczne obliczanie bezpiecznego obszaru wyświetlania. Na przykład:

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

Tryb skalowania stołu montażowego

Ustaw dla właściwości `Stage.scaleMode` wartość `StageScaleMode.NO_SCALE` i wykrywaj zdarzenia zmiany rozmiaru stołu montażowego.

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

To ustawienie powoduje, że współrzędne stołu montażowego są takie same jak współrzędne w pikselach. Wraz ze stanem wyświetlania `FULL_SCREEN_INTERACTIVE` i wyrównaniem stołu montażowego `TOP_LEFT` to ustawienie pozwala skutecznie korzystać z bezpiecznego obszaru wyświetlania.

W aplikacjach pełnoekranowych ten tryb oznacza, że właściwości `stageWidth` i `stageHeight` klasy `Stage` odpowiadają właściwościom `screenResolutionX` i `screenResolutionY` klasy `Capabilities`.

Ponadto gdy zmienia się rozmiar okna aplikacji, zawartość stołu montażowego utrzymuje zdefiniowany rozmiar. Środowisko wykonawcze nie wykonuje automatycznego skalowania ani wyrównywania układu. Gdy zmienia się rozmiar okna, środowisko wykonawcze wywołuje zdarzenie `resize` klasy `Stage`. Oznacza to, że aplikacja w pełni kontroluje sposób dostosowywania zawartości w momencie uruchamiania i zmiany rozmiaru okna.

Uwaga: Ustawienie wartości `NO_SCALE` powoduje takie zachowanie analogiczne do innych aplikacji AIR. W aplikacjach AIR dla urządzeń telewizyjnych stosowanie tego ustawienia ma jednak kluczowe znaczenie podczas używania bezpiecznego obszaru wyświetlania.

Wyrównanie stołu montażowego

Ustaw dla właściwości `Stage.align` wartość `StageAlign.TOP_LEFT`:

```
stage.align = StageAlign.TOP_LEFT;
```

Takie wyrównanie powoduje umieszczenie współrzędnych 0, 0 w lewym górnym rogu ekranu, co jest wygodne w przypadku umieszczania zawartości przy użyciu języka ActionScript.

Wraz z trybem skalowania `NO_SCALE` i stanem wyświetlania `FULL_SCREEN_INTERACTIVE` to ustawienie pozwala skutecznie używać bezpiecznego obszaru wyświetlania.

Stan wyświetlania stołu montażowego

W pełnoekranowej aplikacji AIR dla urządzeń telewizyjnych ustaw dla właściwości `Stage.displayState` wartość `StageDisplayState.FULL_SCREEN_INTERACTIVE`:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Ta wartość powoduje rozszerzenie stołu montażowego aplikacji AIR na cały ekran, gdy jest dozwolone wprowadzanie danych przez użytkownika.

Firma Adobe zaleca stosowanie ustawienia `FULL_SCREEN_INTERACTIVE`. Wraz z trybem skalowania `NO_SCALE` i wyrównaniem stołu montażowego `TOP_LEFT` to ustawienie pozwala skutecznie używać bezpiecznego obszaru wyświetlania.

Z tego powodu w aplikacjach pełnoekranowych w module obsługi zdarzenia `ADDED_TO_STAGE` w głównej klasie dokumentu należy używać następującego kodu:

```
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;
}
```

Następnie w module obsługi zdarzenia `RESIZE`:

- Porównaj wymiary rozdzielczości ekranu z szerokością i wysokością stołu montażowego. Jeśli są one takie same, nastąpiło zdarzenie `RESIZE`, gdyż wartość stanu wyświetlania stołu montażowego zmieniła się na `FULL_SCREEN_INTERACTIVE`.
- Oblicz i zapisz wymiary bezpiecznego obszaru wyświetlania i odpowiednich wstawek.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {
        // Calculate and save safe viewing area dimensions.
    }
}
```

Gdy wymiary stołu montażowego odpowiadają wartościom `Capabilities.screenResolutionX` i `screenResolutionY`, środowisko AIR dla urządzeń telewizyjnych używa dostępnego sprzętu w celu uzyskania możliwie najwyższej jakości wideo i grafiki.

Uwaga: Jakość wyświetlania grafiki i wideo na ekranie telewizora zależy od wartości

`Capabilities.screenResolutionX` i `screenResolutionY`, a te wartości są zależne od urządzenia, na którym działa środowisko AIR dla urządzeń telewizyjnych. Dekoder, na którym działa środowisko AIR dla urządzeń telewizyjnych, może na przykład mieć ekran o rozdzielczości 1280 x 720, podczas gdy podłączony do niego telewizor może mieć rozdzielczość ekranu 1920 x 1080. Środowisko AIR dla urządzeń telewizyjnych powoduje, że sprzęt wyświetla obraz o najwyższej dostępnej jakości. W tym przykładzie sprzęt wyświetla wideo w rozdzielczości 1080p, korzystając z rozdzielczości ekranu 1920 x 1080.

Projektowanie na potrzeby wielu rozmiarów ekranu

Aplikację AIR dla urządzeń telewizyjnych można opracować w taki sposób, aby działała i wyglądała dobrze na wielu urządzeniach telewizyjnych ze środowiskiem AIR. W tym celu należy wykonać następujące czynności:

- 1 Ustaw dla właściwości `scaleMode`, `align` i `displayState` stołu montażowego następujące wartości zalecane: odpowiednio `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` i `StageDisplayState.FULL_SCREEN_INTERACTIVE`.
- 2 Ustaw obszar bezpiecznego wyświetlania na podstawie właściwości `Capabilities.screenResolutionX` i `Capabilities.screenResolutionY`.
- 3 Dostosuj rozmiar i układ zawartości na podstawie szerokości i wysokości bezpiecznego obszaru wyświetlania.

Obiekty zawartości są duże, szczególnie w porównaniu z aplikacjami dla urządzeń przenośnych, ale koncepcje takie jak układ dynamiczny, pozycjonowanie względne i zawartość adaptacyjna są takie same. Więcej informacji dotyczących kodu ActionScript przeznaczonego do obsługi tych koncepcji zawiera temat [Tworzenie materiałów Flash dla urządzeń przenośnych przeznaczonych na ekrany o różnych rozmiarach](#).

Jakość stołu montażowego

Właściwość `Stage.quality` w aplikacji AIR dla urządzeń telewizyjnych ma zawsze wartość `StageQuality.High`. Nie można jej zmienić.

Ta właściwość określa jakość renderowania dla wszystkich obiektów `Stage`.

Obsługa sygnału pilota zdalnego sterowania

Użytkownicy korzystają zazwyczaj z aplikacji AIR dla urządzeń telewizyjnych za pomocą pilota zdalnego sterowania. Jednak dane wprowadzane za pomocą klawiszy należy obsługiwać w taki sam sposób, jak w aplikacji na komputerze stacjonarnym są obsługiwane dane wprowadzane za pomocą klawiatury. Należy obsługiwać zdarzenie `KeyboardEvent.KEY_DOWN`. Więcej informacji można znaleźć w temacie [Przechwytywanie naciśnień klawiszy](#) w dokumencie *ActionScript 3.0 — Podręcznik programistów*.

Klawisze na pilocie zdalnego sterowania są odwzorowywane na stałe ActionScript. Na przykład klawisze na klawiaturze kierunkowej znajdującej się na pilocie zdalnego sterowania są odwzorowywane w następujący sposób:

Klawisz klawiatury kierunkowej pilota zdalnego sterowania	Stała ActionScript 3.0
W górę	<code>Keyboard.UP</code>
W dół	<code>Keyboard.DOWN</code>
W lewo	<code>Keyboard.LEFT</code>
W prawo	<code>Keyboard.RIGHT</code>
Potwierdzenie lub wybór	<code>Keyboard.ENTER</code>

W środowisku AIR 2.5 dodano wiele innych stałych Keyboard przeznaczonych do obsługi sygnałów pilota zdalnego sterowania. Pełną listę można znaleźć w temacie [Klasa Keyboard](#) w *Dokumentacji języka ActionScript 3.0 dla platformy Adobe Flash*.

W celu zapewnienia działania aplikacji na jak największej liczbie urządzeń firma Adobe zaleca przestrzeganie następujących wytycznych:

- Należy używać wyłącznie klawiszy klawiatury kierunkowej, jeśli jest to możliwe.

Różne piloty zdalnego sterowania oferują różne zestawy klawiszy. Jednak zazwyczaj zawsze mają one klawisze klawiatury kierunkowej.

Na przykład pilot zdalnego sterowania odtwarzacza Blu-ray nie ma zazwyczaj klawiszy służących do zmiany kanału na kolejny lub poprzedni. Nawet klawisze odtwarzania, wstrzymywania i zatrzymywania nie występują na wszystkich pilotach zdalnego sterowania.

- Jeśli aplikacja wymaga używania innych klawiszy niż klawisze klawiatury kierunkowej, należy stosować klawisze Menu i Info.

Klawisze Menu i Info są następnymi najczęstszymi klawiszami występującymi na pilotach zdalnego sterowania.

- Należy wziąć pod uwagę częste używanie uniwersalnych pilotów zdalnego sterowania.

Nawet tworząc aplikację przeznaczoną na konkretne urządzenie, należy pamiętać o tym, że wielu użytkowników nie korzysta z pilota zdalnego sterowania dołączonego do urządzenia. Zamiast tego używają oni uniwersalnego pilota zdalnego sterowania. Użytkownicy nie zawsze programują również uniwersalne piloty zdalnego sterowania w taki sposób, aby dopasować klawisze do wszystkich klawiszy na pilocie zdalnego sterowania. Z tego powodu zalecane jest używanie tylko najbardziej typowych klawiszy.

- Należy upewnić się, że użytkownik może zawsze wyjść z danej sytuacji za pomocą jednego klawisza z klawiatury kierunkowej.

Czasami istnieje ważne uzasadnienie, aby aplikacja korzystała z klawisza, który nie jest jednym z najbardziej typowych klawiszy na pilotach zdalnego sterowania. Dzięki zastosowaniu możliwości wyjścia za pomocą jednego klawisza klawiatury kierunkowej obsługa aplikacji będzie możliwa na wszystkich urządzeniach.

- Nie należy wymagać wprowadzania danych za pomocą wskaźnika, chyba że docelowe urządzenie telewizyjne ze środowiskiem AIR oferuje taką możliwość.

Wiele aplikacji dla komputerów oczekuje wprowadzania danych za pomocą myszy, jednak większość telewizorów nie obsługuje wprowadzania danych za pomocą wskaźnika. W przypadku konwertowania aplikacji dla komputerów na aplikacje działające na telewizorach należy pamiętać o zmodyfikowaniu aplikacji w taki sposób, aby nie było oczekiwane wprowadzanie danych za pomocą myszy. Te modyfikacje obejmują zmiany obsługi zdarzeń i zmiany instrukcji przekazywanych użytkownikowi. Na przykład na ekranie startowym aplikacji nie należy wyświetlać komunikatu „Kliknij, aby rozpocząć”.

Zarządzanie aktywnością

Gdy element interfejsu użytkownika jest aktywny w aplikacji na komputery stacjonarne, jest on elementem docelowym zdarzeń wprowadzania danych przez użytkownika, takich jak zdarzenia klawiatury i myszy. Ponadto gdy element interfejsu użytkownika jest aktywny, zostaje on podświetlony przez aplikację. Zarządzanie aktywnością w aplikacji AIR dla urządzeń telewizyjnych różni się od zarządzania aktywnością w aplikacji na komputery stacjonarne, ponieważ:

- W aplikacjach na komputery stacjonarne do przenoszenia punktu skupienia do następnego elementu interfejsu użytkownika jest często używany klawisz Tab. Klawisz Tab nie jest używany w aplikacjach AIR dla urządzeń telewizyjnych. Piloty zdalnego sterowania nie mają zazwyczaj klawisza Tab. Z tego powodu zarządzanie aktywnością za pomocą właściwości `tabEnabled` obiektu `DisplayObject`, jak ma to miejsce w aplikacjach komputerowych, nie działa.
- Aplikacje dla komputerów stacjonarnych często oczekują, że użytkownik uaktywni element interfejsu użytkownika za pomocą myszy.

Dlatego w aplikacji należy wykonać następujące czynności:

- Dodaj do obiektu Stage detektor zdarzeń wykrywający zdarzenia Keyboard, takie jak `KeyboardEvent.KEY_DOWN`.
- Zadbaj o logikę aplikacji określającą, który element interfejsu użytkownika ma zostać wyróżniony dla użytkownika końcowego. Upewnij się, że podczas uruchamiania aplikacji zostaje wyróżniony element interfejsu użytkownika.
- Na podstawie logiki aplikacji powinno zostać wywołane zdarzenie Keyboard odebrane przez obiekt Stage względem odpowiedniego obiektu elementu interfejsu użytkownika.

Element interfejsu użytkownika można również uaktywniać za pomocą właściwości `Stage.focus` lub `Stage.assignFocus()`. Następnie można dodać do tego obiektu `DisplayObject` detektor zdarzeń w celu wykrywania zdarzeń klawiatury.

Projektowanie interfejsu użytkownika

Aby interfejs użytkownika aplikacji AIR dla urządzeń telewizyjnych działał poprawnie na telewizorach, należy przestrzegać poniższych zaleceń dotyczących:

- szybkości reakcji aplikacji,
- użyteczności aplikacji,
- dostosowania do oczekiwań i osobowości użytkownika.

Szybkość reakcji

Przestrzeganie poniższych wskazówek sprawi, że aplikacja AIR dla urządzeń telewizyjnych będzie działać tak szybko, jak to możliwe.

- Początkowy plik SWF aplikacji powinien być jak najmniejszy.

W początkowym pliku SWF należy wczytywać wyłącznie zasoby niezbędne do uruchomienia aplikacji. Na przykład należy wczytywać wyłącznie obraz ekranu startowego aplikacji.

Chociaż to zalecenie dotyczy również komputerowych aplikacji AIR, ma ono większe znaczenie na urządzeniach telewizyjnych ze środowiskiem AIR. Urządzenia telewizyjne ze środowiskiem AIR nie mają na przykład mocy obliczeniowej porównywalnej z komputerami stacjonarnymi. Ponadto aplikacja jest w nich przechowywana w pamięci flash, która nie jest tak szybka jak dyski twarde w komputerach stacjonarnych.

- Aplikacja powinna działać z szybkością co najmniej 20 klatek na sekundę.

Grafikę należy opracować, uwzględniając ten cel. Złożoność operacji graficznych może wpływać na liczbę klatek na sekundę. Wskazówki dotyczące zwiększania wydajności renderowania zawiera temat [Optymalizacja wydajności na platformie Adobe Flash](#).

Uwaga: Sprzęt graficzny w urządzeniach telewizyjnych ze środowiskiem AIR zazwyczaj aktualizuje zawartość ekranu z szybkością 60 Hz lub 120 Hz (60 lub 120 razy na sekundę). Sprzęt skanuje stół montażowy pod kątem aktualizacji co 30 lub 60 klatek na sekundę w przypadku wyświetlania obrazu na ekranie o częstotliwości 60 Hz lub 120 Hz. Jednak to, czy użytkownik skorzysta z tej wyższej szybkości odtwarzania, zależy od stopnia złożoności operacji graficznych aplikacji.

- Zawartość ekranu należy aktualizować w ciągu 100–200 milisekund od wprowadzenia danych przez użytkownika. Jeśli aktualizowanie trwa dłużej, użytkownicy często się niecierpliwią, co prowadzi do wielokrotnego naciskania klawiszy.

Użyteczność

Użytkownicy środowiska AIR dla urządzeń telewizyjnych znajdują się w „otoczeniu domowym”. Siedzą oni po drugiej stronie pokoju względem telewizora, w odległości około 3 metrów. Pomieszczenie jest czasami ciemne. Zazwyczaj do wprowadzania danych służy pilot zdalnego sterowania. Z aplikacji może korzystać więcej niż jedna osoba, czasami razem, czasami po kolei.

Z tego powodu w celu opracowania interfejsu użytkownika pod kątem używania na telewizorze należy postępować zgodnie z następującymi zaleceniami:

- Elementy interfejsu użytkownika należy uczynić dużymi.
Projektując tekst, przyciski lub inne elementy interfejsu użytkownika, należy pamiętać, że użytkownik siedzi po drugiej stronie pomieszczenia. Wszystko powinno być łatwe do zauważenia i odczytania z odległości na przykład 3 m. Nie należy zapełniać ekranu tylko dlatego, że jest on duży.
- Należy używać dobrego kontrastu, tak aby zawartość była łatwa do zauważenia i odczytania z drugiej strony pomieszczenia.
- Należy wyraźnie wskazać, który element interfejsu jest aktywny, oznaczając go jasnym kolorem.
- Ruch należy stosować wyłącznie w razie potrzeby. Na przykład przesuwanie z jednego ekranu na drugi w celu zachowania ciągłości może działać dobrze. Jednak ruch może być rozpraszający, jeśli nie pomaga to użytkownikowi w nawigowaniu lub jeśli nie jest niezbywalną cechą aplikacji.
- Należy zawsze zapewniać czytywisty sposób, za pomocą którego użytkownik może cofać się w interfejsie użytkownika.

Więcej informacji na temat używania pilota zdalnego sterowania zawiera sekcja „[Obsługa sygnału pilota zdalnego sterowania](#)” na stronie 140.

Oczekiwania i osobowość użytkownika

Należy pamiętać, że użytkownicy aplikacji AIR dla urządzeń telewizyjnych poszukują zazwyczaj rozrywki o telewizyjnej jakości w przyjemnym i zrelaksowanym otoczeniu. Niekoniecznie mają oni dużą wiedzę na temat komputerów i techniki.

Dlatego projektując aplikacje AIR dla urządzeń telewizyjnych, należy pamiętać o następujących aspektach:

- Nie należy używać terminów technicznych.
- Należy unikać modalnych okien dialogowych
- Należy używać przyjaznych, nieformalnych instrukcji odpowiednich dla środowiska domowego, a nie środowiska zawodowego czy technicznego.

- Należy używać grafiki charakteryzującej się wysoką jakością produkcji oczekiwaną przez widzów.
- Należy opracować interfejs użytkownika umożliwiający łatwą współpracę z pilotem zdalnego sterowania. Nie należy stosować elementów interfejsu użytkownika ani elementów projektu, które są lepiej dostosowane do aplikacji dla komputerów lub urządzeń przenośnych. Na przykład interfejs użytkownika na komputerach i urządzeniach przenośnych często wymaga wskazywania i klikania przycisków za pomocą myszy lub palca.

Czcionki i tekst

W aplikacji AIR dla urządzeń telewizyjnych można używać czcionek urządzenia lub czcionek osadzonych.

Czcionki urządzenia to czcionki zainstalowane na urządzeniu. Na wszystkich urządzeniach telewizyjnych ze środowiskiem AIR występują następujące czcionki:

Nazwa czcionki	Opis
<code>_sans</code>	Czcionka urządzenia <code>_sans</code> charakteryzuje się krojem pisma sans-serif. Czcionką urządzenia <code>_sans</code> zainstalowaną na wszystkich urządzeniach telewizyjnych ze środowiskiem AIR jest czcionka Myriad Pro. Na telewizorze krój pisma sans serif wygląda zazwyczaj lepiej niż kroje pisma serif ze względu na odległość oglądania.
<code>_serif</code>	Czcionka urządzenia <code>_serif</code> charakteryzuje się krojem pisma serif. Czcionką urządzenia <code>_serif</code> zainstalowaną na wszystkich urządzeniach telewizyjnych ze środowiskiem AIR jest czcionka Minion Pro.
<code>_typewriter</code>	Czcionka urządzenia <code>_typewriter</code> jest czcionką nieproporcjonalną. Czcionką urządzenia <code>_typewriter</code> zainstalowaną na wszystkich urządzeniach telewizyjnych ze środowiskiem AIR jest czcionka Courier Std.

Na wszystkich urządzeniach telewizyjnych ze środowiskiem AIR występują również następujące czcionki azjatyckie:

Nazwa czcionki	Język	Kategoria kroju pisma	Kod regionalny
RyoGothicPlusN-Regular	japoński	sans	ja
RyoTextPlusN-Regular	japoński	serif	ja
AdobeGothicStd-Light	koreański	sans	ko
AdobeHeitiStd-Regular	chiński uproszczony	sans	zh_CN
AdobeSongStd-Light	chiński uproszczony	serif	zh_CN
AdobeMingStd-Light	chiński tradycyjny	serif	zh_TW i zh_HK

Te czcionki urządzeń telewizyjnych ze środowiskiem AIR:

- pochodzą z biblioteki Adobe® Type Library;
- wyglądają dobrze na telewizorach;
- są przeznaczone do tworzenia napisów w filmach wideo;
- stanowią kontury czcionek, a nie czcionki bitmapowe.

Uwaga: Producenci urządzeń często umieszczają na urządzeniach inne czcionki. Takie czcionki zapewniane przez producenta są instalowane oprócz czcionek urządzeń telewizyjnych ze środowiskiem AIR.

Firma Adobe oferuje aplikację o nazwie FontMaster Deluxe, która umożliwia wyświetlenie wszystkich czcionek dostępnych na urządzeniu. Ta aplikacja jest dostępna w temacie [Platforma Flash dla urządzeń telewizyjnych](#).

Czcionki można również osadzić w aplikacji AIR dla urządzeń telewizyjnych. Więcej informacji dotyczących osadzania czcionek zawiera temat [Zaawansowane renderowanie tekstu](#) w dokumencie *ActionScript 3.0 — Podręcznik programistów*.

Oto zalecenia firmy Adobe dotyczące używania pól tekstowych TLF:

- Pól tekstowych TLF należy używać w przypadku tekstu w językach azjatyckich, aby korzystać z ustawień regionalnych, w którym działa aplikacja. Należy ustawić właściwość `locale` obiektu `TextLayoutFormat` powiązanego z obiektem `TLFTextField`. Aby określić bieżące ustawienie regionalne, należy zapoznać się z tematem [Wybieranie ustawień regionalnych](#) w dokumencie *ActionScript 3.0 — Podręcznik programistów*.
- Jeśli czcionka nie jest jedną z czcionek urządzeń telewizyjnych ze środowiskiem AIR, należy określić nazwę tej czcionki we właściwości `fontFamily` obiektu `TextLayoutFormat`. Środowisko AIR dla urządzeń telewizyjnych używa czcionki, jeśli jest ona dostępna na urządzeniu. Jeśli wymagana czcionka nie jest dostępna na urządzeniu, w zależności od ustawienia `locale` środowisko AIR dla urządzeń telewizyjnych zastępuje ją odpowiednią czcionką urządzenia telewizyjnego ze środowiskiem AIR.
- Aby środowisko AIR dla urządzeń telewizyjnych mogło wybrać odpowiednią czcionkę urządzenia telewizyjnego ze środowiskiem AIR, wraz z ustawieniem właściwości `locale` należy określić wartość `_sans`, `_serif`, lub `_typewriter` dla właściwości `fontFamily`. W zależności od ustawienia regionalnego środowisko AIR dla urządzeń telewizyjnych wybierze czcionkę z zestawu azjatyckich lub innych niż azjatyckie czcionki urządzenia. Te ustawienia oferują łatwy sposób automatycznego używania poprawnej czcionki dla czterech głównych azjatyckich ustawień regionalnych i dla języka angielskiego.

Uwaga: W celu zagwarantowania prawidłowego renderowania w przypadku używania klasycznych pól tekstowych dla tekstu w języku azjatyckim należy określić nazwę czcionki urządzenia telewizyjnego ze środowiskiem AIR. Jeśli wiadomo, że na urządzeniu docelowym jest również zainstalowana inna czcionka, można określić także tę czcionkę.

Należy wziąć pod uwagę następujące aspekty dotyczące wydajności aplikacji:

- Klasyczne pola tekstowe oferują wyższą wydajność niż pola tekstowe TLF.
- Najwyższą wydajność oferuje klasyczne pole tekstowe korzystające z czcionek bitmapowych.

W przeciwieństwie do czcionek konturowych, które podają wyłącznie dane konturów dla każdego znaku, czcionki bitmapowe zawierają bitmapę dla każdego znaku. Czcionkami bitmapowymi mogą być zarówno czcionki urządzenia, jak i czcionki osadzone.

- W przypadku określenia czcionki urządzenia należy upewnić się, że na urządzeniu docelowym jest zainstalowana dana czcionka urządzenia. Jeśli nie będzie ona zainstalowana na urządzeniu, środowisko AIR dla urządzeń telewizyjnych znajdzie i zastosuje inną czcionkę zainstalowaną na urządzeniu. Jednak takie zachowanie obniża wydajność aplikacji.
- Tak jak w przypadku każdego innego obiektu ekranowego, jeśli obiekt `TextField` prawie się nie zmienia, dla właściwości `cacheAsBitmap` obiektu należy ustawić wartość `true`. To ustawienie zwiększa wydajność w przypadku przejść takich jak zanikanie, przesuwanie i mieszanie alfa. Do skalowania i translacji należy używać właściwości `cacheAsBitmapMatrix`. Więcej informacji można znaleźć w rozdziale „[Sprzętowe przyspieszanie grafiki](#)” na stronie 134.

Zabezpieczenia systemu plików

Aplikacje AIR dla urządzeń telewizyjnych są aplikacjami AIR, więc mogą uzyskiwać dostęp do systemu plików urządzenia. Jednak w przypadku urządzeń „domowych” jest niezwykle ważne, aby aplikacja nie mogła uzyskiwać dostępu do plików systemowych urządzenia ani do plików innych aplikacji. Użytkownicy telewizorów i powiązanych urządzeń nie oczekują ani nie tolerują żadnych awarii — w końcu oglądają telewizję.

Dlatego aplikacja AIR dla urządzeń telewizyjnych ma ograniczony wgląd w system plików urządzenia. Dzięki użyciu języka ActionScript 3.0 aplikacja może uzyskiwać dostęp do konkretnych katalogów (i ich podkatalogów). Nazwy katalogów używane w języku ActionScript nie są rzeczywistymi nazwami katalogów na urządzeniu. Ten dodatkowy poziom zabezpieczeń chroni aplikacje AIR dla urządzeń telewizyjnych od złośliwego lub przypadkowego uzyskiwania dostępu do plików lokalnych, które do nich nie należą.

Szczegółowe informacje zawiera temat [Widok katalogów w aplikacjach AIR dla urządzeń telewizyjnych](#).

Obszar izolowany aplikacji AIR

Aplikacje AIR dla urządzeń telewizyjnych działają w obszarze izolowanym aplikacji AIR opisanym w temacie [Obszar izolowany aplikacji AIR](#).

Jedyną różnicą w przypadku aplikacji AIR dla urządzeń telewizyjnych jest to, że mają one ograniczony dostęp do systemu plików, zgodnie z opisem w sekcji „[Zabezpieczenia systemu plików](#)” na stronie 145.

Cykl życia aplikacji

W przeciwieństwie do działania w środowisku komputerowym użytkownik nie może zamknąć okna, w którym działa aplikacja AIR dla urządzeń telewizyjnych. Dlatego należy zapewnić mechanizm interfejsu użytkownika służący do kończenia działania aplikacji.

Zazwyczaj urządzenie pozwala użytkownikowi końcowemu bezwarunkowo zakończyć działanie aplikacji za pomocą klawisza zakończenia na pilocie zdalnego sterowania. Środowisko AIR dla urządzeń telewizyjnych nie wywołuje jednak zdarzenia `flash.events.Event.EXITING` względem aplikacji. Z tego powodu należy często zapisywać stan aplikacji, aby mogła ona przywrócić się do rozsądnego stanu przy następnym uruchomieniu.

Pliki cookie języka HTML

Środowisko AIR dla urządzeń telewizyjnych obsługuje trwałe pliki cookie HTTP i pliki cookie sesji. Środowisko AIR dla urządzeń telewizyjnych przechowuje pliki cookie poszczególnych aplikacji AIR w odpowiednich katalogach:

```
/app-storage/<app id>/Local Store
```

Plik zawierający pliki cookie ma nazwę `cookies`.

Uwaga: Środowisko AIR na innych urządzeniach, takich jak komputery, nie przechowuje plików cookie osobno dla każdej aplikacji. Funkcja przechowywania plików cookie specyficznych dla aplikacji obsługuje model zabezpieczeń aplikacji i systemu w środowisku AIR dla urządzeń telewizyjnych.

Właściwości `URLRequest.manageCookies` języka ActionScript należy używać w następujący sposób:

- Ustaw dla właściwości `manageCookies` wartość `true`. Jest to wartość domyślna. Oznacza to, że środowisko AIR dla urządzeń telewizyjnych automatycznie dodaje pliki cookie do żądań HTTP i zapamiętuje pliki cookie w odpowiedzi HTTP.

Uwaga: Nawet gdy wartość właściwości `manageCookies` wynosi `true`, aplikacja może dodać plik cookie do żądania HTTP za pomocą właściwości `URLRequest.requestHeaders`. Jeśli ten plik cookie ma tę samą nazwę co plik cookie zarządzany przez środowisko AIR dla urządzeń telewizyjnych, żądanie zawiera dwa pliki cookie o tej samej nazwie. Wartości obu plików cookie mogą się różnić.

- Dla właściwości `manageCookies` należy ustawić wartość `false`. Ta wartość oznacza, że to aplikacja jest odpowiedzialna za wysyłanie plików cookie w żądaniach HTTP i zapamiętywanie plików cookie w odpowiedzi HTTP.

Więcej informacji zawiera [opis klasy URLRequest](#).

Etapy programowania aplikacji AIR dla urządzeń telewizyjnych

Aplikacje AIR dla urządzeń telewizyjnych można opracowywać za pomocą następujących narzędzi do programowania platformy Adobe Flash:

- Adobe Flash Professional
Program Adobe Flash Professional CS5.5 obsługuje środowisko AIR 2.5 dla urządzeń telewizyjnych — pierwszą wersję środowiska AIR obsługującą aplikacje AIR dla urządzeń telewizyjnych.
- Adobe Flash® Builder®
Program Flash Builder 4.5 obsługuje środowisko AIR 2.5 dla urządzeń telewizyjnych.
- Zestaw SDK środowiska AIR
Począwszy od wersji AIR 2.5, aplikacje można opracowywać za pomocą narzędzi wiersza poleceń dołączonych do zestawu SDK środowiska AIR. Zestaw SDK środowiska AIR można pobrać na stronie <http://www.adobe.com/pl/products/air/sdk/>.

Korzystanie z programu Flash Professional

Używanie oprogramowania Flash Professional do programowania, testowania i publikowania aplikacji AIR dla urządzeń telewizyjnych odbywa się podobnie do stosowania tego narzędzia podczas opracowywania aplikacji AIR dla komputerów.

Jednak podczas pisania kodu ActionScript 3.0 należy używać wyłącznie klas i metod obsługiwanych przez profile `tv` i `extendedTV` środowiska AIR. Szczegółowe informacje zawiera sekcja „[Profile urządzeń](#)” na stronie 259.

Ustawienia projektu

W celu skonfigurowania projektu pod kątem aplikacji AIR dla urządzeń telewizyjnych należy wykonać następujące czynności:

- Na karcie Flash okna dialogowego Ustawienia publikowania ustaw opcję Odtwarzacz na wartość odpowiadającą wersji AIR 2.5 lub nowszej.
- Na karcie Ogólne okna dialogowego Ustawienia środowiska Adobe AIR (Ustawienia instalatora i aplikacji) ustaw profil `Telewizyjny` lub `Rozszerzony telewizyjny`.

Debugowanie

Aplikację można uruchomić za pomocą programu AIR Debug Launcher w oprogramowaniu Flash Professional. W tym celu należy wykonać następujące czynności:

- Aby uruchomić aplikację w trybie debugowania, wybierz opcję:
Debuguj > Debuguj film > W programie AIR Debug Launcher (lokalnie)
Po wybraniu tej opcji można ponownie uruchomić debugowanie, wybierając opcję:
Debuguj > Debuguj film > Debuguj
- Aby uruchomić aplikację bez funkcji debugowania, wybierz opcję:
Sterowanie > Testuj film > W programie AIR Debug Launcher (lokalnie)

Aby ponownie uruchomić aplikację po dokonaniu tego wyboru, można wybrać opcję Sterowanie > Testuj film > Testuj.

Został ustawiony profil telewizyjny lub rozszerzony telewizyjny środowiska AIR, więc program AIR Debug Launcher zawiera menu o nazwie Przyciski pilota zdalnego sterowania. Za pomocą tego menu można symulować naciśnięcie przycisków na pilocie zdalnego sterowania.

Więcej informacji zawiera rozdział „[Zdalne debugowanie za pomocą programu Flash Professional](#)” na stronie 157.

Korzystanie z rozszerzeń natywnych

Jeśli aplikacja używa rozszerzenia natywnego, skorzystaj z instrukcji w rozdziale „[Lista zadań dotycząca korzystania z rozszerzenia natywnego](#)” na stronie 161.

Gdy aplikacja korzysta z rozszerzeń natywnych, mają zastosowanie następujące uwagi:

- Aplikacji nie można opublikować za pomocą oprogramowania Flash Professional. Aby opublikować aplikację, należy użyć narzędzia ADT. Zobacz „[Pakowanie przy użyciu narzędzia ADT](#)” na stronie 153.
- Aplikacji nie można uruchamiać ani debugować za pomocą programu Flash Professional. Aby debugować aplikację na urządzeniu używanym do programowania, należy użyć narzędzia ADL. Zobacz „[Symulowanie urządzenia przy użyciu narzędzia ADL](#)” na stronie 154.

Korzystanie z programu Flash Builder

Począwszy od wersji Flash Builder 4.5, program Flash Builder obsługuje opracowywanie aplikacji AIR dla urządzeń telewizyjnych. Używanie oprogramowania Flash Builder do programowania, testowania i publikowania aplikacji AIR dla urządzeń telewizyjnych jest podobne do stosowania tego narzędzia podczas opracowywania aplikacji AIR na komputery stacjonarne.

Konfigurowanie aplikacji

Należy upewnić się, że aplikacja:

- Używa jako klasy kontenera w pliku MXML elementu `Application`, jeśli jest używany plik MXML.

```
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx">
```

```
<!-- Place elements here.-->
```

```
</s:Application>
```

Ważne: Aplikacje AIR dla urządzeń telewizyjnych nie obsługują elementu `WindowedApplication`.

Uwaga: Używanie pliku MXML nie jest w ogóle konieczne. Zamiast tego można utworzyć projekt w języku `ActionScript 3.0`.

- Korzysta wyłącznie z klas i metod języka `ActionScript 3.0` obsługujących profile `tv` i `extendedTV` środowiska AIR. Szczegółowe informacje zawiera sekcja „[Profile urządzeń](#)” na stronie 259.

Ponadto w pliku XML aplikacji należy upewnić się, że:

- Dla atrybutu `xmlns` elementu `application` jest ustawiona wartość `AIR 2.5`.

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```
- Element `supportedProfiles` zawiera ustawienie `tv` lub `extendedTV`.

```
<supportedProfiles>tv</supportedProfiles>
```

Debugowanie aplikacji

Aplikację można uruchomić za pomocą programu AIR Debug Launcher w oprogramowaniu Flash Builder. W tym celu należy wykonać następujące czynności:

- 1 Wybierz opcję Uruchom > Konfiguracje debugowania.
- 2 Upewnij się, że w polu Profil jest ustawiona wartość Komputerowy.
- 3 Wybierz opcję Uruchom > Debuguj w celu uruchomienia w trybie debugowania lub Uruchom > Uruchom w celu uruchomienia bez funkcji debugowania.

Za pomocą elementu `supportedProfiles` została ustawiona obsługa profilu telewizyjnego lub rozszerzonego profilu telewizyjnego, więc program AIR Debug Launcher zawiera menu o nazwie Przyciski pilota zdalnego sterowania. Za pomocą tego menu można symulować naciskanie przycisków na pilocie zdalnego sterowania.

Więcej informacji zawiera rozdział „Zdalne debugowanie za pomocą programu Flash Builder” na stronie 157.

Korzystanie z rozszerzeń natywnych

Jeśli aplikacja używa rozszerzenia natywnego, skorzystaj z instrukcji w rozdziale „Lista zadań dotycząca korzystania z rozszerzenia natywnego” na stronie 161.

Jednak gdy aplikacja korzysta z rozszerzeń natywnych:

- Aplikacji nie można publikować za pomocą oprogramowania Flash Builder. Aby opublikować aplikację, należy użyć narzędzia ADT. Zobacz „Pakowanie przy użyciu narzędzia ADT” na stronie 153.
- Aplikacji nie można uruchamiać ani debugować za pomocą programu Flash Builder. Aby debugować aplikację na urządzeniu używanym do programowania, należy użyć narzędzia ADL. Zobacz „Symulowanie urządzenia przy użyciu narzędzia ADL” na stronie 154.

Właściwości deskryptora aplikacji AIR dla urządzeń telewizyjnych

Podobnie jak w przypadku innych aplikacji AIR, podstawowe właściwości aplikacji należy ustawić w pliku deskryptora aplikacji. Aplikacje o profilu telewizyjnym ignorują niektóre ustawienia specyficzne dla środowiska na komputerach, takie jak przezroczystość i rozmiar okna. W aplikacjach przeznaczonych dla urządzeń o profilu `extendedTV` mogą być używane rozszerzenia natywne. Te aplikacje określają używane rozszerzenia natywne w elemencie `extensions`.

Ustawienia wspólne

Niektóre ustawienia deskryptora aplikacji są istotne dla wszystkich aplikacji o profilu telewizyjnym.

Wymagana wersja środowiska wykonawczego AIR

Wersję środowiska wykonawczego AIR, która jest wymagana przez aplikację, należy określić przy użyciu przestrzeni nazw pliku deskryptora aplikacji.

Przestrzeń nazw (przypisana w elemencie `application`) w dużej mierze określa funkcje, których aplikacja może używać. Na przykład aplikacja korzysta z przestrzeni nazw środowiska AIR 2.5, ale użytkownik ma zainstalowaną nowszą wersję. W takim wypadku aplikacja nadal rozpoznaje zachowania środowiska AIR 2.5, nawet jeśli są one inne w nowszej wersji środowiska AIR. Dopiero zmiana przestrzeni nazw i opublikowanie aktualizacji pozwoli aplikacji na uzyskanie dostępu do nowych zachowań i funkcji. Poprawki zabezpieczeń stanowią ważny wyjątek od tej reguły.

Przestrzeń nazw należy określić przy użyciu atrybutu `xmlns` głównego elementu `application`.

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

Aplikacje AIR dla urządzeń telewizyjnych są obsługiwane od środowiska AIR w wersji 2.5.

Tożsamość aplikacji

Niektóre ustawienia powinny być niepowtarzalne dla każdej publikowanej aplikacji. Do tych ustawień należą elementy `id`, `name`, i `filename`.

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

Wersja aplikacji

Wersję aplikacji należy określić w elemencie `versionNumber`. Podczas określania wartości w elemencie `versionNumber` można użyć maksymalnie trzech liczb oddzielonych kropkami, na przykład „0.1.2”. Każda z tych liczb w numerze wersji może zawierać do trzech cyfr. Innymi słowy, największym dozwolonym numerem wersji jest „999.999.999”. Numer wersji nie musi zawierać wszystkich trzech liczb. Zarówno „1”, jak i „1.0” stanowią prawidłowe numery wersji.

Element `versionLabel` pozwala określić etykietę wersji. Po dodaniu etykieta wersji jest wyświetlana zamiast numeru wersji.

```
<versionNumber>1.23.7</versionNumber>  
<versionLabel>1.23 Beta 7</versionLabel>
```

Główny plik SWF aplikacji

Należy określić główny plik SWF aplikacji w elemencie potomnym `versionLabel` elementu `initialWindow`. Podczas wskazywania urządzeń docelowych w profilu tv należy użyć pliku SWF. (Aplikacje oparte na standardzie HTML nie są obsługiwane).

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

Plik musi zostać umieszczony w pakiecie aplikacji AIR (przy użyciu narzędzia ADT lub środowiska programistycznego). Samo umieszczenie odniesienia do nazwy w deskrytorze aplikacji nie powoduje automatycznego zawarcia pliku w pakiecie.

Właściwości ekranu głównego

Kilka elementów potomnych elementu `initialWindow` steruje początkowym wyglądem i działaniem ekranu głównego aplikacji. Większość z tych właściwości jest ignorowana na urządzeniach o profilu telewizyjnym. Można używać elementu `fullScreen`.

- Element `fullScreen` określa, czy aplikacja ma zajmować cały ekran urządzenia, czy współużytkować go z normalną karnacją systemu operacyjnego.

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

Element visible

Element `visible` jest elementem potomnym elementu `initialWindow`. Środowisko AIR dla urządzeń telewizyjnych ignoruje element `visible`, ponieważ zawartość aplikacji jest zawsze widoczna na urządzeniach telewizyjnych ze środowiskiem AIR.

Jednak dla elementu `visible` należy ustawić wartość `true`, jeśli aplikacja jest przeznaczona również dla komputerów.

Na komputerach ten element ma wartość domyślną `false`. Oznacza to, że jeśli element `visible` nie zostanie uwzględniony, zawartość aplikacji nie będzie widoczna na komputerach. Klasa `NativeWindow` języka `ActionScript` pozwala wyświetlać zawartość na komputerach, ale profile urządzeń telewizyjnych nie obsługują tej klasy. Jeśli zostanie podjęta próba użycia klasy `NativeWindow` w aplikacji uruchamianej na urządzeniu telewizyjnym ze środowiskiem AIR, wczytywanie aplikacji nie powiedzie się. Niezależnie od tego, czy jest wywoływana dowolna z metod klasy `NativeWindow`, aplikacja korzystająca z tej klasy nie jest wczytywana na urządzeniu telewizyjnym ze środowiskiem AIR.

Obsługiwane profile

Jeśli działanie aplikacji ma sens wyłącznie na urządzeniu telewizyjnym, wówczas można uniemożliwić jej instalację na urządzeniach innych typów. Element `supportedProfiles` pozwala wykluczyć inne profile z listy obsługiwanych.

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Jeśli aplikacja korzysta z rozszerzenia natywnego, na liście obsługiwanych profilów należy umieścić tylko profil `extendedTV`.

```
<supportedProfiles>extendedTV</supportedProfiles>
```

Pominięcie elementu `supportedProfiles` jest jednoznaczne z obsługiwaniem przez aplikację wszystkich profilów.

Nie należy umieszczać *wyłącznie* profilu `tv` na liście `supportedProfiles`. Niektóre urządzenia telewizyjne zawsze uruchamiają środowisko AIR dla urządzeń telewizyjnych w trybie odpowiadającym profilowi `extendedTV`. Dzięki temu środowisko AIR dla urządzeń telewizyjnych może uruchamiać aplikacje korzystające z rozszerzeń natywnych. Jeśli w elemencie `supportedProfiles` jest określony wyłącznie profil `tv`, taka deklaracja oznacza, że zawartość jest niezgodna z trybem środowiska AIR dla urządzeń telewizyjnych dotyczącym profilu `extendedTV`. Dlatego niektóre urządzenia telewizyjne nie wczytują aplikacji określających wyłącznie profil `tv`.

Listę klas `ActionScript` obsługiwanych w profilach `tv` i `extendedTV` podano w temacie „[Możliwości różnych profili](#)” na stronie 260.

Wymagane rozszerzenia natywne

Aplikacje obsługujące profil `extendedTV` mogą korzystać z rozszerzeń natywnych.

Wszystkie rozszerzenia natywne, których używa aplikacja AIR, należy zadeklarować w deskrytorze aplikacji przy użyciu elementów `extensions` i `extensionID`. W poniższym przykładzie przedstawiono składnię umożliwiającą określenie dwóch wymaganych rozszerzeń natywnych:

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

Jeśli dane rozszerzenie nie występuje na liście, aplikacja nie może z niego korzystać.

Element `extensionID` ma wartość taką samą jak element `id` w pliku deskryptora rozszerzenia. Plik deskryptora rozszerzenia jest plikiem XML o nazwie `extension.xml`. Jest on spakowany w pliku ANE otrzymanym od producenta urządzenia.

Jeśli rozszerzenie występuje na liście w elemencie `extensions`, ale nie jest zainstalowane na urządzeniu telewizyjnym ze środowiskiem AIR, to nie można uruchomić aplikacji. Wyjątkiem od tej zasady jest sytuacja, gdy plik ANE spakowany z aplikacją AIR dla urządzeń telewizyjnych zawiera wersję będącą symulatorem rozszerzenia. W takim przypadku aplikacja będzie mogła działać, korzystając z symulatora rozszerzenia. Wersja będąca symulatorem nie zawiera kodu natywnego, lecz kod `ActionScript`.

Ikony aplikacji

Wymagania dotyczące ikon aplikacji na urządzeniach telewizyjnych zależą od urządzenia. Producent urządzenia określa na przykład:

- wymagane ikony i rozmiary ikon;
- wymagane typy plików i konwencje nazewnictwa;
- sposób udostępniania ikon dla aplikacji, na przykład czy ikony są pakowane z aplikacją;
- czy w elemencie `icon` w pliku deskryptora aplikacji mają być określone ikony;
- zachowanie w przypadku, gdy aplikacja nie udostępnia ikon.

W celu uzyskania szczegółowych informacji należy skontaktować się z producentem.

Ustawienia ignorowane

Aplikacje uruchamiane na urządzeniach telewizyjnych ignorują ustawienia aplikacji dotyczące funkcji urządzeń przenośnych, okien macierzystych i systemów operacyjnych. Ignorowane są następujące ustawienia:

- `allowBrowserInvocation`
- `aspectRatio`
- `autoOrients`
- `customUpdateUI`
- `fileTypes`
- `height`
- `installFolder`
- `maximizable`
- `maxSize`
- `minimizable`
- `minSize`
- `programMenuFolder`
- `renderMode`
- `resizable`
- `systemChrome`
- `title`
- `transparent`
- `visible`
- `width`
- `x`
- `y`

Pakowanie pliku AIR aplikacji dla urządzeń telewizyjnych

Pakowanie przy użyciu narzędzia ADT

Korzystając z narzędzia wiersza polecenia ADT, można spakować plik AIR aplikacji dla urządzeń telewizyjnych. Począwszy od zestawu SDK środowiska AIR 2.5, narzędzie ADT obsługuje pakowanie plików na potrzeby urządzeń telewizyjnych. Przed przystąpieniem do pakowania należy skompilować cały kod ActionScript i MXML. Ponadto należy uzyskać certyfikat podpisywania kodu. Certyfikat można utworzyć przy użyciu polecenia `-certificate` narzędzia ADT.

Szczegółowy opis poleceń i opcji narzędzia ADT zawiera sekcja „[Narzędzie ADT](#)” na stronie 175.

Tworzenie pakietu AIR

Aby utworzyć pakiet AIR, należy użyć polecenia `package` narzędzia ADT.

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

Założenia dotyczące przykładu:

- Ścieżka do narzędzia ADT znajduje się w definicji ścieżek powłoki wiersza poleceń. (Zobacz „[Zmienne środowiskowe ścieżek](#)” na stronie 320).
- Certyfikat `codesign.p12` znajduje się w katalogu nadrzędnym, z którego jest uruchamiane polecenie ADT.

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Pliki aplikacji użyte w tym przykładzie to `myApp-app.xml` (plik deskryptora aplikacji), `myApp.swf` i katalog ikon.

Po uruchomieniu polecenia w przedstawiony sposób narzędzie ADT wyświetla monit o podanie hasła magazynu kluczy. W niektórych programach powłoki podczas wpisywania nie są wyświetlane znaki hasła. Po zakończeniu wpisywania należy po prostu nacisnąć klawisz `Enter`. Hasło można również dołączyć do polecenia ADT za pomocą parametru `storepass`.

Tworzenie pakietu AIRN

Jeśli aplikacja AIR dla urządzeń telewizyjnych korzysta z rozszerzenia natywnego, zamiast pakietu AIR należy utworzyć pakiet AIRN. Aby utworzyć pakiet AIRN, należy użyć polecenia `package` narzędzia ADT, wybierając wartość `airn` jako typ docelowy.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml myApp.swf icons -extdir C:\extensions
```

Założenia dotyczące przykładu:

- Ścieżka do narzędzia ADT znajduje się w definicji ścieżek powłoki wiersza poleceń. (Zobacz „[Zmienne środowiskowe ścieżek](#)” na stronie 320).
- Certyfikat `codesign.p12` znajduje się w katalogu nadrzędnym, z którego jest uruchamiane polecenie ADT.
- Parametr `-extdir` określa katalog zawierający pliki ANE używane przez aplikację.

Te pliki ANE zawierają składającą się tylko z kodu wersję symulatora rozszerzenia języka ActionScript. Wersja rozszerzenia języka zawierająca kod natywny jest instalowana na urządzeniu telewizyjnym ze środowiskiem AIR.

Polecenie należy uruchomić w katalogu zawierającym pliki aplikacji. Pliki aplikacji użyte w tym przykładzie to `myApp-app.xml` (plik deskryptora aplikacji), `myApp.swf` i katalog ikon.

Po uruchomieniu polecenia w przedstawiony sposób narzędzie ADT wyświetla monit o podanie hasła magazynu kluczy. W niektórych programach powłoki podczas wpisywania nie są wyświetlane znaki hasła. Po zakończeniu wpisywania należy po prostu nacisnąć klawisz Enter. Hasło można również dołączyć do polecenia za pomocą parametru `storepass`.

W przypadku aplikacji AIR dla urządzeń telewizyjnych korzystającej z rozszerzeń natywnych można również utworzyć plik AIRI. Plik AIRI jest taki sam jak plik AIRN, ale nie jest podpisany. Na przykład:

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Gdy aplikacja jest gotowa do podpisania, z pliku AIRI można utworzyć plik AIRN.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Więcej informacji zawiera rozdział [Programowanie rozszerzeń natywnych dla środowiska Adobe AIR](#).

Pakowanie przy użyciu programów Flash Builder i Flash Professional

Programy Flash Professional i Flash Builder umożliwiają publikowanie i eksportowanie pakietów AIR bez konieczności samodzielnego uruchamiania narzędzia ADT. Procedury tworzenia pakietów AIR dla aplikacji AIR opisano w dokumentacjach tych programów.

Jednak obecnie tylko narzędzie ADT pozwala tworzyć pakiety AIRN (pakiety aplikacji środowiska AIR dla urządzeń telewizyjnych, które korzystają z rozszerzeń natywnych).

Więcej informacji:

- [Pakowanie aplikacji AIR za pomocą programu Flash Builder](#)
- [Publikowanie dla środowiska Adobe AIR za pomocą programu Flash Professional](#)

Debugowanie aplikacji AIR dla urządzeń telewizyjnych

Symulowanie urządzenia przy użyciu narzędzia ADL

Najszybszym i najprostszym sposobem testowania i debugowania większości funkcji aplikacji jest uruchomienie jej na komputerze używanym do programowania za pośrednictwem narzędzia ADL (Adobe Debug Launcher).

Narzędzie ADL używa elementu `supportedProfiles` z deskryptora aplikacji w celu wybrania profilu do zastosowania. Szczegóły:

- Jeśli na liście jest więcej niż jeden profil, narzędzie ADL używa pierwszej pozycji na liście.
- Parametr `-profile` narzędzia ADL pozwala wybrać dowolny profil z listy `supportedProfiles`.
- Jeśli w deskrytorze aplikacji nie jest umieszczony element `supportedProfiles`, wówczas w argumencie `-profile` można określić dowolny profil.

Przykładowo za pomocą następującego polecenia można uruchomić aplikację w celu zasymulowania profilu `tv`.

```
adl -profile tv myApp-app.xml
```

Gdy na komputerze jest symulowany profil `tv` lub `extendedTV` przy użyciu narzędzia ADL, aplikacja jest uruchamiana w środowisku, które możliwie najbardziej przypomina urządzenie docelowe. Na przykład:

- Interfejsy API języka ActionScript, które nie są częścią profilu wskazanego w argumencie `-profile`, nie są dostępne.

- Narzędzie ADL umożliwia wprowadzanie poleceń sterujących urządzeniem, takich jak polecenia z pilota zdalnego sterowania, za pomocą poleceń menu.
- Określenie profilu `tv` lub `extendedTV` w argumencie `-profile` umożliwia narzędziu ADL symulowanie klasy `StageVideo` na komputerze.
- Określenie profilu `extendedTV` w argumencie `-profile` pozwala aplikacji korzystać z symulatorów rozszerzeń natywnych spakowanych w pliku aplikacji AIRN.

Narzędzie ADL uruchamia aplikację na komputerze, dlatego podczas testowania aplikacji AIR dla urządzeń telewizyjnych występują ograniczenia:

- Testowanie nie odzwierciedla wydajności aplikacji na urządzeniu. Wydajność należy przetestować na urządzeniu docelowym.
- Testowanie nie symuluje ograniczeń obiektu `StageVideo`. Zwykle klasa `StageVideo` (a nie klasa `Video`) jest używana do odtwarzania wideo podczas określania docelowych urządzeń telewizyjnych ze środowiskiem AIR. Klasa `StageVideo` korzysta z wydajności oferowanej przez rozwiązanie sprzętowe urządzenia, ale występują przy tym ograniczenia dotyczące wyświetlania. Narzędzie ADL odtwarza wideo na komputerze bez takich ograniczeń. Z tego względu odtwarzanie wideo należy przetestować na urządzeniu docelowym.
- Testowanie nie symuluje kodu natywnego zawartego w rozszerzeniu natywnym. W argumencie `-profile` narzędzia ADL można jednak określić profil `extendedTV`, który obsługuje rozszerzenia natywne. Narzędzie ADL pozwala przeprowadzać testy za pomocą zawierającej wyłącznie procedurę pośredniczącą lub wersję wersji symulatora rozszerzenia języka ActionScript (dołączoną do pakietu ANE). Zazwyczaj odpowiednie rozszerzenie języka zainstalowane na urządzeniu zawiera również kod natywny. Aby przetestować rozszerzenie języka z kodem natywnym, należy uruchomić aplikację na urządzeniu docelowym.

Więcej informacji zawiera rozdział „[AIR Debug Launcher \(ADL\)](#)” na stronie 169.

Korzystanie z rozszerzeń natywnych

Jeśli aplikacja korzysta z rozszerzeń natywnych, polecenie ADL przypomina to w następującym przykładzie:

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

Założenia dotyczące przykładu:

- Ścieżka do narzędzia ADL znajduje się w definicji ścieżek powłoki wiersza poleceń. (Zobacz „[Zmienne środowiskowe ścieżek](#)” na stronie 320).
- Bieżący katalog zawiera pliki aplikacji. Obejmują one pliki SWF i plik deskryptora aplikacji, którym w tym przykładzie jest plik `myApp-app.xml`.
- Parametr `-extdir` określa nazwę katalogu zawierającego katalog dla każdego rozszerzenia natywnego używanego przez aplikację. Każdy z tych katalogów zawiera *rozpakowany* plik ANE rozszerzenia natywnego. Na przykład:


```
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
```

Te rozpakowane pliki ANE zawierają wersję rozszerzenia w postaci symulatora zawierającego wyłącznie kod ActionScript. Wersja rozszerzenia języka zawierająca kod natywny jest instalowana na urządzeniu telewizyjnym ze środowiskiem AIR.

Więcej informacji zawiera rozdział [Programowanie rozszerzeń natywnych dla środowiska Adobe AIR](#).

Wprowadzanie poleceń sterujących

Narzędzie ADL symuluje przyciski pilota zdalnego sterowania urządzenia telewizyjnego. Naciśnięcia tych przycisków można wysłać do symulowanego urządzenia, używając menu wyświetlanego po uruchomieniu narzędzia ADL za pomocą jednego z profili telewizyjnych.

Rozmiar ekranu

Ustawiając parametr `-screensize` narzędzia ADL, można przetestować aplikację na ekranach o różnych rozmiarach. W tym celu należy określić ciąg zawierający cztery wartości wskazujące szerokości i wysokości normalnego i zmaksymalizowanego okna.

Zawsze należy określać wymiary w pikselach układu pionowego — z wartością szerokości mniejszą niż wartość wysokości. Na przykład:

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

Instrukcje śledzenia

Gdy aplikacja telewizyjna zostanie uruchomiona na komputerze, informacje wyjściowe śledzenia będą zapisywane w debugerze lub w oknie terminalu użytym do uruchomienia narzędzia ADL.

Zdalne debugowanie za pomocą programu Flash Professional

Za pomocą oprogramowania Flash Professional można zdalnie debugować aplikację AIR dla urządzeń telewizyjnych, gdy działa ona na urządzeniu docelowym. Czynności wymagane do skonfigurowania debugowania zdalnego zależą od urządzenia. Na przykład w zestawie Adobe® AIR® for TV MAX 2010 Hardware Development Kit znajduje się dokumentacja zawierająca szczegółowe instrukcje dla tego urządzenia.

Niezależnie od urządzenia docelowego w celu przygotowania do debugowania zdalnego należy wykonać następujące czynności:

- 1 W oknie dialogowym Ustawienia publikowania na karcie Flash zaznacz opcję *Zezwalaj na debugowanie*.
Ta opcja powoduje, że oprogramowanie Flash Professional dołącza informacje o debugowaniu do wszystkich plików SWF tworzonych z pliku FLA.
- 2 Na karcie Podpis okna dialogowego Ustawienia środowiska Adobe AIR (Ustawienia instalatora i aplikacji) wybierz opcję przygotowania pliku AIR Intermediate (AIRI).
Podczas programowania aplikacji wystarczy korzystać z pliku AIRI, który nie wymaga podpisu cyfrowego.
- 3 Opublikuj aplikację, tworząc plik AIRI.

Ostatnie czynności to zainstalowanie i uruchomienie aplikacji na urządzeniu docelowym. Te czynności zależą jednak od urządzenia.

Zdalne debugowanie za pomocą programu Flash Builder

Do zdalnego debugowania aplikacji AIR dla urządzeń telewizyjnych, gdy działa ona na urządzeniu docelowym, można również używać oprogramowania Flash Builder. Wymagane do tego czynności zależą od urządzenia.

Niezależnie od urządzenia docelowego w celu przygotowania do debugowania zdalnego należy wykonać następujące czynności:

- 1 Wybierz opcję Projekt > Eksportuj kompilację do publikacji. Wybierz opcję przygotowania pliku AIR Intermediate (AIRI).
Podczas programowania aplikacji wystarczy korzystać z pliku AIRI, który nie wymaga podpisu cyfrowego.
- 2 Opublikuj aplikację, tworząc plik AIRI.
- 3 Zmień pakiet AIRI aplikacji w taki sposób, aby zawierał on pliki SWF z informacjami o debugowaniu.
Pliki SWF zawierające informacje o debugowaniu znajdują się w katalogu projektu aplikacji oprogramowania Flash Builder, w katalogu o nazwie bin-debug. Zastąp pliki SWF w pakiecie AIRI plikami SWF z katalogu bin-debug.

W przypadku programowania na komputerze z systemem Windows można to zrobić, wykonując następujące czynności:

- 1 Zmień nazwę pliku pakietu AIRI w taki sposób, aby zamiast rozszerzenia nazwy pliku airi miała ona rozszerzenie zip.
- 2 Wyodrębnij zawartość pliku ZIP.
- 3 Zastąp pliki SWF w strukturze wyodrębnionych katalogów plikami z katalogu bin-debug.
- 4 Ponownie spakuj pliki w wyodrębnionym katalogu.
- 5 Ponownie zmień spakowany plik pakietu tak, aby miał on rozszerzenie nazwy pliku airi.

W przypadku programowania za pomocą komputera Mac czynności wymagane do zastąpienia plików zależą od urządzenia. Ogólnie wymagają one jednak wykonania następujących czynności:

- 1 Zainstaluj pakiet AIRI na urządzeniu docelowym.

- 2 Zastąp pliki SWF w katalogu instalacyjnym aplikacji na urządzeniu docelowym plikami SWF z katalogu bin-debug. Rozważmy na przykład urządzenie wchodzące w skład zestawu Adobe AIR for TV MAX 2010 Hardware Development Kit. Zainstaluj pakiety AIRI zgodnie z opisem w dokumentacji urządzenia. Następnie użyj polecenia telnet w wierszu poleceń na komputerze Mac używanym do programowania, aby uzyskać dostęp do urządzenia docelowego. Zastąp pliki SWF w katalogu instalacyjnym aplikacji /opt/adobe/stagecraft/apps/<nazwa aplikacji>/ plikami SWF z katalogu bin-debug.

Poniższe kroki dotyczą debugowania zdalnego za pomocą oprogramowania Flash Builder i urządzenia wchodzącego w skład zestawu Adobe AIR for TV MAX 2010 Hardware Development Kit.

- 1 Na komputerze z oprogramowaniem Flash Builder, czyli na komputerze używanym do programowania, uruchom aplikację AIR for TV Device Connector wchodzącą w skład zestawu MAX 2010 Hardware Development Kit. Pokazuje ona adres IP komputera używanego do programowania.
- 2 Na urządzeniu wchodzącym w skład zestawu uruchom aplikację DevMaster, która również wchodzi w skład zestawu.
- 3 W aplikacji DevMaster wpisz adres IP komputera używanego do programowania uzyskany za pomocą programu AIR for TV Device Connector.
- 4 W aplikacji DevMaster upewnij się, że jest zaznaczona opcja Enable Remote Debugging (Zezwalaj na debugowanie zdalne).
- 5 Zamknij aplikację DevMaster.
- 6 Na komputerze używanym do programowania wybierz w programie AIR for TV Connector opcję Start (Uruchom).
- 7 Na urządzeniu wchodzącym w skład zestawu uruchom inną aplikację. Sprawdź, czy w programie AIR for TV Device Connector są wyświetlane informacje o śledzeniu.

Jeśli nie są one wyświetlane, komputer używany do programowania i urządzenie wchodzące w skład zestawu nie są połączone. Upewnij się, że jest dostępny port na komputerze używanym do programowania stosowany do przekazywania informacji o śledzeniu. Można wybrać inny port w oprogramowaniu AIR for TV Device Connector. Sprawdź również, czy zapora sieciowa zezwala na dostęp do wybranego portu.

Następnie uruchom debugger w oprogramowaniu Flash Builder. W tym celu należy wykonać następujące czynności:

- 1 W programie Flash Builder wybierz opcję Uruchom > Konfiguracje debugowania.
- 2 Skopiuj nazwę projektu z istniejącej konfiguracji debugowania, która jest przeznaczona do debugowania lokalnego.
- 3 W oknie dialogowym Konfiguracje debugowania zaznacz opcję Aplikacja internetowa. Następnie wybierz ikonę Nowa konfiguracja uruchamiania.
- 4 Wklej nazwę projektu w polu Projekt.
- 5 W sekcji Adres URL lub Ścieżka do uruchomienia usuń zaznaczenie opcji Użyj domyślnych. Wpisz również w polu tekstowym wartość `about:blank`.
- 6 Naciśnij przycisk Zastosuj w celu zapisania zmian.
- 7 Wybierz opcję Debuguj w celu uruchomienia debugera oprogramowania Flash Builder.
- 8 Uruchom aplikację na urządzeniu wchodzącym w skład zestawu.

Teraz można używać oprogramowania Flash Builder na przykład do ustawiania punktów kontrolnych i badania zmiennych.

Rozdział 9: Korzystanie z rozszerzeń natywnych dla środowiska Adobe AIR

Rozszerzenia natywne dla środowiska Adobe AIR oferują interfejsy API języka ActionScript zapewniające dostęp do funkcji specyficznych dla urządzeń programowanych przy użyciu kodu natywnego. Programiści rozszerzeń natywnych czasami współpracują z producentami urządzeń. Mogą być również pracować w innych firmach.

Jeśli programujesz rozszerzenie natywne, zobacz [Programowanie rozszerzeń natywnych dla środowiska Adobe AIR](#).

Rozszerzenie natywne jest połączeniem następujących elementów:

- Klasy ActionScript
- Kod natywny

Programiści aplikacji AIR korzystający z rozszerzeń natywnych pracują tylko z klasami ActionScript.

Rozszerzenia natywne są przydatne w następujących sytuacjach:

- Implementacja w kodzie natywnym zapewnia dostęp do funkcji specyficznych dla platformy. Te funkcje specyficzne dla platformy nie są dostępne we wbudowanych klasach ActionScript i nie można ich implementować w klasach ActionScript specyficznych dla aplikacji. Implementacja w kodzie natywnym może zapewnić taką funkcjonalność, ponieważ ma dostęp do sprzętu i oprogramowania specyficznego dla urządzenia.
- Implementacja w kodzie natywnym może czasami działać szybciej niż implementacja korzystająca wyłącznie z języka ActionScript.
- Implementacja w kodzie natywnym może zapewnić dostęp do starszego kodu natywnego za pośrednictwem języka ActionScript.

Niektóre przykłady rozszerzeń natywnych są dostępne w Centrum programistów Adobe. Na przykład jedno z rozszerzeń natywnych zapewnia aplikacjom AIR dostęp do funkcji wibracji w systemie Android. Zobacz [Rozszerzenia natywne dla środowiska Adobe AIR](#).

Pliki rozszerzeń natywnych środowiska AIR (ANE)

Programiści rozszerzeń natywnych umieszczają rozszerzenia natywne w plikach ANE. Plik ANE jest plikiem archiwum zawierającym wszystkie niezbędne biblioteki i zasoby dla rozszerzenia natywnego.

W przypadku niektórych urządzeń plik ANE zawiera bibliotekę kodu natywnego używaną przez rozszerzenie natywne. Natomiast w przypadku innych urządzeń biblioteka kodu natywnego jest zainstalowana na urządzeniu. Czasami rozszerzenie natywne w ogóle nie zawiera kodu natywnego dla określonego urządzenia, lecz jest zaimplementowane wyłącznie przy użyciu kodu ActionScript.

Programista aplikacji AIR używa pliku ANE w następujący sposób:

- Umieszcza plik ANE w ścieżce biblioteki aplikacji, tak samo jak w przypadku dołączania pliku SWC w ścieżce biblioteki. Dzięki temu aplikacja może odnosić się do klas ActionScript w rozszerzeniu.

Uwaga: Podczas kompilowania aplikacji należy pamiętać o tym, aby w pliku ANE używać łączenia dynamicznego. Jeśli jest używany program Flash Builder, w panelu właściwości ścieżki kodu ActionScript w programie Builder należy wybrać opcję *Zewnętrzne*. Jeśli jest używany wiersz poleceń, należy podać opcję *-external-library-path*.

- Pakuje plik ANE razem z aplikacją AIR.

Rozszerzenia natywne a klasa NativeProcess języka ActionScript

Język ActionScript 3.0 udostępnia klasę NativeProcess. Ta klasa umożliwia aplikacji AIR uruchamianie procesów natywnych w systemie operacyjnym komputera. Ta możliwość jest podobna do działania rozszerzeń natywnych, które zapewniają dostęp do funkcji i bibliotek specyficznych dla platformy. Podejmując decyzję o użyciu klasy NativeProcess lub rozszerzenia natywnego, należy rozważyć następujące kwestie:

- Tylko profil AIR `extendedDesktop` obsługuje klasę NativeProcess. Dlatego w aplikacjach korzystających z profiliw AIR `mobileDevice` i `extendedMobileDevice` można stosować tylko rozszerzenia natywne.
- Programiści rozszerzeń natywnych często dostarczają implementacje natywne dla różnych platform, ale oferowany przez nich interfejs API języka ActionScript jest zazwyczaj taki sam na wszystkich platformach. Jeśli jest używana klasa NativeProcess, kod ActionScript do uruchamiania procesu natywnego może mieć różną postać na poszczególnych platformach.
- Klasa NativeProcess uruchamia osobny proces, natomiast rozszerzenie natywne działa w tym samym procesie co aplikacja AIR. Dlatego w przypadku obaw dotyczących awarii kodu użycie klasy NativeProcess jest bezpieczniejsze. Jednak stosowanie osobnego procesu może wymagać zaimplementowania obsługi komunikacji między procesami.

Rozszerzenia natywne a biblioteki klas ActionScript (pliki SWC)

Plik SWC to biblioteka klas ActionScript w formacie archiwum. Zawiera on plik SWF i inne pliki zasobów. Przy użyciu pliku SWC można wygodnie udostępnić klasy ActionScript zamiast poszczególnych plików kodu ActionScript i zasobów.

Pakiet rozszerzenia natywnego jest plikiem ANE. Podobnie jak plik SWC, plik ANE również jest biblioteką klas ActionScript zawierającą plik SWF i inne pliki w formacie archiwum. Najważniejszą różnicą między plikami ANE i SWC jest to, że tylko plik ANE może zawierać bibliotekę kodu natywnego.

***Uwaga:** Podczas kompilowania aplikacji należy pamiętać o tym, aby w pliku ANE używać łączenia dynamicznego. Jeśli jest używany program Flash Builder, w panelu właściwości ścieżki kodu ActionScript w programie Builder należy wybrać opcję *Zewnętrzne*. Jeśli jest używany wiersz poleceń, należy podać opcję `-external-library-path`.*

Więcej tematów Pomocy

[Informacje o plikach SWC](#)

Obsługiwane urządzenia

Począwszy od środowiska AIR 3, można korzystać z rozszerzeń natywnych w aplikacjach dla następujących urządzeń:

- Urządzenia z systemem Android (od wersji Android 2.2)
- Urządzenia z systemem iOS (od wersji iOS 4.0)

- Symulator systemu iOS (od wersji AIR 3.3)
- Urządzenia BlackBerry PlayBook
- Komputery z systemem Windows obsługujące środowisko AIR 3.0
- Komputery z systemem Mac OS X obsługujące środowisko AIR 3.0

W wielu przypadkach to samo rozszerzenie natywne jest przeznaczone dla kilku platform. Plik ANE rozszerzenia zawiera kod ActionScript i biblioteki natywne dla każdej obsługiwanej platformy. Zazwyczaj biblioteki ActionScript mają takie same interfejsy publiczne dla wszystkich tych platform. Biblioteki natywne z konieczności różnią się.

Czasami rozszerzenie natywne obsługuje platformę domyślną. Implementacja dla platformy domyślnej zawiera tylko kod ActionScript, bez kodu natywnego. Jeśli zostanie utworzony pakiet z aplikacją dla platformy nieobsługiwanej przez dane rozszerzenie, podczas uruchamiania aplikacji będzie używana implementacja domyślna. Rozszerzenie może na przykład udostępniać funkcję dotyczącą tylko urządzeń przenośnych. Takie rozszerzenie może również oferować domyślną implementację pozwalającą symulować tę funkcję w aplikacji dla komputerów.

Obsługiwane profile urządzeń

Następujące profile środowiska AIR obsługują rozszerzenia natywne:

- `extendedDesktop` (od wersji AIR 3.0)
- `mobileDevice` (od wersji AIR 3.0)
- `extendedMobileDevice` (od wersji AIR 3.0)

Więcej tematów Pomocy

[Obsługa profili AIR](#)

Lista zadań dotycząca korzystania z rozszerzenia natywnego

Aby korzystać z rozszerzenia natywnego w aplikacji, wykonaj następujące zadania:

- 1 Zadeklaruj rozszerzenie w pliku deskryptora aplikacji.
- 2 Umieść plik ANE w ścieżce biblioteki aplikacji.
- 3 Spakuj aplikację.

Deklarowanie rozszerzenia w pliku deskryptora aplikacji

Wszystkie aplikacje AIR mają plik deskryptora aplikacji. Gdy aplikacja korzysta z rozszerzenia natywnego, plik deskryptora aplikacji zawiera element `<extensions>`. Na przykład:

```
<extensions>
  <extensionID>com.example.Extension1</extensionID>
  <extensionID>com.example.Extension2</extensionID>
</extensions>
```

Element `extensionID` ma wartość taką samą jak element `id` w pliku deskryptora rozszerzenia. Plik deskryptora rozszerzenia jest plikiem XML o nazwie `extension.xml`. Jest on spakowany w pliku ANE. Plik `extension.xml` można wyświetlić za pomocą narzędzia do wyodrębniania archiwum.

Uwzględnianie pliku ANE w ścieżce biblioteki aplikacji

Aby skompilować aplikację korzystającą z rozszerzenia natywnego, należy umieścić plik ANE w ścieżce biblioteki.

Używanie pliku ANE z programem Flash Builder

Jeśli aplikacja korzysta z rozszerzenia natywnego, w ścieżce biblioteki należy umieścić plik ANE dotyczący tego rozszerzenia. Można wówczas kompilować kod ActionScript za pomocą oprogramowania Flash Builder.

Wykonaj następujące czynności, używając programu Flash Builder 4.5.1:

- 1 Zmień rozszerzenie nazwy pliku ANE z `ane` na `swc`. Ten krok jest konieczny, aby program Flash Builder mógł znaleźć plik.
- 2 W projekcie Flash Builder wybierz opcję Projekt > Właściwości.
- 3 W oknie dialogowym Właściwości wybierz ścieżkę kompilacji Flex.
- 4 Na karcie Ścieżka biblioteki wybierz opcję Dodaj plik SWC.
- 5 Przejdź do pliku SWC i wybierz opcję Otwórz.
- 6 Kliknij przycisk OK w oknie dialogowym dodawania pliku SWC.
Plik ANE zostanie wyświetlony na karcie Ścieżka biblioteki okna dialogowego Właściwości.
- 7 Rozwiń pozycję pliku SWC. Kliknij dwukrotnie opcję Typ łącza w celu otwarcia okna dialogowego Opcje elementu ścieżki biblioteki.
- 8 W oknie dialogowym Opcje elementu ścieżki biblioteki zmień Typ łącza na Zewnętrzne.
Aplikację można teraz skompilować na przykład za pomocą opcji Projekt > Utwórz projekt.

Używanie pliku ANE z programem Flash Professional

Jeśli aplikacja korzysta z rozszerzenia natywnego, w ścieżce biblioteki należy umieścić plik ANE dotyczący tego rozszerzenia. Można wówczas kompilować kod ActionScript za pomocą oprogramowania Flash Professional CS5.5. W tym celu należy wykonać następujące czynności:

- 1 Zmień rozszerzenie nazwy pliku ANE z `ane` na `swc`. Ten krok jest konieczny, aby program Flash Professional mógł znaleźć plik.
- 2 W pliku FLA wybierz opcję Plik > Ustawienia ActionScript.
- 3 Wybierz kartę Ścieżka biblioteki w oknie dialogowym Zaawansowane ustawienia języka ActionScript 3.0.
- 4 Naciśnij przycisk Przejdź do pliku SWC.
- 5 Przejdź do pliku SWC i wybierz opcję Otwórz.

Plik SWC jest teraz wyświetlany na karcie Ścieżka biblioteki okna dialogowego Zaawansowane ustawienia języka ActionScript 3.0.

- 6 Przy zaznaczonym pliku SWC naciśnij przycisk Wybierz opcje tworzenia połączeń biblioteki.
- 7 W oknie dialogowym Opcje elementu ścieżki biblioteki zmień opcję Typ łącza na Zewnętrzne.

Pakowanie aplikacji korzystającej z rozszerzeń natywnych

Za pomocą narzędzia ADT spakuj aplikację korzystającą z rozszerzeń natywnych. Nie można spakować aplikacji za pomocą programu Flash Professional CS5.5 lub Flash Builder 4.5.1.

Szczegółowe informacje o korzystaniu z narzędzia ADT znajdują się na stronie [Narzędzie ADT](#).

Na przykład poniższe polecenie narzędzia ADT powoduje utworzenie pliku DMG (pliku instalatora natywnego dla systemu Mac OS X) dla aplikacji korzystającej z rozszerzeń natywnych:

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

Następujące polecenie powoduje utworzenie pakietu APK dla urządzenia z systemem Android:

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

Następujące polecenie powoduje utworzenie pakietu systemu iOS dla aplikacji przeznaczonej dla telefonu 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
```

Należy zwrócić uwagę na następujące kwestie:

- Należy użyć typu pakietu instalatora natywnego.
- Należy określić katalog rozszerzeń.
- Należy upewnić się, że plik ANE obsługuje urządzenie docelowe aplikacji.

Używanie typu pakietu instalatora natywnego

Pakiet aplikacji musi być przeznaczony dla instalatora natywnego. W przypadku aplikacji korzystającej z rozszerzenia natywnego nie można utworzyć pakietu AIR przeznaczonego dla różnych platform (pakietu z rozszerzeniem air), ponieważ takie rozszerzenia często zawierają kod natywny. Jednak rozszerzenie natywne zazwyczaj obsługuje wiele platform natywnych przy użyciu tych samych interfejsów API języka ActionScript. W takich przypadkach można użyć tego samego pliku ANE w różnych pakietach instalatora natywnego.

Poniższa tabela zawiera podsumowanie wartości, których należy używać dla opcji `-target` w poleceniu narzędzia ADT:

Platforma docelowa aplikacji	-target
Komputery z systemem Mac OS X lub Windows	-target native -target bundle
Android	-target apk (lub inne typy docelowe pakietu Android)
iOS	-target ipa-ad-hoc (lub inne typy docelowe pakietu iOS)
Symulator systemu iOS	-target ipa-test-interpretor-simulator -target ipa-debug-interpretor-simulator

Określanie katalogu rozszerzeń

Za pomocą opcji `-extdir` narzędzia ADT należy wskazać katalog zawierający rozszerzenia natywne (pliki ANE).

Szczegółowe informacje na temat tej opcji zawiera sekcja „[Opcje plików i ścieżek](#)” na stronie 192.

Sprawdzanie, czy plik ANE obsługuje urządzenie docelowe aplikacji

Udostępniając plik ANE, programista rozszerzenia natywnego informuje użytkownika o tym, jakie platformy są przez niego obsługiwane. Można również sprawdzić zawartość pliku ANE za pomocą narzędzia do wyodrębniania archiwum. Struktura wyodrębnionych plików zawiera odpowiedni katalog dla każdej obsługiwanej platformy.

Znajomość platform obsługiwanych przez rozszerzenie jest ważna w przypadku pakowania aplikacji korzystającej z pliku ANE. Należy stosować następujące reguły:

- W przypadku pakowania aplikacji dla systemu Android plik ANE musi uwzględniać platformę `Android-ARM`. Innym rozwiązaniem jest uwzględnienie w pliku ANE platformy domyślnej i co najmniej jednej innej platformy.
- W przypadku pakowania aplikacji dla systemu iOS plik ANE musi uwzględniać platformę `iPhone-ARM`. Innym rozwiązaniem jest uwzględnienie w pliku ANE platformy domyślnej i co najmniej jednej innej platformy.
- W przypadku tworzenia pakietu aplikacji dla symulatora systemu iOS plik ANE musi uwzględniać platformę `iPhone-x86`.
- W przypadku pakowania aplikacji dla systemu Mac OS X plik ANE musi uwzględniać platformę `MacOS-x86`. Innym rozwiązaniem jest uwzględnienie w pliku ANE platformy domyślnej i co najmniej jednej innej platformy.
- W przypadku pakowania aplikacji dla systemu Windows plik ANE musi uwzględniać platformę `Windows-x86`. Innym rozwiązaniem jest uwzględnienie w pliku ANE platformy domyślnej i co najmniej jednej innej platformy.

Rozdział 10: Kompilatory języka ActionScript

Aby można było dołączyć kod ActionScript i MXML do aplikacji AIR, należy go skompilować. W przypadku używania środowiska programistycznego, na przykład oprogramowania Adobe Flash Builder lub Adobe Flash Professional, środowisko przeprowadza kompilację w tle. Można również wywoływać kompilatory języka ActionScript z poziomu wiersza poleceń, aby tworzyć pliki SWF, gdy nie jest używane środowisko programistyczne lub gdy jest używany skrypt kompilacji.

Informacje o narzędziach wiersza poleceń AIR w pakiecie Flex SDK

Każde z narzędzi wiersza poleceń używane do tworzenia aplikacji Adobe AIR wywołuje odpowiednie narzędzia używane do tworzenia aplikacji.

- `amxmlc calls mxmcl` — do kompilowania klas aplikacji
- `acompc calls compc` — do kompilowania klas bibliotek i składników
- `aasdoc calls asdoc` — do generowania plików dokumentacji z komentarzy w kodzie źródłowym

Jedyną różnicą między wersjami Flex i AIR tych programów narzędziowych jest to, że wersje AIR ładują opcje konfiguracji z pliku `air-config.xml` zamiast z pliku `flex-config.xml`.

Narzędzia zestawu SDK programu Flex i odpowiednie opcje wiersza poleceń zostały opisane kompleksowo w [dokumentacji programu Flex](#). Narzędzia Flex SDK zostały opisane w niniejszej sekcji na poziomie podstawowym, aby ułatwić rozpoczęcie pracy oraz aby wskazać różnice między tworzeniem aplikacji Flex i budowaniem aplikacji AIR.

Więcej tematów Pomocy

„[Tworzenie pierwszej aplikacji AIR na komputery stacjonarne za pomocą zestawu SDK środowiska Flex](#)” na stronie 39

Konfiguracja kompilatora

Opcje kompilacji zwykle określa się w wierszu poleceń oraz w jednym lub większej liczbie plików konfiguracyjnych. Globalny plik konfiguracyjny Flex SDK zawiera wartości domyślne, które są używane zawsze po uruchomieniu kompilatorów. Ten plik można edytować w celu dostosowania środowiska programowania. Katalog środowiska instalacji Flex SDK zawiera dwa globalne pliki konfiguracyjne Flex. Plik `air-config.xml` jest używany po uruchomieniu kompilatora `amxmlc`. Ten plik konfiguruje kompilatora dla środowiska AIR poprzez dołączanie bibliotek AIR. Plik `flex-config.xml` jest używany po uruchomieniu `mxmcl`.

Domyślne wartości konfiguracji są odpowiednie do wykrywania sposobów działania środowisk Flex i AIR, jednak w przypadku prac nad projektem w pełnej skali należy bliżej zapoznać się z dostępnymi opcjami. Do lokalnego pliku konfiguracyjnego można wprowadzić wartości właściwe dla projektu — wówczas te wartości będą miały priorytet wyższy niż wartości globalne.

Uwaga: Dla aplikacji AIR nie są używane żadne konkretne opcje kompilacji, ale podczas kompilowania aplikacji AIR należy tworzyć odwołania do bibliotek AIR. Zwykle odwołania te są tworzone w pliku konfiguracyjnym na poziomie projektu; w pliku dla narzędzia do tworzenia, takiego jak Ant, lub bezpośrednio w wierszu poleceń.

Kompilowanie plików źródłowych MXML i ActionScript dla środowiska AIR

Zasoby Adobe® ActionScript® 3.0 i MXML aplikacji AIR użytkownika można kompilować za pomocą kompilatora MXML (`amxmlc`) wywoływanego z wiersza poleceń. (Nie ma potrzeby kompilowania aplikacji opartych na kodzie HTML. W celu skompilowania pliku SWF w Flash Professional wystarczy opublikować film do pliku SWF).

Podstawowy wzorzec wiersza poleceń dla kompilatora `amxmlc`:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

gdzie *[compiler options]* określa opcje wiersza poleceń używane do kompilowania aplikacji AIR.

Polecenie `amxmlc` zawiera standardowy kompilator Flex `mxmlc` oraz dodatkowy parametr `+configname=air`. Ten parametr informuje kompilator o konieczności użycia pliku `air-config.xml` zamiast pliku `flex-config.xml`. Pod innymi względami stosowanie kompilatora `amxmlc` przebiega tak samo jak stosowanie kompilatora `mxmlc`.

Kompilator ładuje plik konfiguracji `air-config.xml`, określając biblioteki AIR i Flex wymagane do skompilowania aplikacji AIR. Możliwe jest również zastosowanie lokalnego pliku konfiguracji na poziomie projektu w celu zastąpienia opcji lub dodania opcji do konfiguracji globalnej. Zwykle najprostszym sposobem utworzenia pliku konfiguracji lokalnej jest edytowanie kopii wersji globalnej. Plik lokalny można załadować za pomocą opcji `-load-config`:

-load-config=project-config.xml Zastępuje opcje globalne.

-load-config+=project-config.xml Dodaje wartości do opcji globalnych, które przyjmują więcej niż jedną wartość, np. opcja `-library-path`. Globalne opcje, które przyjmują tylko jedną wartość, są zastępowane.

Jeśli stosowana jest określona konwencja nazewnictwa dla lokalnego pliku konfiguracji, wówczas kompilator `amxmlc` automatycznie ładuje plik lokalny. Przykład: jeśli główny plik MXML to `RunningMan.mxml`, wówczas lokalny plik konfiguracji powinien mieć nazwę: `RunningMan-config.xml`. W celu skompilowania aplikacji należy wpisać:

```
amxmlc RunningMan.mxml
```

Plik `RunningMan-config.xml` jest ładowany automatycznie, ponieważ nazwa jego pliku jest zgodna z nazwą skompilowanego pliku MXML.

amxmlc, przykłady

Poniższe przykłady prezentują zastosowania kompilatora `amxmlc`. (Wymagane jest skompilowanie samych zasobów ActionScript i MXML aplikacji).

Kompilacja pliku AIR MXML:

```
amxmlc myApp.mxml
```

Kompilacja i ustawianie nazwy wynikowej:

```
amxmlc -output anApp.swf -- myApp.mxml
```

Kompilacja pliku AIR ActionScript:

```
amxmlc myApp.as
```

Określenie pliku konfiguracji kompilatora:

```
amxmlc -load-config config.xml -- MyApp.mxml
```

Dodanie opcji z innego pliku konfiguracji:

```
amxmlc -load-config+=moreConfig.xml -- MyApp.mxml
```

Dodanie bibliotek w wierszu poleceń (oprócz bibliotek, które znajdują się już w pliku konfiguracji):

```
amxmlc -library-path+=/libs/libOne.swc,/libs/libTwo.swc -- MyApp.mxml
```

Kompilacja pliku AIR MXML bez użycia pliku kompilacji (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
```

Kompilacja pliku AIR MXML bez użycia pliku kompilacji (Mac OS X lub 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
```

Kompilacja pliku AIR MXML w celu użycia biblioteki współużytkowanej w środowisku wykonawczym:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --  
MyApp.mxml
```

Kompilacja pliku AIR MXML tak, aby był używany plik ANE (należy pamiętać o podaniu opcji

-external-library-path dla pliku ANE):

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- MyApp.mxml
```

Kompilacja z Java (ścieżka klasy uwzględnia mxmlc.jar):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional  
compiler options] -- MyApp.mxml
```

Opcja flexlib identyfikuje lokalizację katalogu struktur Flex SDK, dzięki czemu kompilator może zlokalizować plik flex_config.xml.

Kompilacja z Java (bez ustawionej ścieżki klasy):

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air  
[additional compiler options] -- MyApp.mxml
```

W celu wywołania kompilatora za pomocą Apache Ant (w przykładzie wykorzystano zadanie Java w celu uruchomienia pliku 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>
```

Kompilowanie składnika AIR oraz biblioteki kodu AIR (Flex)

Kompilator składników `acompc` służy do kompilowania bibliotek AIR i niezależnych składników. Kompilator składnika `acompc` działa jak kompilator `amxmlc` z następującymi wyjątkami:

- Należy określić klasy w kodzie, które będą zawierały bibliotekę lub składnik.
- `acompc` nie wyszukuje automatycznie lokalnego pliku konfiguracji. W celu użycia pliku konfiguracji projektu należy użyć opcji `-load-config`.

Polecenie `acompc` wywołuje standardowy kompilator Flex `compc`, ale ładuje jego opcje konfiguracji z pliku `air-config.xml` zamiast z pliku `flex-config.xml`.

Plik konfiguracji kompilatora składników

Użycie lokalnego pliku konfiguracji eliminuje potrzebę wpisywania ścieżki źródła oraz nazw klas w wierszu poleceń. W celu załadowania lokalnego pliku konfiguracji należy załadować opcję `-load-config` do wiersza poleceń `acompc`.

Poniższy przykład prezentuje konfigurację przeznaczoną do tworzenia biblioteki za pomocą dwóch klas: `ParticleManager` i `Particle` — obydwie pochodzą z pakietu: `com.adobe.samples.particles`. Pliki klas znajdują się w folderze `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>
```

W celu skompilowania biblioteki za pomocą pliku konfiguracji o nazwie `ParticleLib-config.xml` należy wpisać:

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

W celu wykonania polecenia w całości z wiersza poleceń należy wpisać:

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Całe polecenie należy wpisać do jednego wiersza lub należy użyć znaku kontynuacji wiersza dla powłoki poleceń).

acompc, przykłady

W tych przykładach założono, że używany jest plik konfiguracji o nazwie `myLib-config.xml`.

Kompilacja biblioteki lub składnika AIR:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Kompilacja biblioteki współużytkowanej w środowisku wykonawczym:

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Uwaga: folder `lib` musi istnieć i być pusty przed uruchomieniem polecenia).

Rozdział 11: AIR Debug Launcher (ADL)

Program uruchamiający ADL (AIR Debug Launcher) służy do uruchamiania aplikacji w formacie SWF oraz w formacie HTML podczas programowania. Za pomocą programu ADL można uruchamiać aplikacje bez wcześniejszego pakowania i instalowania. Domyślnie program uruchamiający ADL korzysta ze środowiska wykonawczego dostępnego z SDK, co oznacza, że nie ma konieczności osobnego instalowania środowiska wykonawczego w celu korzystania z ADL.

Program ADL wyświetla instrukcje trace oraz błędy środowiska wykonawczego na standardowym wyjściu, ale nie obsługuje punktów zatrzymań ani innych funkcji debugowania. Debuggera Flash (lub zintegrowanego środowiska programistycznego, takiego jak program Flash Builder) można używać w przypadku złożonych problemów programistycznych.

Uwaga: Jeśli instrukcje `trace()` nie są wyświetlane w konsoli, upewnij się, że w pliku `mm.cfg` nie jest ustawiony parametr `ErrorReportingEnable` ani `TraceOutputFileEnable`. Więcej informacji o lokalizacji tego pliku na poszczególnych platformach zawiera artykuł [Edytowanie pliku mm.cfg](#).

Środowisko AIR obsługuje debugowanie bezpośrednio, dlatego nie jest wymagana wersja środowiska wykonawczego z debuggerem (jak w przypadku programu Adobe® Flash® Player). W celu przeprowadzenia debugowania z wiersza poleceń można użyć debuggera Flash Debugger oraz narzędzia AIR Debug Launcher (ADL).

Debugger Flash Debugger jest dystrybuowany w katalogu Flex SDK. Rodzime wersje, takie jak `fdb.exe` dla systemu Windows, znajdują się w podkatalogu `bin`. Wersja Java znajduje się w podkatalogu `lib`. Program AIR Debug Launcher, `adl.exe`, znajduje się w katalogu `bin` instalacji Flex SDK. (Nie ma osobnej wersji Java).

Uwaga: Aplikacji AIR nie można uruchomić bezpośrednio za pomocą `fdb`, ponieważ `fdb` podejmuje próbę uruchomienia jej w programie Flash Player. Zamiast tego należy zezwolić, aby aplikacja AIR nawiązała połączenie z trwającą sesją `fdb`.

Korzystanie z programu ADL

Aby uruchomić aplikację za pomocą programu ADL, należy użyć następującego schematu:

```
adl application.xml
```

Fragment `application.xml` oznacza plik deskryptora dla aplikacji.

Oto pełny opis składni wywołania programu ADL:

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screenize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

Elementy w nawiasach kwadratowych ([]) są opcjonalne.

-runtime runtime-directory Określa katalog zawierający środowisko wykonawcze, jakie będzie używane. Jeśli nie zostanie określony, wówczas katalog środowiska wykonawczego będzie należał do tego samego folderu SDK, z którego używany jest program ADL. Jeśli program uruchamiający ADL zostanie przeniesiony ze swojego folderu SDK, należy określić katalog środowiska wykonawczego. W systemach Windows i Linux należy określić katalog zawierający katalog `Adobe AIR`. W systemie Mac OS X należy określić katalog zawierający `Adobe AIR.framework`.

-pubid publisher-id Przypisuje określoną wartość jako identyfikator wydawcy uruchamianej aplikacji AIR. Określenie tymczasowego identyfikatora wydawcy umożliwia testowanie funkcji w aplikacji AIR, np. komunikowanie przez połączenie lokalne, które korzysta z id. wydawcy w celu jednoznacznego zidentyfikowania aplikacji. Począwszy od wersji AIR 1.5.3 możliwe jest także określenie identyfikatora wydawcy w pliku deskryptora aplikacji (nie należy wówczas korzystać z tego parametru).

Uwaga: Począwszy od wersji AIR 1.5.3 identyfikator wydawcy nie jest obliczany automatycznie i przypisywany do aplikacji AIR. Identyfikator wydawcy można określić podczas tworzenia aktualizacji istniejącej aplikacji AIR, ale nowe aplikacje nie potrzebują i dla nich nie należy określać identyfikatora wydawcy.

-nodebug Wyłącza obsługę debugowania. Jeśli ten parametr jest używany, wówczas proces aplikacji nie może się połączyć z debuggerem Flash, co powoduje zatrzymanie wyświetlania okien dialogowych z informacjami o nieobsłużonych wyjątkach. (Jednak instrukcje trace nadal są wyświetlane w oknie konsoli). Wyłączenie debugowania umożliwia szybsze działanie aplikacji, a ponadto powoduje wierniejsze emulowanie trybu wykonywania zainstalowanej aplikacji.

-atlogin Symuluje uruchamianie aplikacji przy logowaniu. Ta flaga umożliwia testowanie logiki aplikacji wykonywanej dopiero po zalogowaniu się użytkownika. Gdy używana jest opcja `-atlogin`, właściwość `reason` obiektu `InvokeEvent` wywołanego w aplikacji będzie miała wartość `login`, a nie `standard` (chyba że aplikacja jest już uruchomiona).

-profile profileName Powoduje debugowanie aplikacji przy użyciu określonego profilu. Opcja `profileName` może mieć jedną z następujących wartości:

- `desktop`
- `extendedDesktop`
- `mobileDevice`

Jeśli deskryptor aplikacji zawiera element `supportedProfiles`, profil określany za pomocą opcji `-profile` musi występować na liście obsługiwanych pozycji. Jeśli flaga `-profile` nie została użyta, pierwszy profil w deskrytorze aplikacji jest używany jako aktywny profil. Jeśli deskryptor aplikacji nie zawiera elementu `supportedProfiles` i nie została użyta flaga `-profile`, używany jest profil `desktop`.

Więcej informacji zawierają rozdziały „[supportedProfiles](#)” na stronie 253 i „[Profile urządzeń](#)” na stronie 259.

-screenSize wartość Symulowany rozmiar ekranu używany podczas uruchamiania aplikacji o profilu `mobileDevice` na komputerze stacjonarnym. Należy podać rozmiar ekranu jako predefiniowany typ ekranu lub jako wymiary w pikselach (określając normalną szerokość i wysokość układu pionowego i szerokość oraz wysokość pełnego ekranu). Aby określić wartość jako typ, należy użyć jednego z następujących predefiniowanych typów ekranu.

Typ ekranu	Szerokość x wysokość (normalnie)	Szerokość x wysokość (pełny ekran)
480	720 x 480	720 x 480
720	1280 x 720	1280 x 720
1080	1920 x 1080	1920 x 1080
Droid	480 x 816	480 x 854

Typ ekranu	Szerokość x wysokość (normalnie)	Szerokość x wysokość (pełny ekran)
FWQVGA	240 x 432	240 x 432
FWVGA	480 x 854	480 x 854
HVGA	320 x 480	320 x 480
iPad	768 x 1004	768 x 1024
iPad (Retina)	1536 x 2008	1536 x 2048
iPhone	320 x 460	320 x 480
iPhone (Retina)	640 x 920	640 x 960
iPhone 5 (Retina)	640 x 1096	640 x 1136
iPhone6	750 x 1294	750 x 1334
iPhone6Plus	1242 x 2148	1242 x 2208
iPod	320 x 460	320 x 480
iPod (Retina)	640 x 920	640 x 960
iPod 5 (Retina)	640 x 1096	640 x 1136
Nexus One	480 x 762	480 x 800
QVGA	240 x 320	240 x 320
Samsung Galaxy S	480 x 762	480 x 800
Samsung Galaxy Tab	600 x 986	600 x 1024
WQVGA	240 x 400	240 x 400
WVGA	480 x 800	480 x 800

Aby bezpośrednio określić wymiary ekranu w pikselach, należy użyć następującego formatu:

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Zawsze należy określać wymiary w pikselach układu pionowego — z wartością szerokości mniejszą niż wartość wysokości. Można na przykład zdefiniować ekran telefonu NexusOne w następujący sposób:

```
-screensize 480x762:480x800
```

-extdir extension-directory Katalog, w którym środowisko wykonawcze powinno szukać rozszerzeń natywnych. Ten katalog zawiera odpowiedni podkatalog dla każdego rozszerzenia natywnego używanego przez aplikację. Każdy z tych podkatalogów zawiera *rozpakowany* plik ANE rozszerzenia. Na przykład:


```
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
```

W przypadku używania parametru `-extdir` należy wziąć pod uwagę następujące kwestie:

- Polecenie ADL wymaga, aby każdy z określonych katalogów miał rozszerzenie nazwy pliku `ane`. Jednak część nazwy pliku przed sufiksem „`ane`” może być dowolną prawidłową nazwą pliku. *Nie* musi być ona zgodna z wartością elementu `extensionID` w pliku deskryptora aplikacji.
- Parametr `-extdir` można określić więcej niż raz.
- Zastosowania parametru `-extdir` są różne w narzędziach ADT i ADL. W narzędziu ADT określa on katalog zawierający pliki ANE.
- Katalogi rozszerzeń można również określić za pomocą zmiennej środowiskowej `AIR_EXTENSION_PATH`. Zobacz „Zmienne środowiskowe narzędzia ADT” na stronie 199.

application.xml Plik deskryptora aplikacji. Zobacz „Pliki deskryptora aplikacji AIR” na stronie 217. Deskryptor aplikacji jest jedynym wymaganym parametrem programu ADL i w większości przypadków jedynym potrzebnym parametrem.

root-directory Określa katalog główny aplikacji, która ma być uruchamiana. Jeśli katalog nie jest uruchomiony, zostanie wykorzystany katalog zawierający plik deskryptora aplikacji.

-- arguments Wszystkie znaki, jakie pojawią się za znakami „`--`”, zostaną przekazane do aplikacji jako argumenty wiersza poleceń.

Uwaga: Po uruchomieniu aplikacji AIR, która już działa, nie dochodzi do uruchomienia nowej instancji tej aplikacji. Zamiast tego zdarzenie `invoke` jest wywoływane do działającej aplikacji.

Przykłady ADL

Uruchomienie aplikacji w katalogu bieżącym:

```
adl myApp-app.xml
```

Uruchomienie aplikacji w podkatalogu katalogu bieżącego:

```
adl source/myApp-app.xml release
```

Uruchomienie aplikacji i przekazanie dwóch argumentów wiersza poleceń: „tick” i „tock”.

```
adl myApp-app.xml -- tick tock
```

Uruchomienie aplikacji przy użyciu określonego środowiska wykonawczego:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Uruchomienie aplikacji bez obsługi debugowania:

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

Uruchomienie aplikacji o profilu dla urządzeń przenośnych i zasymulowanie rozmiaru ekranu urządzenia Nexus.

```
adl -profile mobileDevice -screenize NexusOne myMobileApp-app.xml
```

Uruchomienie aplikacji za pomocą programu Apache Ant. (Ścieżki podane w przykładzie dotyczą systemu 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>
```

Kody wyjścia i kody błędów w programie ADL

Poniższa tabela przedstawia kody wyjścia wyświetlane przez program ADL:

Kod wyjścia	Opis
0	Pomyślne uruchomienie. Program ADL wyłącza się po wyłączeniu aplikacji AIR.
1	Pomyślne wywołanie działającej aplikacji AIR. Program ADL wyłącza się natychmiast.
2	Błąd składni. Argumenty wprowadzone do ADL są niepoprawne.
3	Nie można znaleźć środowiska wykonawczego.
4	Nie można uruchomić środowiska wykonawczego. Często przyczyną takiej sytuacji jest niezgodność wersji określonej w aplikacji z wersją środowiska wykonawczego.
5	Błąd z nieznanymi przyczyn.
6	Nie można znaleźć pliku deskryptora aplikacji.
7	Zawartość deskryptora aplikacji jest niepoprawna. Ten błąd zwykle wskazuje, że plik XML nie jest poprawnie sformatowany.
8	Nie można znaleźć głównego pliku zawartości aplikacji (określony w elemencie <content> pliku deskryptora aplikacji).

Kod wyjścia	Opis
9	Główny plik zawartości aplikacji nie jest poprawnym plikiem SWF lub HTML.
10	Aplikacja nie obsługuje profilu określonego za pomocą opcji -profile.
11	Argument -screensize nie jest obsługiwany w bieżącym profilu.

Rozdział 12: Narzędzie ADT

Program ADT (AIR Developer Tool) jest wielozadaniowym narzędziem wiersza poleceń przeznaczonym do programowania aplikacji AIR. Za pomocą narzędzia ADT można wykonywać następujące zadania:

- Pakowanie aplikacji AIR w postaci pliku instalacyjnego air
- Pakowanie aplikacji AIR w postaci instalatora natywnego, na przykład pliku instalatora exe w systemie Windows, pliku ipa w systemie iOS czy też pliku apk w systemie Android
- Pakowanie rozszerzenia natywnego w postaci pliku rozszerzenia natywnego środowiska AIR (ANE, AIR Native Extension)
- Podpisywanie aplikacji AIR za pomocą certyfikatu cyfrowego
- Zmiana (migracja) podpisu cyfrowego używanego do aktualizacji aplikacji
- Określanie urządzeń podłączonych do komputera
- Tworzenie cyfrowego certyfikatu podpisywania kodu z podpisem automatycznym
- Zdalne instalowanie, uruchamianie i odinstalowywanie aplikacji na urządzeniu przenośnym
- Zdalne instalowanie i odinstalowywanie środowiska wykonawczego AIR na urządzeniu przenośnym

Narzędzie ADT jest programem napisanym w języku Java. To narzędzie jest częścią [zestawu SDK środowiska AIR](#). Do używania tego narzędzia jest wymagane środowisko Java 1.5 lub nowsze. Zestaw SDK zawiera skrypt przeznaczony do wywoływania narzędzia ADT. Aby można było używać tego skryptu, lokalizacja programu Java musi być podana w zmiennej środowiskowej ścieżek. Jeśli w zmiennej środowiskowej ścieżek jest również podany katalog `bin` zestawu SDK środowiska AIR, to w celu wywołania narzędzia ADT można wpisać w wierszu poleceń polecenie `adt` z odpowiednimi argumentami. (Informacje na temat konfigurowania zmiennej środowiskowej ścieżek można znaleźć w dokumentacji systemu operacyjnego. W celu udostępnienia dodatkowej pomocy opisano procedury konfigurowania ścieżki na większości komputerów w sekcji „[Zmienne środowiskowe ścieżek](#)” na stronie 320).

Do korzystania z narzędzia ADT jest wymagane co najmniej 2 GB pamięci komputera. W przypadku mniejszej ilości programowi ADT może zabraknąć pamięci, szczególnie podczas pakowania aplikacji dla systemu iOS.

Zakładając, że w zmiennej ścieżki znajdują się katalogi `bin` zarówno środowiska Java, jak i zestawu SDK środowiska AIR, narzędzie ADT można uruchamiać przy użyciu następującej składni:

```
adt -command options
```

Uwaga: Większość zintegrowanych środowisk programistycznych, takich jak *Adobe Flash Builder* i *Adobe Flash Professional*, może pakować i podpisywać aplikacje AIR za użytkownika. Gdy jest używane takie środowisko programistyczne, w celu wykonania typowych zadań nie jest zwykle konieczne korzystanie z narzędzia ADT. Jednak nadal konieczne może być korzystanie z narzędzia ADT jako narzędzia wiersza poleceń w przypadku funkcji, które nie są obsługiwane przez zintegrowane środowisko programistyczne. Ponadto narzędzia ADT można używać jako narzędzia wiersza poleceń w ramach automatycznego procesu tworzenia.

Polecenia narzędzia ADT

Pierwszy argument przekazywany do narzędzia ADT określa jedno z poniższych poleceń.

- Argument `-package` powoduje spakowanie aplikacji AIR w postaci natywnego rozszerzenia środowiska AIR (ANE, AIR Native Extension).

- Argument `-prepare` powoduje spakowanie aplikacji AIR w postaci pliku pośredniego (AIRI), ale nie następuje podpisanie tego pliku. Plików AIRI nie można instalować.
- Argument `-sign` powoduje podpisanie pakietu AIRI utworzonego za pomocą polecenia `-prepare`. Polecenia `-prepare` i `-sign` pozwalają na przeprowadzanie pakowania i podpisywania w różnych momentach. Do podpisania lub ponownego podpisania pakietu AIRI można również użyć polecenia `-sign`.
- Argument `-migrate` powoduje zastosowanie podpisu migracji względem podpisanego pakietu AIR, dzięki czemu można używać nowego lub odnowionego certyfikatu podpisywania kodu.
- Argument `-certificate` powoduje utworzenie cyfrowego certyfikatu podpisywania kodu z podpisem automatycznym.
- Argument `-checkstore` powoduje sprawdzenie, czy można uzyskać dostęp do certyfikatu cyfrowego w magazynie kluczy.
- Argument `-installApp` powoduje zainstalowanie aplikacji AIR na urządzeniu lub w emulatorze urządzenia.
- Argument `-launchApp` powoduje uruchomienie aplikacji AIR na urządzeniu lub w emulatorze urządzenia.
- Argument `-appVersion` powoduje podanie informacji o wersji aplikacji AIR, która jest obecnie zainstalowana na urządzeniu lub w emulatorze urządzenia.
- Argument `-uninstallApp` powoduje odinstalowanie aplikacji AIR z urządzenia lub z emulatora urządzenia.
- Argument `-installRuntime` powoduje zainstalowanie środowiska wykonawczego AIR na urządzeniu lub w emulatorze urządzenia.
- Argument `-runtimeVersion` powoduje podanie informacji o wersji środowiska wykonawczego AIR, która jest obecnie zainstalowana na urządzeniu lub w emulatorze urządzenia.
- Argument `-uninstallRuntime` powoduje odinstalowanie środowiska wykonawczego AIR, które jest obecnie zainstalowane na urządzeniu lub w emulatorze urządzenia.
- Argument `-version` powoduje podanie informacji o numerze wersji narzędzia ADT.
- Argument `-devices` powoduje zgłoszenie informacji o podłączonych urządzeniach przenośnych lub emulatorach.
- Argument `-help` powoduje wyświetlenie listy poleceń i opcji.

Wiele poleceń narzędzia ADT korzysta z tych samych zestawów flag opcji i parametrów. Te zestawy opcji są opisane szczegółowo oddzielnie:

- „[Opcje podpisywania kodu ADT](#)” na stronie 190
- „[Opcje plików i ścieżek](#)” na stronie 192
- „[Opcje połączenia debugera](#)” na stronie 194
- „[Opcje rozszerzeń natywnych](#)” na stronie 194

Polecenie `package` narzędzia ADT

Polecenie `-package` powinno być uruchamiane z poziomu głównego katalogu aplikacji. W tym poleceniu jest stosowana następująca składnia:

Aby utworzyć pakiet AIR ze składowych plików aplikacji:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Aby utworzyć pakiet natywny ze składowych plików aplikacji:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Aby utworzyć pakiet natywny zawierający rozszerzenie natywne ze składowych plików aplikacji:

```
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
```

Aby utworzyć pakiet natywny z pliku AIR lub AIRI:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Aby utworzyć pakiet rozszerzenia natywnego ze składowych plików rozszerzenia natywnego:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Uwaga: Nie trzeba podpisywać pliku ANE, więc parametry `AIR_SIGNING_OPTIONS` są opcjonalne w tym przykładzie.

AIR_SIGNING_OPTIONS Opcje podpisywania środowiska AIR określają certyfikat używany do podpisywania pliku instalacyjnego środowiska AIR. Opcje podpisywania zostały w pełni opisane w sekcji „[Opcje podpisywania kodu ADT](#)” na stronie 190.

-migrate Ta flaga powoduje podpisanie aplikacji przy użyciu certyfikatu migracji dodatkowo do certyfikatu określonego w parametrach `AIR_SIGNING_OPTIONS`. Ta flaga jest prawidłowa tylko w przypadku pakowania aplikacji komputerowej jako instalatora natywnego, gdy aplikacja korzysta z rozszerzenia natywnego. W innych sytuacjach występuje błąd. Opcje podpisywania dotyczące certyfikatu migracji są określone przez parametry `MIGRATION_SIGNING_OPTIONS`. Te opcje podpisywania są w pełni opisane w rozdziale „[Opcje podpisywania kodu ADT](#)” na stronie 190. Za pomocą flagi `-migrate` można utworzyć aktualizację dla aplikacji komputerowej z instalatorem natywnym, która korzysta z rozszerzenia natywnego, a także zmienić certyfikat podpisu kodu tej aplikacji (na przykład w razie wygaśnięcia oryginalnego certyfikatu). Więcej informacji można znaleźć w rozdziale „[Podpisywanie zaktualizowanej wersji aplikacji AIR](#)” na stronie 211.

Flaga `-migrate` polecenia `-package` jest dostępna w środowisku AIR 3.6 lub nowszym.

-target Typ pakietu, który ma zostać utworzony. Obsługiwane typy pakietów:

- `air` — pakiet AIR. Wartość „`air`” jest wartością domyślną, a flaga `-target` nie musi zostać określona podczas tworzenia plików AIR i AIRL.
- `airn` — macierzysty pakiet aplikacji przeznaczony dla urządzeń w rozszerzonym profilu telewizyjnym.
- `ane` — pakiet natywnego rozszerzenia środowiska AIR.
- Elementy docelowe pakietu Android:
 - `apk` — pakiet systemu Android. Pakiet utworzony w tym formacie docelowym można zainstalować wyłącznie na urządzeniu z systemem Android, a nie w emulatorze.
 - `apk-captive-runtime` — pakiet systemu Android zawierający zarówno aplikację, jak i dołączoną wersję środowiska wykonawczego AIR. Pakiet utworzony w tym formacie docelowym można zainstalować wyłącznie na urządzeniu z systemem Android, a nie w emulatorze.
 - `apk-debug` — pakiet systemu Android z dodatkowymi informacjami o debugowaniu. (Pliki SWF w aplikacji muszą być również skompilowane z włączoną obsługą debugowania).
 - `apk-emulator` — pakiet systemu Android przeznaczony do użytku w emulatorze bez obsługi debugowania. (W celu umożliwienia debugowania zarówno w emulatorach, jak i na urządzeniach należy użyć formatu docelowego `apk-debug`).
 - `apk-profile` — pakiet systemu Android, który obsługuje profilowanie pamięci i wydajności aplikacji.
- Elementy docelowe pakietu iOS:
 - `ipa-ad-hoc` — pakiet systemu iOS przeznaczony do szybkiego rozpowszechniania.
 - `ipa-app-store` — pakiet systemu iOS przeznaczony do rozpowszechniania w sklepie App Store firmy Apple.
 - `ipa-debug` — pakiet systemu iOS z dodatkowymi informacjami o debugowaniu. (Pliki SWF w aplikacji muszą być również skompilowane z włączoną obsługą debugowania).
 - `ipa-test` — pakiet systemu iOS skompilowany bez informacji o optymalizacji czy debugowaniu.
 - `ipa-debug-interpreter` — odpowiednik pakietu debugowania oferujący podobne funkcje oraz szybszą kompilację. Kod bajtowy ActionScript jest interpretowany, a nie przekształcany w kod maszynowy. Oznacza to, że wykonywanie kodu przebiega wolniej w przypadku pakietu z interpreterem.
 - `ipa-debug-interpreter-simulator` — działa analogicznie do opcji `ipa-debug-interpreter`, ale pakiet jest przygotowywany do symulatora systemu iOS. Tylko dla komputerów Macintosh. W przypadku użycia tej opcji należy również dodać opcję `-platformsdk` określającą ścieżkę do zestawu SDK symulatora systemu iOS.
 - `ipa-test-interpreter` — odpowiednik pakietu testowania oferujący podobne funkcje oraz szybszą kompilację. Kod bajtowy ActionScript jest interpretowany, a nie przekształcany w kod maszynowy. Oznacza to, że wykonywanie kodu przebiega wolniej w przypadku pakietu z interpreterem.

- `ipa-test-interpreter-simulator` — działa analogicznie do opcji `ipa-test-interpreter`, ale pakiet jest przygotowywany dla symulatora systemu iOS. Tylko dla komputerów Macintosh. W przypadku użycia tej opcji należy również dodać opcję `-platformsdk` określającą ścieżkę do zestawu SDK symulatora systemu iOS.
- `native` — macierzysty instalator komputera stacjonarnego. Plik jest tworzony przy użyciu typu będącego natywnym formatem instalacyjnym systemu operacyjnego, w którym jest wykonywane polecenie:
 - EXE — Windows
 - DMG — Mac
 - DEB — Ubuntu Linux (AIR 2.6 i starsze wersje)
 - RPM — Fedora albo OpenSuse Linux (AIR 2.6 i starsze wersje)

Więcej informacji zawiera rozdział „[Pakowanie instalatora natywnego dla komputerów](#)” na stronie 59.

-sampler (Tylko w systemie iOS i środowisku AIR 3.4 lub nowszym) Umożliwia użycie w aplikacjach systemu iOS opartego na telemetrii próbnika języka ActionScript. Użycie tej flagi pozwala na określenie profilu aplikacji za pomocą oprogramowania Adobe Scout. Program [Scout](#) pozwala na utworzenie profilu dowolnej zawartości dla platformy Flash, ale włączenie szczegółowej telemetrii umożliwia dogłębną analizę czasu wywoływania funkcji języka ActionScript, listy wyświetlania, renderowania obiektu Stage3D i innych elementów. Uwaga: Użycie tej flagi może spowodować nieznaczny spadek wydajności. Nie należy jej stosować w aplikacjach produkcyjnych.

-hideAneLibSymbols (tylko w systemie iOS i środowisku AIR 3.4 lub nowszym) Programiści aplikacji mają możliwość używania wielu rozszerzeń natywnych z wielu źródeł. Jeśli pliki ANE mają wspólną nazwę symboli, program ADT wyświetla komunikat o błędzie powielonego symbolu. W niektórych przypadkach błąd ten może nawet spowodować awarię programu w środowisku wykonawczym. Za pomocą opcji `hideAneLibSymbols` można określić, czy symbole biblioteki ANE mają być widoczne tylko dla źródeł danej biblioteki (parametr `yes`), czy mają być widoczne globalnie (parametr `no`).

- **yes** — Ukrywa wszystkie symbole ANE, co pozwala rozwiązać niezamierzone problemy z konfliktem symboli.
- **no** — (Parametr domyślny) Nie ukrywa symboli ANE. Jest to zachowanie występujące w środowisku AIR w wersji starszej niż 3.4.

Opcja **-embedBitcode** (tylko dla systemu iOS, środowisko AIR w wersji 25 lub nowszej) Za pomocą opcji `embedBitcode` programiści aplikacji mogą określić, czy kod bitowy ma być osadzony w ich aplikacji dla systemu iOS, wybierając wariant tak lub nie. Wartość domyślna przełącznika to nie.

DEBUGGER_CONNECTION_OPTIONS Opcje połączenia debugera określają, czy pakiet debugowania powinien próbować połączyć się ze zdalnym debugerem uruchomionym na innym komputerze, czy wykrywać połączenie nawiązywane przez zdalny debuger. Ten zestaw opcji jest obsługiwany wyłącznie w przypadku pakietów debugowania dla urządzeń przenośnych (formaty docelowe `apk-debug` i `ipa-debug`). Te opcje są opisane w rozdziale „[Opcje połączenia debugera](#)” na stronie 194.

-airDownloadURL Określa alternatywny adres URL przeznaczony do pobierania i instalowania środowiska wykonawczego AIR na urządzeniach z systemem Android. Jeśli ta opcja nie została określona i nie zainstalowano jeszcze środowiska wykonawczego AIR, aplikacja AIR przekieruje użytkownika do środowiska wykonawczego w sklepie Android Market.

Jeśli aplikacja jest rozpowszechniana za pośrednictwem sklepu innego niż sklep Android Market zarządzany przez firmę Google, wówczas może być konieczne określenie adresu URL w celu pobrania środowiska wykonawczego AIR z tego sklepu. Niektóre alternatywne sklepy nie pozwalają aplikacjom na pobieranie spoza danego sklepu. Ta opcja jest obsługiwana wyłącznie w przypadku pakietów Android.

NATIVE_SIGNING_OPTIONS Opcje podpisywania natywnego określają certyfikat użyty do podpisania pliku pakietu natywnego. Te opcje podpisywania są stosowane w celu zastosowania podpisu używanego przez macierzysty system operacyjny, a nie przez środowisko wykonawcze AIR. Poza tym te opcje są takie same jak opcje AIR_SIGNING_OPTIONS. Dokładnie opisano je w sekcji „[Opcje podpisywania kodu ADT](#)” na stronie 190.

Podpisy macierzyste są obsługiwane zarówno w systemie Windows, jak i Android. W systemie Windows należy określić zarówno opcje podpisywania środowiska AIR, jak i opcje podpisywania macierzystego. W systemie Android można określić wyłącznie opcje podpisywania macierzystego.

W wielu przypadkach można używać tego samego certyfikatu podpisywania do zastosowania zarówno podpisu środowiska AIR, jak i podpisu macierzystego. Jednak nie zawsze jest to prawdą. Na przykład według zasad firmy Google dla aplikacji przesyłanych do sklepu Android Market wszystkie takie aplikacje muszą być podpisane za pomocą certyfikatu ważnego co najmniej do roku 2033. Oznacza to, że do podpisywania aplikacji Android nie powinien być używany certyfikat wystawiony przez dobrze znany ośrodek certyfikacji, wybrany z ośrodków zalecanych w przypadku stosowania podpisu środowiska AIR. (Żadne ośrodki certyfikacji nie wystawiają certyfikatu podpisywania kodu o tak długim okresie ważności).

output Nazwa pliku pakietu, który ma zostać utworzony. Określenie rozszerzenia pliku jest opcjonalne. Jeśli nie zostało ono określone, zostaje dodane rozszerzenie odpowiednie dla wartości `-target` i aktualnego systemu operacyjnego.

app_descriptor Ścieżka do pliku deskryptora aplikacji. Ścieżka może zostać określona względem katalogu bieżącego lub jako ścieżka bezwzględna. (W pliku AIR nazwa pliku deskryptora aplikacji zostaje zmieniona na *application.xml*).

-platformsdk — ścieżka platformy zestawu SDK dla urządzenia docelowego:

- Android — zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową AIR_ANDROID_SDK_HOME. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).
- iOS — zestaw SDK środowiska AIR jest dostarczany w pakiecie z zestawem SDK dla systemu iOS. Opcja `-platformsdk` umożliwia pakowanie aplikacji z zewnętrznym zestawem SDK, aby wyeliminować ograniczenie związane z używaniem spakowanego razem z aplikacją zestawu SDK dla systemu iOS. Jeśli na przykład opracowano rozszerzenie przy użyciu najnowszego zestawu SDK dla systemu iOS, można wskazać ten zestaw SDK podczas pakowania aplikacji. Oprócz tego w przypadku używania narzędzia ADT z symulatorem systemu iOS należy zawsze dołączyć opcję `-platformsdk`, aby podać ścieżkę do zestawu SDK symulatora systemu iOS.

-arch Programista aplikacji może za pomocą tego argumentu utworzyć pakiet APK dla platform x86. Są dostępne dwie wartości:

- `armv7` — Program ADT pakuje plik APK dla platformy Android `armv7`.
- `x86` — Program ADT pakuje plik APK dla platformy Android `x86`.

Gdy nie jest określona żadna wartość, jest domyślnie przyjmowane ustawienie `armv7`.

FILE_OPTIONS Określa pliki aplikacji, które mają zostać dołączone do pakietu. Opcje plików są szczegółowo opisane w sekcji „[Opcje plików i ścieżek](#)” na stronie 192. Podczas tworzenia macierzystego pakietu z pliku AIR lub AIRI nie należy określać opcji plików.

input_airi Ten element należy określić w przypadku tworzenia pakietu natywnego z pliku AIRI. Opcje AIR_SIGNING_OPTIONS są wymagane, jeśli format docelowy to *air* (lub nie określono żadnego formatu docelowego).

input_air Ten element należy określić w przypadku tworzenia pakietu natywnego z pliku AIR. Nie należy określać opcji AIR_SIGNING_OPTIONS.

ANE_OPTIONS Określa opcje i pliki przeznaczone do tworzenia pakietu rozszerzenia natywnego. Opcje pakietu rozszerzenia są w pełni opisane w sekcji „[Opcje rozszerzeń natywnych](#)” na stronie 194.

Przykłady poleceń dotyczących pakietu ADT

Pakowanie plików aplikacji w bieżącym katalogu dla aplikacji AIR opartych na SWF:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Pakowanie plików aplikacji w bieżącym katalogu dla aplikacji AIR opartych na HTML:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js image.gif
```

Pakowanie wszystkich plików i podkatalogów w bieżącym katalogu roboczym:

```
adt -package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Uwaga: Plik magazynu kluczy zawiera klucz prywatny używany do podpisania aplikacji. Nigdy nie należy umieszczać certyfikatu podpisującego w pakiecie AIR! Jeśli w komendzie ADT używane są znaki wieloznaczne, plik magazynu kluczy należy umieścić w innej lokalizacji, tak aby nie został dołączony do pakietu. W tym przykładzie plik magazynu kluczy cert.p12 znajduje się w katalogu nadrzędnym.

Pakowanie tylko plików głównych i podkatalogu images:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Pakowanie aplikacji HTML oraz wszystkich plików HTML, skryptów i podkatalogów images:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRAliases.js html scripts images
```

Pakowanie pliku application.xml oraz głównego pliku SWF znajdującego się w katalogu roboczym(release/bin):

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml -C release/bin myApp.swf
```

Pakowanie zasobów z więcej niż jednego miejsca w systemie plików kompilacji. W tym przykładzie przed pakowaniem zasoby aplikacji znajdują się następujących folderach:

```
/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
```

Uruchomienie poniższego polecenia ADT z katalogu /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
```

Powoduje utworzenie następującej struktury pakietu:

```
/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
```

Uruchamianie ADT jako programu Java dla prostej aplikacji opartej na SWF (bez ustawiania ścieżki klasy):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Uruchamianie ADT jako programu Java dla prostej aplikacji opartej na HTML (bez ustawiania ścieżki klasy):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Uruchamianie ADT jako programu Java (ze ścieżką klasy Java ustawioną w taki sposób, aby zawierała pakiet ADT.jar):

```
java -com.adobe.air.ADT -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

Uruchom narzędzie ADT jako zadanie Java za pomocą biblioteki Apache Ant. (Przeważnie optymalne jest uruchomienie polecenia narzędzia ADT bezpośrednio w skryptach biblioteki Ant). Ścieżki podane w poniższym przykładzie dotyczą systemu 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>
```

Uwaga: W niektórych systemach znaki dwubajtowe w ścieżkach systemu plików mogą zostać nieprawidłowo zinterpretowane. W takiej sytuacji należy spróbować skonfigurować środowisko JRE używane do uruchamiania narzędzia ADT w taki sposób, aby był stosowany zestaw znaków UTF-8. Domyślnie służy do tego skrypt używany do uruchamiania narzędzia ADT na komputerach Mac i na komputerach z systemem Linux. Opcję - `Dfile.encoding=UTF-8` należy określić w wierszu poleceń środowiska Java w pliku `adt.bat` systemu Windows lub podczas uruchamiania narzędzia ADT bezpośrednio w środowisku Java.

Polecenie `prepare` narzędzia ADT

Polecenie `-prepare` służy do tworzenia niepodpisanego pakietu AIRI. Pakietu AIRI nie można używać samodzielnie. Do konwertowania pliku AIRI na podpisany plik AIR należy używać polecenia `-sign`, a do konwertowania pliku AIRI na pakiet natywny należy używać polecenia `package`.

Dla polecenia `-prepare` jest stosowana następująca składnia:

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output Nazwa tworzonego pliku AIRI.

app_descriptor Ścieżka do pliku deskryptora aplikacji. Ścieżka może zostać określona względem katalogu bieżącego lub jako ścieżka bezwzględna. (W pliku AIR nazwa pliku deskryptora aplikacji zostaje zmieniona na *application.xml*).

FILE_OPTIONS Określa pliki aplikacji, które mają zostać dołączone do pakietu. Opcje plików są szczegółowo opisane w sekcji „[Opcje plików i ścieżek](#)” na stronie 192.

Polecenie `sign` narzędzia ADT

Polecenie `-sign` służy do podpisywania plików AIRI i ANE.

Dla polecenia `-sign` jest stosowana następująca składnia:

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Opcje podpisywania środowiska AIR określają certyfikat użyty do podpisania pliku pakietu. Opcje podpisywania zostały w pełni opisane w sekcji „[Opcje podpisywania kodu ADT](#)” na stronie 190.

input Nazwa pliku AIRI lub ANE do podpisania.

output Nazwa podpisanego pakietu, który ma zostać utworzony.

Jeśli plik ANE jest już podpisany, stary podpis zostaje odrzucony. Nie można ponownie podpisywać plików AIR. Jeśli do aktualizacji aplikacji ma zostać użyty nowy podpis, należy użyć polecenia `-migrate`.

Polecenie migrate narzędzia ADT

Polecenie `-migrate` powoduje zastosowanie podpisu migracji do pliku AIR. Podpis migracji musi być używany podczas odnawiania lub zmieniania certyfikatu cyfrowego, gdy muszą zostać zaktualizowane aplikacje podpisane przy użyciu starego certyfikatu.

Więcej informacji o pakowaniu aplikacji AIR przy użyciu podpisu migracji można znaleźć w rozdziale „[Podpisywanie zaktualizowanej wersji aplikacji AIR](#)” na stronie 211.

Uwaga: Certyfikat migracji należy zastosować w ciągu 365 dni od terminu wygaśnięcia certyfikatu. Po upływie tego okresu prolongaty aktualizacje aplikacji nie mogą już być podpisywane przy użyciu podpisu migracji. Użytkownicy mogą najpierw przeprowadzić aktualizację do wersji aplikacji, która została podpisana przy użyciu podpisu migracji, a następnie zainstalować najnowszą aktualizację. Mogą także odinstalować oryginalną aplikację i zainstalować nowy pakiet AIR.

Aby użyć podpisu migracji, należy najpierw podpisać aplikację AIR przy użyciu nowego lub odnowionego certyfikatu (za pomocą poleceń `-package` lub `-sign`), a następnie zastosować podpis migracji za pomocą starego certyfikatu i polecenia `-migrate`.

Dla polecenia `-migrate` jest stosowana następująca składnia:

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Opcje podpisywania środowiska AIR określające oryginalny certyfikat, który został użyty do podpisania istniejących wersji aplikacji AIR. Opcje podpisywania zostały w pełni opisane w sekcji „[Opcje podpisywania kodu ADT](#)” na stronie 190.

input Plik AIR, który został już podpisany przy użyciu NOWEGO certyfikatu aplikacji.

output Nazwa ostatecznego pakietu oznaczonego podpisami zarówno z nowego, jak i ze starego certyfikatu.

Nazwy plików używanych dla plików wejściowych i wyjściowych AIR muszą się różnić.

Uwaga: Polecenia `migrate` programu ADT nie można używać z aplikacjami komputerowymi AIR, które zawierają rozszerzenia natywne, ponieważ takie aplikacje są spakowane jako instalatory natywne, a nie jako pliki AIR. Aby zmienić certyfikaty aplikacji komputerowej AIR, która zawiera rozszerzenie natywne, należy spakować aplikację przy użyciu „[Polecenie package narzędzia ADT](#)” na stronie 176 z flagą `-migrate`.

Polecenie checkstore narzędzia ADT

Polecenie `-checkstore` pozwala zweryfikować ważność magazynu kluczy. Dla tego polecenia jest stosowana następująca składnia:

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS Opcje podpisywania określające magazyn kluczy, który ma zostać zweryfikowany. Opcje podpisywania zostały w pełni opisane w sekcji „[Opcje podpisywania kodu ADT](#)” na stronie 190.

Polecenie certificate narzędzia ADT

Polecenie `-certificate` pozwala utworzyć cyfrowy certyfikat podpisywania kodu z podpisem automatycznym. Dla tego polecenia jest stosowana następująca składnia:

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type  
output password
```

-cn Ciąg znaków przypisany jako wspólna nazwa nowego certyfikatu.

-ou Ciąg znaków przypisany jako jednostka organizacyjna wydająca certyfikat. (Opcjonalnie).

-o Ciąg znaków przypisany jako organizacja wydająca certyfikat. (Opcjonalnie).

-c Dwucyfrowy kod kraju ISO-3166. Certyfikat nie zostanie wygenerowany, jeśli zostanie wprowadzony niepoprawny kod. (Opcjonalnie).

-validityPeriod Liczba lat, przez które certyfikat będzie obowiązywać. Jeśli nie zostanie określona, certyfikat zostanie ustawiony jako ważny przez pięć lat. (Opcjonalnie).

key_type Typ klucza do użycia na potrzeby certyfikatu to *2048-RSA*.

output Ścieżka i nazwa pliku certyfikatu, który ma zostać utworzony.

password Hasło dostępu dla nowego certyfikatu. Hasło jest wymagane podczas podpisywania plików AIR tym certyfikatem.

Polecenie `installApp` narzędzia ADT

Polecenie `installApp` powoduje zainstalowanie aplikacji na urządzeniu lub w emulatorze.

Przed ponowną instalacją za pomocą tego polecenia należy odinstalować istniejącą aplikację.

Dla tego polecenia jest stosowana następująca składnia:

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package  
fileName
```

-platform Nazwa platformy urządzenia. Podaj opcję *ios* lub *android*.

-platformsdk — ścieżka platformy zestawu SDK dla urządzenia docelowego (opcjonalna):

- **Android** — zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową `AIR_ANDROID_SDK_HOME`. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).
- **iOS** — zestaw SDK środowiska AIR jest dostarczany w pakiecie z zestawem SDK dla systemu iOS. Opcja `-platformsdk` umożliwia pakowanie aplikacji z zewnętrznym zestawem SDK, aby wyeliminować ograniczenie związane z używaniem spakowanego razem z aplikacją zestawu SDK dla systemu iOS. Jeśli na przykład opracowano rozszerzenie przy użyciu najnowszego zestawu SDK dla systemu iOS, można wskazać ten zestaw SDK podczas pakowania aplikacji. Oprócz tego w przypadku używania narzędzia ADT z symulatorem systemu iOS należy zawsze dołączyć opcję `-platformsdk`, aby podać ścieżkę do zestawu SDK symulatora systemu iOS.

Opcja **-device** określa wartość *ios_simulator*, numer seryjny (Android) lub uchwyt (iOS) podłączonego urządzenia. W systemie iOS ten parametr jest wymagany. W systemie Android ten parametr należy określić tylko wtedy, gdy do komputera jest podłączonych więcej działających urządzeń lub emulatorów systemu Android. Jeśli określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia (Android) lub Określono nieprawidłowe urządzenie (iOS). Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

Uwaga: Instalacja pliku IPA bezpośrednio w urządzeniu iOS jest możliwa w środowisku AIR 3.4 lub nowszym i wymaga programu iTunes 10.5.0 lub nowszego.

Za pomocą polecenia `adt -devices` (dostępnego w środowisku AIR 3.4 lub nowszym) można określić uchwyt lub numer seryjny podłączonego urządzenia. W systemie iOS zamiast identyfikatora UUID jest używany uchwyt. Więcej informacji zawiera rozdział „[Polecenie devices programu ADT](#)” na stronie 189.

Ponadto w systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów:

```
adb devices
```

-package — nazwa pliku pakietu, który ma zostać zainstalowany. W systemie iOS musi to być plik IPA. W systemie Android musi to być pakiet APK. Jeśli zainstalowano już określony pakiet, narzędzie ADT zwróci kod błędu 14: Błąd urządzenia.

Polecenie appVersion narzędzia ADT

Polecenie `-appVersion` podaje informacje o wersji aplikacji zainstalowanej na urządzeniu lub w emulatorze. Dla tego polecenia jest stosowana następująca składnia:

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Nazwa platformy urządzenia. Podaj opcję *ios* lub *android*.

-platformsdk — ścieżka platformy zestawu SDK dla urządzenia docelowego:

- **Android** — zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową `AIR_ANDROID_SDK_HOME`. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).
- **iOS** — zestaw SDK środowiska AIR jest dostarczany w pakiecie z zestawem SDK dla systemu iOS. Opcja `-platformsdk` umożliwia pakowanie aplikacji z zewnętrznym zestawem SDK, aby wyeliminować ograniczenie związane z używaniem spakowanego razem z aplikacją zestawu SDK dla systemu iOS. Jeśli na przykład opracowano rozszerzenie przy użyciu najnowszego zestawu SDK dla systemu iOS, można wskazać ten zestaw SDK podczas pakowania aplikacji. Oprócz tego w przypadku używania narzędzia ADT z symulatorem systemu iOS należy zawsze dołączyć opcję `-platformsdk`, aby podać ścieżkę do zestawu SDK symulatora systemu iOS.

-device — podaj opcję `ios_simulator` lub numer seryjny urządzenia. Określenie urządzenia jest wymagane tylko wtedy, gdy do komputera jest podłączone więcej niż jedno działające urządzenie z systemem Android lub jest podłączony i uruchomiony więcej niż jeden emulator. Jeśli określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia. Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

W systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów.

```
adb devices
```

-appid Identyfikator aplikacji AIR zainstalowanej aplikacji. Jeśli na urządzeniu nie zainstalowano aplikacji o określonym identyfikatorze, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia.

Polecenie launchApp narzędzia ADT

Polecenie `-launchApp` powoduje uruchomienie zainstalowanej aplikacji na urządzeniu lub w emulatorze. Dla tego polecenia jest stosowana następująca składnia:

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Nazwa platformy urządzenia. Podaj opcję *ios* lub *android*.

-platformsdk — ścieżka platformy zestawu SDK dla urządzenia docelowego:

- **Android** — zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową `AIR_ANDROID_SDK_HOME`. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).
- **iOS** — zestaw SDK środowiska AIR jest dostarczany w pakiecie z zestawem SDK dla systemu iOS. Opcja `-platformsdk` umożliwia pakowanie aplikacji z zewnętrznym zestawem SDK, aby wyeliminować ograniczenie związane z używaniem spakowanego razem z aplikacją zestawu SDK dla systemu iOS. Jeśli na przykład opracowano rozszerzenie przy użyciu najnowszego zestawu SDK dla systemu iOS, można wskazać ten zestaw SDK podczas pakowania aplikacji. Oprócz tego w przypadku używania narzędzia ADT z symulatorem systemu iOS należy zawsze dołączyć opcję `-platformsdk`, aby podać ścieżkę do zestawu SDK symulatora systemu iOS.

-device — podaj opcję `ios_simulator` lub numer seryjny urządzenia. Określenie urządzenia jest wymagane tylko wtedy, gdy do komputera jest podłączone więcej niż jedno działające urządzenie z systemem Android lub jest podłączony i uruchomiony więcej niż jeden emulator. Jeśli określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia. Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

W systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów.

```
adb devices
```

-appid Identyfikator aplikacji AIR zainstalowanej aplikacji. Jeśli na urządzeniu nie zainstalowano aplikacji o określonym identyfikatorze, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia.

Polecenie `uninstallApp` narzędzia ADT

Polecenie `-uninstallApp` powoduje całkowite usunięcie aplikacji zainstalowanej na zdalnym urządzeniu lub w zdalnym emulatorze. Dla tego polecenia jest stosowana następująca składnia:

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Nazwa platformy urządzenia. Podaj opcję *ios* lub *android*.

-platformsdk — ścieżka platformy zestawu SDK dla urządzenia docelowego:

- **Android** — zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową `AIR_ANDROID_SDK_HOME`. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).
- **iOS** — zestaw SDK środowiska AIR jest dostarczany w pakiecie z zestawem SDK dla systemu iOS. Opcja `-platformsdk` umożliwia pakowanie aplikacji z zewnętrznym zestawem SDK, aby wyeliminować ograniczenie związane z używaniem spakowanego razem z aplikacją zestawu SDK dla systemu iOS. Jeśli na przykład opracowano rozszerzenie przy użyciu najnowszego zestawu SDK dla systemu iOS, można wskazać ten zestaw SDK podczas pakowania aplikacji. Oprócz tego w przypadku używania narzędzia ADT z symulatorem systemu iOS należy zawsze dołączyć opcję `-platformsdk`, aby podać ścieżkę do zestawu SDK symulatora systemu iOS.

-device — podaj opcję *ios_simulator* lub numer seryjny urządzenia. Określenie urządzenia jest wymagane tylko wtedy, gdy do komputera jest podłączone więcej niż jedno działające urządzenie z systemem Android lub jest podłączony i uruchomiony więcej niż jeden emulator. Jeśli określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia. Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

W systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów.

```
adb devices
```

-appid Identyfikator aplikacji AIR zainstalowanej aplikacji. Jeśli na urządzeniu nie zainstalowano aplikacji o określonym identyfikatorze, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia.

Polecenie `installRuntime` narzędzia ADT

Polecenie `-installRuntime` powoduje zainstalowanie na urządzeniu środowiska wykonawczego AIR.

Przed ponownym zainstalowaniem środowiska wykonawczego AIR za pomocą tego polecenia należy odinstalować istniejącą wersję środowiska.

Dla tego polecenia jest stosowana następująca składnia:

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package fileName
```

-platform Nazwa platformy urządzenia. W chwili obecnej to polecenie jest obsługiwane wyłącznie na platformie Android. Należy użyć nazwy *android*.

-platformsdk Ścieżka platformy zestawu SDK dla urządzenia docelowego. Obecnie jedyny obsługiwany zestaw SDK jest przeznaczony dla systemu Android. Zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową `AIR_ANDROID_SDK_HOME`. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).

-device Numer seryjny urządzenia. Urządzenie musi zostać określone wyłącznie wówczas, gdy do komputera jest podłączone więcej niż jedno działające urządzenie lub jest podłączony i uruchomiony więcej niż jeden emulator. Jeśli określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia. Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

W systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów.

```
adb devices
```

-package Nazwa pliku środowiska wykonawczego, który ma zostać zainstalowany. W systemie Android musi to być pakiet APK. Jeśli nie określono żadnego pakietu, odpowiednie środowisko wykonawcze dla urządzenia lub emulatora zostanie wybrane z dostępnych w zestawie SDK środowiska AIR. Jeśli zainstalowano już środowisko wykonawcze, narzędzie ADT zwróci kod błędu 14: Błąd urządzenia.

Polecenie `runtimeVersion` narzędzia ADT

Polecenie `-runtimeVersion` podaje informacje o wersji środowiska wykonawczego AIR zainstalowanej na urządzeniu lub w emulatorze. Dla tego polecenia jest stosowana następująca składnia:

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Nazwa platformy urządzenia. W chwili obecnej to polecenie jest obsługiwane wyłącznie na platformie Android. Należy użyć nazwy *android*.

-platformsdk Ścieżka platformy zestawu SDK dla urządzenia docelowego. Obecnie jedyny obsługiwany zestaw SDK jest przeznaczony dla systemu Android. Zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową AIR_ANDROID_SDK_HOME. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).

-device Numer seryjny urządzenia. Urządzenie musi zostać określone wyłącznie wówczas, gdy do komputera jest podłączone więcej niż jedno działające urządzenie lub jest podłączony i uruchomiony więcej niż jeden emulator. Jeśli nie zainstalowano środowiska wykonawczego lub określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia. Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

W systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów.

```
adb devices
```

Polecenie `uninstallRuntime` narzędzia ADT

Polecenie `uninstallRuntime` powoduje całkowite usunięcie środowiska wykonawczego AIR z urządzenia lub z emulatora. Dla tego polecenia jest stosowana następująca składnia:

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Nazwa platformy urządzenia. W chwili obecnej to polecenie jest obsługiwane wyłącznie na platformie Android. Należy użyć nazwy *android*.

-platformsdk Ścieżka platformy zestawu SDK dla urządzenia docelowego. Obecnie jedyny obsługiwany zestaw SDK jest przeznaczony dla systemu Android. Zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Ścieżka zestawu SDK platformy nie musi być podawana w wierszu poleceń, jeśli ustawiono już zmienną środowiskową AIR_ANDROID_SDK_HOME. (W przypadku ustawienia obu tych funkcji jest używana ścieżka podana w wierszu poleceń).

-device Numer seryjny urządzenia. Urządzenie musi zostać określone wyłącznie wówczas, gdy do komputera jest podłączone więcej niż jedno działające urządzenie lub jest podłączony i uruchomiony więcej niż jeden emulator. Jeśli określone urządzenie nie jest podłączone, narzędzie ADT zwróci kod wyjścia 14: Błąd urządzenia. Jeśli podłączono więcej niż jedno urządzenie lub więcej niż jeden emulator, narzędzie ADT zwróci kod wyjścia 2: Błąd użytkownika.

W systemie Android narzędzie ADB systemu Android służy do podania listy numerów seryjnych podłączonych urządzeń i uruchomionych emulatorów.

```
adb devices
```

Polecenie `devices` programu ADT

Polecenie `-help` narzędzia ADT wyświetla identyfikatory aktualnie podłączonych urządzeń przenośnych i emulatorów:

```
adt -devices -platform iOS|android
```

Argument **-platform** określa nazwę platformy do sprawdzenia. Określ platformę `android` lub `iOS`.

Uwaga: W systemie iOS polecenie to wymaga programu iTunes 10.5.0 lub nowszego.

Polecenie help narzędzia ADT

Polecenie `-help` narzędzia ADT powoduje wyświetlenie zwięzłego przypomnienia opcji wiersza poleceń.

```
adt -help
```

W wynikach polecenia `help` są stosowane następujące oznaczenia symboliczne:

- `<>` — elementy między nawiasami trójkątnymi oznaczają informacje, które należy podać.
- `()` — elementy w zwykłych nawiasach oznaczają opcje, które w wynikach działania polecenia `help` są traktowane jak grupa.
- `WIELKIE_LITERY` — elementy podane za pomocą wielkich liter oznaczają zestaw opcji, który jest opisany oddzielnie.
- `|` — LUB. Na przykład zapis `(A | B)` oznacza element A lub element B.
- `?` — 0 lub 1. Znak zapytania umieszczony po elemencie oznacza, że dany element jest opcjonalny i w razie zastosowania może pojawiać się tylko jedno wystąpienie.
- `*` — 0 lub więcej. Gwiazdka umieszczona po elemencie oznacza, że element jest opcjonalny i może być dowolnie wiele jego wystąpień.
- `+` — 1 lub więcej. Znak plus umieszczona po elemencie oznacza, że element jest wymagany i że może być wiele jego wystąpień.
- Brak symbolu — jeśli po elemencie nie umieszczono żadnego przyrostka, wówczas ten element jest wymagany i może być tylko jedno wystąpienie tego elementu.

Zestawy opcji narzędzia ADT

Kilka poleceń narzędzia ADT korzysta z tych samych zestawów opcji.

Opcje podpisywania kodu ADT

Narzędzie ADT korzysta z architektury JCA (Java Cryptography Architecture) w celu uzyskiwania dostępu do kluczy prywatnych i certyfikatów przeznaczonych do podpisywania aplikacji AIR. Opcje podpisywania określają magazyn kluczy, klucz prywatny i certyfikat w tym magazynie kluczy.

Magazyn kluczy musi zawierać klucz prywatny oraz skojarzony z nim łańcuch certyfikatów. Jeśli certyfikat podpisujący prowadzi do zaufanego certyfikatu na komputerze, wówczas zawartość pola wspólnej nazwy certyfikatu jest wyświetlana jako nazwa wydawcy w oknie dialogowym instalowania środowiska AIR.

Narzędzie ADT wymaga, aby certyfikat był zgodny ze standardem x509v3 ([RFC3280](#)) i zawierał rozszerzenie Extended Key Usage z odpowiednimi wartościami dla podpisywania kodu. Ograniczenia zdefiniowane w ramach certyfikatu są uwzględniane i mogą wykluczać użycie niektórych certyfikatów na potrzeby podpisywania aplikacji AIR.

Uwaga: Narzędzie ADT korzysta w razie potrzeby z ustawień proxy środowiska JRE (Java Runtime Environment), aby nawiązywać połączenia z zasobami internetowymi w celu sprawdzania list unieważnienia certyfikatów oraz pobierania znaczników czasu. W razie problemów z połączeniem z tymi zasobami internetowymi — podczas korzystania z narzędzia ADT oraz wtedy, gdy sieć wymaga specjalnych ustawień proxy — konieczne może być skonfigurowanie ustawień proxy środowiska JRE.

Składnia opcji podpisywania środowiska AIR

Opcje podpisywania korzystają z następującej składni (w jednym wierszu polecenia):

Narzędzie ADT

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias Alias klucza w magazynie kluczy. Określenie aliasa nie jest konieczne, jeśli magazyn kluczy zawiera wyłącznie pojedynczy certyfikat. Jeśli alias nie zostanie określony, wówczas narzędzie ADT użyje pierwszego klucza z magazynu kluczy.

Nie wszystkie aplikacje do zarządzania magazynami kluczy zezwalają na przypisywanie aliasów do certyfikatów. Jeśli używany jest magazyn kluczy systemu Windows, jako aliasu należy użyć nazwy wyróżniającej certyfikatu. Za pomocą programu narzędziowego Java Keytool można wyświetlić listę dostępnych certyfikatów, dzięki czemu możliwe będzie określenie aliasu. Przykład: uruchomienie polecenia:

```
keytool -list -storetype Windows-MY
```

powoduje wyświetlenie następujących danych dla certyfikatu:

```
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
```

W celu utworzenia odwołania do tego certyfikatu w wierszu poleceń ADT należy ustawić następujący alias:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

W systemie Mac OS X alias certyfikatu w łańcuchu kluczy jest nazwą wyświetlaną w aplikacji Keychain Access.

-storetype Typ magazynu kluczy określony w implementacji magazynu kluczy. Domyślna implementacja magazynu kluczy dołączona do większości instalacji Java obsługuje typy JKS i PKCS12. Java 5.0 obsługuje typ PKCS11, który umożliwia dostęp do magazynów kluczy w tokenach sprzętowych, a także typ Keychain przeznaczony do dostępu do łańcucha kluczy Mac OS X. Java 6.0 obsługuje typ MSCAPI (w systemie Windows). W przypadku zainstalowania i skonfigurowania innych dostawców JCA mogą być dostępne inne typy magazynów kluczy. Jeśli nie określono typu magazynu kluczy, wówczas używany jest domyślny typ dla domyślnego dostawcy JCA.

Typ magazynu	Format magazynu kluczy	Minimalna wersja Java
JKS	Plik magazynu kluczy Java (.keystore)	1.2
PKCS12	Plik PKCS12 (.p12 lub .pfx)	1.4
PKCS11	Token sprzętowy	1.5
KeychainStore	Mac OS X Keychain	1.5
Windows-MY lub Windows-ROOT	MSCAPI	1.6

-keystore Ścieżka do pliku magazynu kluczy dla magazynów opartych na plikach.

-storepass Hasło wymagane do uzyskania dostępu do magazynu kluczy. Jeśli nie zostanie określone, program ADT wyświetli monit o podanie hasła.

-keypass Hasło wymagane do uzyskania dostępu do klucza prywatnego, który służy do podpisania aplikacji AIR. Jeśli nie zostanie określone, program ADT wyświetli monit o podanie hasła.

Uwaga: W przypadku wpisania hasła jako części polecenia ADT znaki hasła zostaną zapisane w historii wiersza poleceń. Z tego powodu nie jest zalecane stosowanie opcji `-keypass` ani `-storepass`, gdy jest ważne bezpieczeństwo certyfikatu. W przypadku pominięcia opcji hasła znaki wpisywane w odpowiedzi na żądania hasła nie są wyświetlane (z tych samych powodów związanych z bezpieczeństwem). Wystarczy wpisać hasło i nacisnąć klawisz Enter.

-providerName Dostawca JCA dla magazynu kluczy określonego typu. Jeśli ten argument nie zostanie określony, program ADT użyje domyślnego dostawcy dla magazynu kluczy tego typu.

-tsa Określa adres URL serwera znaczników czasu zgodnego ze standardem [RFC3161](#) na potrzeby oznaczenia podpisu cyfrowego znacznikiem czasu. Jeśli nie określono adresu URL, używany jest domyślny serwer znaczników czasu udostępniany przez Geotrust. Jeśli podpis aplikacji AIR jest oznaczony znacznikiem czasu, aplikacja może zostać zainstalowana po utracie ważności certyfikatu, ponieważ znacznik czasu potwierdza, że certyfikat był poprawny w czasie podpisywania.

Jeśli narzędzie ADT nie może połączyć się z serwerem znaczników czasu, wówczas podpisywanie zostaje anulowane i pakiet nie zostanie wygenerowany. W celu wyłączenia wstawiania znaczników czasu należy wprowadzić opcję `-tsa none`. Jednak po utracie ważności certyfikatu podpisującego nie będzie możliwości zainstalowania aplikacji AIR zapakowanej bez znacznika czasu.

Uwaga: Liczne opcje podpisywania odpowiadają tej samej opcji narzędzia Java Keytool. Za pomocą programu narzędziowego Keytool można sprawdzać magazyny kluczy Windows i zarządzać nimi. Do tego celu można w systemie Mac OS X wykorzystać program narzędziowy zabezpieczeń Apple®.

-provisioning-profile Plik informacyjny systemu Apple iOS. (Ta opcja jest wymagana wyłącznie w przypadku pakowania aplikacji systemu iOS).

Przykłady opcji podpisywania

Podpisywanie za pomocą pliku .p12:

```
-storetype pkcs12 -keystore cert.p12
```

Podpisywanie za pomocą domyślnego magazynu kluczy Java:

```
-alias AIRcert -storetype jks
```

Podpisywanie za pomocą określonego magazynu kluczy Java:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Podpisywanie za pomocą łańcucha kluczy Mac OS X:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Podpisywanie za pomocą magazynu kluczy Windows:

```
-alias cn=AIRCert -storetype Windows-MY
```

Podpisywanie za pomocą tokenu sprzętowego (zapoznać się z instrukcjami producenta tokenu dotyczącymi konfigurowania Java do użycia tokenu oraz w celu określenia poprawnej wartości `providerName`):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Podpisywanie bez osadzania znacznika czasu:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Opcje plików i ścieżek

Opcje plików i ścieżek określają wszystkie pliki, które wchodzi w skład pakietu. Dla opcji plików i ścieżek jest stosowana następująca składnia:

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs Pliki i katalogi, które będą pakowane w pliku AIR. Możliwe jest określenie dowolnej liczby plików i katalogów — poszczególne nazwy należy rozdzielać znakiem spacji. Po umieszczeniu na liście katalogu do pakietu zostaną dodane wszystkie jego pliki i podkatalogi, oprócz plików ukrytych. (Jeśli ponadto określono plik deskryptora aplikacji bezpośrednio, za pomocą znaku wieloznacznego lub rozszerzenia katalogu, ten plik zostanie zignorowany i nie zostanie dodany do pakietu po raz drugi). Wybrane pliki i katalogi muszą znajdować się w bieżącym katalogu lub w jednym z jego podkatalogów. W celu zmiany bieżącego katalogu należy użyć opcji `-c`.

Ważne: Znaki wieloznaczne nie mogą być używane w argumentach `file_or_dir` za opcją `-c`. (Powłoki poleceń rozszerzają znaki wieloznaczne przed wprowadzeniem argumentów do narzędzia ADT, co sprawia, że narzędzie ADT poszukuje plików w błędnej lokalizacji). Zamiast nazwy katalogu bieżącego można użyć znaku „.”. Na przykład polecenie: `-c assets .` powoduje skopiowanie całej zawartości katalogu `assets`, łącznie z jego podkatalogami, do głównego poziomu pakietu aplikacji.

Argument `-c katalog pliki_i_katalogi` zmienia katalog roboczy na wartość `katalog` przed rozpoczęciem przetwarzania kolejnych plików i katalogów dodawanych do pakietu aplikacji (określonych w lokalizacji `pliki_i_katalogi`). Pliki i katalogi są dodawane do katalogu głównego pakietu aplikacji. Opcja `-c` może być używana dowolną ilość razy w celu uwzględnienia plików z wielu punktów w systemie plików. Jeśli `dir` określa ścieżkę względną, ścieżka jest odczytywana zawsze względem oryginalnego katalogu roboczego.

W miarę przetwarzania przez narzędzie ADT plików i katalogów zawartych w pakiecie następuje zapisywanie względnych ścieżek między katalogiem bieżącym a plikami docelowymi. Podczas instalowania aplikacji te ścieżki są rozwijane do struktury katalogu aplikacji. Dlatego polecenie `-c release/bin lib/feature.swf` powoduje umieszczenie pliku `release/bin/lib/feature.swf` w podkatalogu `lib` głównego folderu aplikacji.

`-e file_or_dir dir` Powoduje umieszczenie pliku lub katalogu w określonym katalogu pakietu. Tej opcji nie można stosować podczas pakowania pliku ANE.

Uwaga: Element `<content>` pliku deskryptora aplikacji musi określać końcową lokalizację głównego pliku aplikacji w drzewie katalogów pakietu aplikacji.

`-extdir dir` Wartość parametru `dir` jest nazwą katalogu, w którym mają być wyszukiwane rozszerzenia natywne (pliki ANE). Należy podać ścieżkę bezwzględną lub określić ścieżkę względem katalogu bieżącego. Opcję `-extdir` można określić wielokrotnie.

Podany katalog zawiera pliki ANE dla rozszerzeń natywnych używanych przez aplikację. Każdy plik ANE w tym katalogu ma rozszerzenie nazwy pliku `ane`. Nazwa pliku przed rozszerzeniem nazwy pliku `ane` *nie musi* być jednak zgodna z wartością elementu `extensionID` w pliku deskryptora aplikacji.

Jeśli na przykład jest używana opcja `-extdir ./extensions`, katalog `extensions` może mieć strukturę podobną do następującej:

```
extensions/
  extension1.ane
  extension2.ane
```

Uwaga: Zastosowania opcji `-extdir` są różne w narzędziach ADT i ADL. W narzędziu ADL opcja ta określa katalog zawierający podkatalogi, z których każdy zawiera rozpakowany plik ANE. W narzędziu ADT opcja ta określa katalog zawierający pliki ANE.

Opcje połączenia debugera

Gdy formatem docelowym pakietu jest `apk-debug`, `ipa-debug` lub `ipa-debug-interpreter`, opcje połączenia pozwalają określić, czy aplikacja będzie próbowała nawiązać połączenie z debugerem zdalnym (typowe rozwiązanie w przypadku debugowania przez sieć Wi-Fi), czy będzie wykrywała połączenia przychodzące z debugera zdalnego (typowe rozwiązanie w przypadku debugowania przez połączenie USB). Opcja `-connect` służy do łączenia z debugerem. Opcja `-listen` służy do akceptowania połączenia z debugera za pośrednictwem połączenia USB. Opcje te wykluczają się wzajemnie. Nie można ich użyć jednocześnie.

Opcja `-connect` korzysta z następującej składni:

```
-connect hostString
```

-connect Jeśli ta opcja występuje, aplikacja próbuje połączyć się ze zdalnym debugerem.

hostString Ciąg identyfikujący komputer, na którym uruchomiono narzędzie FDB (Flash Debugging Tool). Jeśli ten ciąg nie zostanie określony, aplikacja spróbuje połączyć się z debugerem działającym na komputerze, na którym utworzono pakiet. Ciąg hosta może być w pełni kwalifikowaną nazwą domeny komputerowej (*nazwaurządzenia.podgrupa.example.com*) lub adresem IP (*192.168.4.122*). Jeśli nie można odnaleźć określonego (lub domyślnego) urządzenia, wówczas środowisko wykonawcze wyświetla okno dialogowe z prośbą o podanie poprawnej nazwy hosta.

Opcja `-listen` korzysta z następującej składni:

```
-listen port
```

-listen Jeśli ta opcja występuje, środowisko wykonawcze czeka na połączenie ze zdalnego debugera.

port (opcjonalnie) Port na którym jest prowadzone wykrywanie. Domyślnie środowisko wykonawcze prowadzi wykrywanie na porcie 7936. Więcej informacji na temat używania opcji `-listen` zawiera sekcja „[Zdalne debugowanie za pomocą programu FDB i połączenia USB](#)” na stronie 113.

Opcje profilowania aplikacji w systemie Android

Gdy formatem docelowym pakietu jest `apk-profile`, można użyć opcji programu profilującego do określenia, który wstępnie wczytany plik SWF ma być używany w celu profilowania pamięci i wydajności. Dla opcji programu profilującego jest stosowana następująca składnia:

```
-preloadSWFPath directory
```

-preloadSWFPath Jeśli występuje ta opcja, aplikacja próbuje znaleźć wstępnie wczytywany plik SWF w określonym katalogu. Jeśli ta opcja nie zostanie określona, narzędzie ADT dołącza wstępnie wczytany plik SWF z zestawu SDK środowiska AIR.

directory Katalog zawierający wstępnie wczytywany plik SWF programu profilującego.

Opcje rozszerzeń natywnych

Opcje rozszerzeń natywnych określają opcje i pliki używane do pakowania pliku ANE dla rozszerzenia natywnego. Tych opcji należy używać razem z poleceniem `package` narzędzia ADT, gdy opcja `-target` ma wartość `ane`.

```
extension-descriptor -swc swcPath  
  -platform platformName  
  -platformoptions path/platform.xml  
  FILE_OPTIONS
```

extension-descriptor Plik deskryptora rozszerzenia natywnego.

-swc Plik SWC zawierający kod ActionScript i zasoby rozszerzenia natywnego.

-platform Nazwa platformy obsługiwanej przez ten plik ANE. Można użyć wielu opcji `-platform`, z których każda będzie miała osobną wartość `FILE_OPTIONS`.

-platformoptions Ścieżka do pliku opcji platformy (`platform.xml`). Za pomocą tego pliku można określić inne niż domyślne opcje programu konsolidującego, bibliotek współużytkowanych i bibliotek statycznych innych firm stosowanych w rozszerzeniu. Więcej informacji oraz przykłady można znaleźć w artykule Natywne biblioteki systemu iOS.

FILE_OPTIONS Wskazuje pliki platformy natywnej, które mają zostać uwzględnione w pakiecie, na przykład biblioteki statyczne, które należy dołączyć do pakietu rozszerzeń natywnych. Opcje plików są szczegółowo opisane w sekcji „[Opcje plików i ścieżek](#)” na stronie 192. (W przypadku pakowania pliku ANE nie można używać opcji `-e`).

Więcej tematów Pomocy

[Pakowanie rozszerzenia natywnego](#)

Komunikaty o błędzie programu ADT

Poniższa tabela zawiera listę możliwych błędów, które mogą być zgłaszane przez program ADT oraz prawdopodobne przyczyny.

Błędy sprawdzania poprawności deskryptora aplikacji

Kod błędu	Opis	Uwagi
100	Nie można dokonać analizy deskryptora aplikacji	Sprawdź błędy składni XML w pliku deskryptora aplikacji, np. niezamknięte znaczniki.
101	Brak przestrzeni nazw	Dodaj brakującą przestrzeń nazw.
102	Niepoprawna przestrzeń nazw	Sprawdź pisownię dla przestrzeni nazw.
103	Nieoczekiwany element lub atrybut	Usuń nieprawidłowe elementy atrybuty. Niestandardowe wartości nie są dozwolone w pliku deskryptora. Sprawdź pisownię dla nazw elementów i atrybutów. Upewnij się, że elementy są umieszczone w poprawnym elemencie nadrzędnym oraz atrybuty są używane z poprawnymi elementami.
104	Brak elementu lub atrybutu	Dodaj wymagany element lub atrybut.
105	Element lub atrybut zawiera niepoprawną wartość	Popraw nieprawidłową wartość.
106	Nieprawidłowa kombinacja atrybutów okna	Nie można używać razem niektórych ustawień okna, np. <code>transparency = true</code> i <code>systemChrome = standard</code> . Należy zmienić jedno z niezgodnych ustawień.
107	Minimalny rozmiar okna jest większy niż maksymalny rozmiar okna	Zmień ustawienie rozmiaru minimalnego lub maksymalnego.
108	Atrybut został już użyty we wcześniejszym elemencie	

Kod błędu	Opis	Uwagi
109	Powielony element.	Należy usunąć powielony element.
110	Jest wymagany co najmniej jeden element określonego typu.	Należy dodać brakujący element.
111	Żaden z profili podanych w deskrytorze aplikacji nie obsługuje rozszerzeń natywnych.	Do listy supportedProfiles należy dodać profil, który obsługuje natywne rozszerzenia środowiska
112	Format docelowy aplikacji AIR nie obsługuje rozszerzeń natywnych.	Należy wybrać typ docelowy, który obsługuje rozszerzenia natywne.
113	Elementy <nativeLibrary> i <initializer> muszą zostać określone razem.	Dla każdej biblioteki natywnej w rozszerzeniu natywnym musi zostać określona funkcja inicjująca.
114	Odnaleziono element <finalizer> bez elementu <nativeLibrary>.	Nie należy określać obiektu kończącego, chyba że platforma korzysta z biblioteki natywnej.
115	Domyślna platforma nie może zawierać implementacji natywnej.	W domyślnym elemencie platformy nie należy określać biblioteki natywnej.
116	Wywołanie z przeglądarki nie jest obsługiwane w przypadku tego typu docelowego.	Element <allowBrowserInvocation> nie może mieć wartości true dla określonego typu docelowego pakowania.
117	Ten typ docelowy wymaga przynajmniej przestrzeni nazw n do pakowania rozszerzeń natywnych.	Należy zmienić przestrzeń nazw środowiska AIR w deskrytorze aplikacji na obsługiwaną wartość.

Informacje na temat przestrzeni nazw, elementów, atrybutów i ich poprawnych wartości zawiera rozdział „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Błędy ikon aplikacji

Kod błędu	Opis	Uwagi
200	Nie można otworzyć pliku ikony	Sprawdź, czy w określonej ścieżce istnieje plik. Użyj innej aplikacji, aby upewnić się, że plik można otworzyć.
201	Ikona ma nieprawidłowy rozmiar	Rozmiar ikony (w pikselach) musi być zgodny ze znacznikiem XML. Na przykład gdy mamy następujący element deskryptora aplikacji: <image32x32>icon.png</image32x32> Obraz w pliku icon.png musi mieć dokładnie 32x32 piksele.
202	Plik ikony zawiera nieobsługiwany format obrazu	Jedynym obsługiwany formatem jest PNG. Skonwertuj obrazy w innych formatach przed spakowaniem aplikacji.

Błędy pliku aplikacji

Kod błędu	Opis	Uwagi
300	Brak pliku lub nie można go otworzyć	Nie można odnaleźć pliku określonego w wierszu poleceń lub nie można go otworzyć.
301	Brak pliku deskryptora aplikacji lub nie można go otworzyć	Nie można znaleźć pliku deskryptora aplikacji w określonej ścieżce lub nie można go otworzyć.
302	Brak w pakiecie głównego pliku treści	Plik SWF lub HTML, do którego istnieje odwołanie w elemencie <code><content></code> deskryptora aplikacji, należy dodać do pakietu, umieszczając go w plikach wymienionych w wierszu poleceń programu ADT.
303	W pakiecie brakuje pliku ikony	Pliki ikon określone w deskrytorze aplikacji należy dodać do pakietu, umieszczając je wśród plików wymienionych w wierszu poleceń programu ADT. Pliki ikon nie są dodawane automatycznie.
304	Początkowa treść okna jest niepoprawna	Plik, do którego istnieje odwołanie w elemencie <code><content></code> deskryptora aplikacji, nie został rozpoznany jako poprawny plik HTML lub SWF.
305	Początkowa treść okna w wersji SWF nie jest obsługiwana w tej przestrzeni nazw	Wersja SWF pliku, do którego istnieje odwołanie w elemencie <code><content></code> deskryptora aplikacji, nie jest obsługiwana przez wersję środowiska AIR określonego w przestrzeni nazw deskryptora. Na przykład próba spakowania pliku SWF10 (Flash Player 10) jako początkowej treści aplikacji AIR 1.1 spowoduje powstanie tego błędu.
306	Profil nieobsługiwany.	Profil określony w pliku deskryptora aplikacji nie jest obsługiwany. Zobacz „supportedProfiles” na stronie 253.
307	Musi być dostępna przestrzeń nazw w wersji co najmniej <i>nnn</i> .	Należy korzystać z przestrzeni nazw odpowiedniej dla funkcji używanych w aplikacji (takiej jak przestrzeń nazw 2.0).

Kody wyjścia dla innych błędów

Kod wyjścia	Opis	Uwagi
2	Błąd składni	Sprawdź, czy argumenty podane w wierszu poleceń są poprawne.
5	Nieznany błąd	Ten błąd wskazuje na sytuację, w której nie można wyjaśnić warunków błędu. Możliwe główne przyczyny obejmują niezgodność między programem ADT oraz środowiskiem Java Runtime Environment, błąd instalacji ADT lub JRE oraz błędy programistyczne w programie ADT.
6	Nie można zapisać do katalogu wyjściowego	Upewnij się, że określony (lub niejawni) katalog wyjściowy jest dostępny i dysk zawierający katalog ma dostateczną ilość wolnego miejsca.

Kod wyjścia	Opis	Uwagi
7	Nie można uzyskać dostępu do certyfikatu	Upewnij się, że ścieżka do magazynu kluczy została określona poprawnie. Sprawdź, czy do certyfikatu w magazynie kluczy można uzyskać dostęp. Do pomocy w rozwiązywaniu problemów z dostępem do certyfikatów służy narzędzie Java 1.6 Keytool.
8	Niepoprawny certyfikat	Plik certyfikatu jest nieprawidłowy, zmodyfikowany lub nieważny.
9	Nie można podpisać pliku AIR	Sprawdź opcje podpisywania przekazane do programu ADT.
10	Nie można utworzyć znacznika czasu	Program ADT nie może nawiązać połączenia z serwerem znacznika czasu. W przypadku połączenia z Internetem za pośrednictwem serwera proxy może zaistnieć potrzeba konfiguracji ustawień proxy środowiska JRE.
11	Błąd tworzenia certyfikatu	Sprawdź argumenty wiersza poleceń użyte do tworzenia sygnatur.
12	Niepoprawne wejście	Sprawdź ścieżki plików oraz inne argumenty przekazane do programu ADT w wierszu poleceń.
13	Brak zestawu SDK urządzenia	Należy sprawdzić konfigurację zestawu SDK urządzenia. Narzędzie ADT nie może zlokalizować zestawu SDK urządzenia wymaganego do wykonania określonego polecenia.
14	Błąd urządzenia	Narzędzie ADT nie może wykonać polecenia z powodu problemu lub ograniczenia związanego z urządzeniem. Ten kod wyjścia jest na przykład podawany w przypadku próby odinstalowania aplikacji, która nie jest faktycznie zainstalowana.
15	Brak urządzeń	Należy sprawdzić, czy urządzenie jest podłączone i włączone lub czy jest uruchomiony emulator.
16	Brakujące składniki GPL	Bieżący zestaw SDK środowiska AIR nie zawiera wszystkich składników wymaganych do wykonania żądanej operacji.
17	Działanie narzędzia do tworzenia pakietów dla urządzeń zakończyło się niepowodzeniem.	Nie można utworzyć pakietu, ponieważ brakuje oczekiwanych składników systemu operacyjnego.

Błędy w systemie Android

Kod wyjścia	Opis	Uwagi
400	Bieżąca wersja zestawu SDK systemu Android nie obsługuje tego atrybutu.	Należy sprawdzić, czy nazwa atrybutu jest poprawnie zapisana i czy jest prawidłowym atrybutem dla elementu, w którym występuje. Jeśli atrybut został wprowadzony w wersji systemu Android nowszej niż 2.2, w poleceniu ADT może okazać się konieczne ustawienie flagi -platformsdk.
401	Bieżąca wersja zestawu SDK systemu Android nie obsługuje wartości atrybutu.	Należy sprawdzić, czy wartość atrybutu jest poprawnie zapisana i czy jest prawidłową wartością dla atrybutu. Jeśli wartość atrybutu została wprowadzona w wersji systemu Android nowszej niż 2.2, w poleceniu ADT może okazać się konieczne ustawienie flagi -platformsdk.
402	Bieżąca wersja zestawu SDK systemu Android nie obsługuje znacznika XML.	Należy sprawdzić, czy nazwa znacznika XML jest poprawnie zapisana i czy jest ona prawidłowym elementem dokumentu manifestu systemu Android. Jeśli element został wprowadzony w wersji systemu Android nowszej niż 2.2, w poleceniu ADT może okazać się konieczne ustawienie flagi -platformsdk.
403	Nie jest dozwolone nadpisywanie znacznika systemu Android.	Aplikacja próbuje nadpisać element manifestu systemu Android, który jest zarezerwowany do użytku przez środowisko AIR. Zobacz „ Ustawienia w systemie Android ” na stronie 80.
404	Nie jest dozwolone nadpisywanie atrybutu systemu Android.	Aplikacja próbuje nadpisać atrybut manifestu systemu Android, który jest zarezerwowany do użytku przez środowisko AIR. Zobacz „ Ustawienia w systemie Android ” na stronie 80.
405	Znacznik systemu Android %1 musi być pierwszym elementem w znaczniku manifestAdditions.	Należy przenieść określony znacznik do wymaganej lokalizacji.
406	Atrybut %1 znacznika %2 systemu Android zawiera nieprawidłową wartość %3.	Należy podać poprawną wartość atrybutu.

Zmienne środowiskowe narzędzia ADT

Narzędzie ADT odczytuje wartości następujących zmiennych środowiskowych (jeśli są one ustawione):

Wartość `AIR_ANDROID_SDK_HOME` określa ścieżkę do głównego katalogu zestawu SDK systemu Android (katalogu zawierającego folder `tools`). Zestaw SDK środowiska AIR 2.6+ zawiera narzędzia z zestawu SDK systemu Android wymagane do implementowania odpowiednich poleceń narzędzia ADT. Tę wartość należy ustawiać wyłącznie w celu użycia innej wersji zestawu SDK systemu Android. Jeśli jest ustawiona ta opcja, wówczas w przypadku uruchamiania poleceń ADT, które jej wymagają, nie trzeba określać opcji `-platformsdk`. W razie ustawienia zarówno tej opcji, jak i opcji wiersza poleceń, jest używana ścieżka określona w wierszu poleceń.

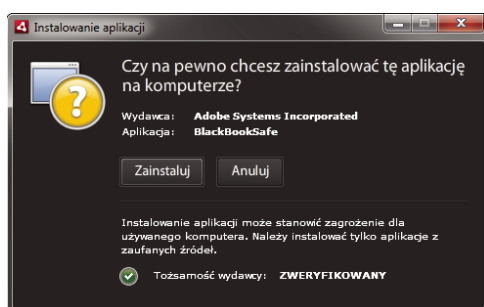
Wartość `AIR_EXTENSION_PATH` określa listę katalogów, w których mają być wyszukiwane rozszerzenia natywne wymagane przez aplikację. Ta lista katalogów jest kolejno przeszukiwana po przeszukaniu wszelkich katalogów rozszerzeń natywnych określonych w wierszu polecenia dla narzędzia ADT. Polecenie ADL również korzysta z tej zmiennej środowiskowej.

***Uwaga:** W niektórych systemach komputerowych dwubajtowe znaki w ścieżkach systemu plików przechowywane w tych zmiennych środowiskowych mogą zostać nieprawidłowo zinterpretowane. W takiej sytuacji należy spróbować skonfigurować środowisko JRE używane do uruchamiania narzędzia ADT w taki sposób, aby był stosowany zestaw znaków UTF-8. Domyślnie służy do tego skrypt używany do uruchamiania narzędzia ADT na komputerach Mac i na komputerach z systemem Linux. Opcję `-Dfile.encoding=UTF-8` należy określić w wierszu poleceń środowiska Java w pliku `adt.bat` systemu Windows lub podczas uruchamiania narzędzia ADT bezpośrednio w środowisku Java.*

Rozdział 13: Podpisywanie aplikacji AIR

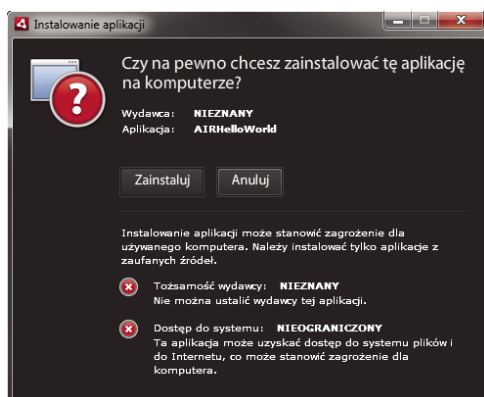
Cyfrowe podpisywanie pliku AIR

Elektroniczne podpisywanie plików instalacyjnych AIR za pomocą certyfikatu wystawionego przez zatwierdzony ośrodek certyfikacji (CA) stanowi istotne potwierdzenie dla użytkowników aplikacji, że aplikacja, którą instalują nie została przypadkowo lub złośliwie zmodyfikowana. Ponadto certyfikat stanowi potwierdzenie tożsamości autora podpisu. Środowisko wykonawcze AIR wyświetla nazwę wydawcy podczas instalowania pod warunkiem, że aplikacja AIR została podpisana certyfikatem zaufanym lub takim, który *kieruje* do certyfikatu zaufanego na komputerze instalacji:



Okno dialogowe instalacji dla aplikacji podpisanych za pomocą certyfikatu zaufanego

Jeśli aplikacja zostanie podpisana za pomocą certyfikatu samopodpisanego (lub certyfikatu, który nie wskazuje na certyfikat zaufany), wówczas użytkownik instalujący taką aplikację będzie musiał zaakceptować większe ryzyko. Okno dialogowe instalacji zawiera informację o tym dodatkowym ryzyku:



Okno dialogowe instalacji dla aplikacji podpisanych za pomocą certyfikatu podpisanego automatycznie

Ważne: Kod złośliwy może sfalszować plik AIR, korzystając z tożsamości programisty, jeśli w jakiś sposób uzyska plik magazynu kluczy do podpisywania lub wykryje klucz prywatny.

Certyfikaty do podpisywania kodu

Zabezpieczenia, ograniczenia oraz obowiązki prawne wynikające ze stosowania certyfikatów podpisujących kod zostały przedstawione w oświadczeniu CPS (Certificate Practice Statements) oraz umowach dotyczących subskrypcji wystawianych przez ośrodek certyfikacji. Więcej informacji na temat umów z ośrodkami certyfikacji, które obecnie wydają certyfikaty do podpisywania kodu dla środowiska AIR, można znaleźć na następujących stronach internetowych:

[ChosenSecurity](http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm) (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

[ChosenSecurity CPS](http://www.chosensecurity.com/resource_center/repository.htm) (http://www.chosensecurity.com/resource_center/repository.htm)

[GlobalSign](http://www.globalsign.com/code-signing/index.html) (<http://www.globalsign.com/code-signing/index.html>)

[GlobalSign CPS](http://www.globalsign.com/repository/index.htm) (<http://www.globalsign.com/repository/index.htm>)

[Thawte CPS](http://www.thawte.com/cps/index.html) (<http://www.thawte.com/cps/index.html>)

[VeriSign CPS](http://www.verisign.com/repository/CPS/) (<http://www.verisign.com/repository/CPS/>)

[Umowa subskrybenta — VeriSign](https://www.verisign.com/repository/subscriber/SUBAGR.html) (<https://www.verisign.com/repository/subscriber/SUBAGR.html>)

Informacje o podpisywaniu kodu w AIR

Podpisywanie pliku AIR oznacza dołączenie podpisu elektronicznego do pliku instalacyjnego. Podpis zawiera streszczenie pakietu, które służy do weryfikacji, że plik AIR nie został zmodyfikowany od czasu jego podpisania, a ponadto zawiera informacje o certyfikacie, które służą do sprawdzenia tożsamości wydawcy.

W środowisku AIR wykorzystywana jest infrastruktura kluczy publicznych za pośrednictwem magazynu certyfikatów systemu operacyjnego — w tej strukturze określane jest, czy certyfikat może być zaufany. Komputer, na którym zainstalowana jest aplikacja AIR musi bezpośrednio ufać certyfikatu używanemu do podpisania aplikacji AIR lub musi zaufać łańcuchowi certyfikatów łączących certyfikat z zaufanym ośrodkiem certyfikacji w celu weryfikacji informacji o wydawcy.

Jeśli plik AIR jest podpisany certyfikatem, który nie łączy się z jednym z zaufanych certyfikatów głównych (zwykle należą do nich certyfikaty samopodpisane), wówczas nie ma możliwości sprawdzenia informacji o wydawcy. Środowisko AIR może ustalić, czy pakiet AIR nie został zmodyfikowany od czasu jego podpisania, ale nie ma możliwości określenia autora pliku ani autora podpisu.

***Uwaga:** Użytkownik może zaufać certyfikatu samopodpisanemu, a wówczas wszystkie aplikacje AIR podpisane tym certyfikatem będą wyświetlały wartość pola nazwy wspólnej z certyfikatu jako nazwę wydawcy. Środowisko AIR nie udostępni użytkownikowi funkcji wyznaczenia certyfikatu jako podpisany. Certyfikat (bez klucza prywatnego) musi zostać udostępniony użytkownikowi osobno, a użytkownik musi korzystać z jednego z mechanizmów dostępnych w systemie operacyjnym lub z odpowiedniego narzędzia w celu zaimportowania certyfikatu do odpowiedniej lokalizacji w magazynie certyfikatów systemu.*

Informacje o identyfikatorach wydawców AIR

***Ważne:** Identyfikator wydawcy nie jest używany począwszy od wersji AIR 1.5.3 i nie jest już obliczany na podstawie certyfikatu podpisującego kod. Nowe aplikacje nie potrzebują identyfikatora wydawcy i nie powinny z niego korzystać. W przypadku aktualizacji istniejących aplikacji należy określić oryginalny identyfikator wydawcy w pliku deskryptora aplikacji.*

W wersjach wcześniejszych niż AIR 1.5.3 instalator aplikacji AIR generował identyfikator wydawcy podczas instalowania pliku AIR. Był to identyfikator unikalny dla certyfikatu służącego do podpisania pliku AIR. Jeśli ten sam certyfikat został wykorzystany dla wielu aplikacji AIR, wówczas identyfikator wydawcy dla każdej z tych aplikacji był taki sam. Podpisanie aktualizacji aplikacji innym certyfikatem, a czasami nawet odnowioną wersją oryginalnego certyfikatu, powodowało zmianę identyfikatora wydawcy.

W wersji AIR 1.5.3 i wersjach późniejszych ID wydawcy nie jest przypisywany przez AIR. Aplikacja opublikowana za pomocą AIR 1.5.3 może określać ciąg znaków identyfikatora wydawcy w deskrytorze aplikacji. Identyfikator wydawcy należy określić tylko w przypadku publikowania aktualizacji dla aplikacji pierwotnie opublikowanych dla wersji AIR wcześniejszych niż 1.5.3. Jeśli oryginalny identyfikator nie zostanie określony w deskrytorze aplikacji, wówczas nowy pakiet AIR nie będzie traktowany jako aktualizacja istniejącej aplikacji.

Aby ustalić oryginalny identyfikator wydawcy, należy odszukać plik `publisherid` w podkatalogu META-INF/AIR, w którym zainstalowana jest oryginalna aplikacja. Ciąg znaków w tym pliku jest identyfikatorem wydawcy. Deskrytor aplikacji powinien określać środowisko wykonawcze AIR 1.5.3 (lub późniejszą wersję) w deklaracji przestrzeni nazw pliku deskryptora aplikacji — tylko wówczas możliwe jest ręczne określenie identyfikatora wydawcy.

Identyfikator wydawcy — jeśli jest dostępny — jest używany do następujących celów:

- Jako część klucza szyfrowania dla zaszyfrowanej składnicy lokalnej
- Jako część ścieżki katalogu zapisu aplikacji
- Jako część ciągu znaków połączenia dla połączeń lokalnych
- Jako część ciągu znaków tożsamości używanego w celu wywołania aplikacji z interfejsem API AIR dostępnym w przeglądarce
- Jako część OSID (używany podczas tworzenia niestandardowych programów instalacyjnych/deinstalacyjnych)

Gdy dochodzi do zmiany identyfikatora wydawcy, następuje również zmiana działania wszystkich funkcji środowiska AIR zależnych od tego identyfikatora. Na przykład: dochodzi do utraty dostępu do danych w istniejącej składnicy lokalnej, a wszelkie instancje Flash lub AIR, które tworzą lokalne połączenie z aplikacją, muszą stosować nowy identyfikator w ciągu znaków połączenia. W środowisku AIR 1.5.3 lub nowszym nie można zmienić identyfikatora wydawcy zainstalowanej aplikacji. Jeśli w przypadku publikowania pakietu AIR używany jest inny identyfikator wydawcy, instalator traktuje nowy pakiet jak inną aplikację, a nie jak aktualizację.

Informacje o formatach certyfikatów

Narzędzia do podpisywania AIR akceptują dowolne magazyny kluczy dostępne za pośrednictwem architektury Java Cryptography Architecture (JCA). Ta architektura zawiera magazyny plików oparte na plikach, takie jak pliki w formacie PKCS12 (zwykle z rozszerzeniem `.pfx` lub `.p12`), pliki `.keystore` Java, magazyny kluczy PKCS11 urządzeń oraz systemowe magazyny kluczy. Formaty magazynów kluczy, do których narzędzie ADT może uzyskać dostęp, są uzależnione od wersji i konfiguracji środowiska wykonawczego Java używanego do uruchamiania narzędzia ADT. Dostęp do pewnych typów magazynów kluczy, np. do tokenów PKCS11 urządzeń, może wymagać zainstalowania i konfiguracji dodatkowych sterowników oprogramowania i wtyczek JCA.

Do podpisywania plików AIR można używać większości istniejących certyfikatów do podpisywania kodu. Można też uzyskać nowy certyfikat wydany specjalnie do podpisywania aplikacji AIR. Na przykład możliwe jest stosowanie dowolnego z wymienionych niżej certyfikatów wydawanych przez ośrodki VeriSign, Thawte, GlobalSign lub ChosenSecurity:

- [ChosenSecurity](#)
 - TC Publisher ID for Adobe AIR

- [GlobalSign](#)
 - ObjectSign Code Signing Certificate
- [Thawte](#):
 - AIR Developer Certificate
 - Apple Developer Certificate
 - JavaSoft Developer Certificate
 - Microsoft Authenticode Certificate
- [VeriSign](#):
 - Adobe AIR Digital ID
 - Microsoft Authenticode Digital ID
 - Sun Java Signing Digital ID

Uwaga: Certyfikat musi być utworzony jako certyfikat podpisujący kod. Do podpisywania plików AIR nie można używać certyfikatów SSL ani certyfikatów innego typu.

Znaczniki czasowe

Podczas podpisywania pliku AIR narzędzie do pakowania zadaje do serwera zapytanie dotyczące ośrodka wystawiającego znaczki czasu w celu niezależnej weryfikacji daty i czasu podpisu. Uzyskany znaczek czasu zostaje osadzony w pliku AIR. Jeśli certyfikat podpisujący obowiązuje podczas podpisywania, plik AIR może zostać zainstalowany nawet po utracie ważności certyfikatu. Z drugiej strony: jeśli nie uzyskano znaczka czasu, wówczas po utracie ważności lub odrzuceniu certyfikatu nie ma możliwości zainstalowania pliku AIR.

Domyślnie narzędzia do pakowania AIR uzyskują znaczki czasu. Jednak aby umożliwić pakowanie przy braku dostępności do usługi znaczka czasu, można wyłączyć wstawianie znaczka czasu. Adobe zaleca, aby wszystkie dystrybuowane publicznie pliki AIR zawierały znaczki czasu.

Domyślnym ośrodkiem wystawiającym znaczki czasu, z którego korzystają narzędzia pakowania AIR, jest Geotrust.

Uzyskiwanie certyfikatu

W celu uzyskania certyfikatu należy odwiedzić witrynę sieci Web ośrodka certyfikacji, a następnie przeprowadzić proces związany z uzyskiwaniem. Narzędzia, jakie są używane do generowania pliku magazynu kluczy wymaganego dla narzędzi AIR, są uzależnione od typu zakupionego certyfikatu, sposobu przechowywania certyfikatu na komputerze odbierającym oraz w niektórych przypadkach — od przeglądarki używanej w celu odebrania certyfikatu. Na przykład, aby uzyskać i wyeksportować certyfikat Adobe Developer wydany przez ośrodek Thawte, należy użyć przeglądarki Mozilla Firefox. Certyfikat można wyeksportować jako plik P12 lub PFX bezpośrednio z interfejsu użytkownika programu Firefox.

Uwaga: Środowisko Java w wersji 1.5 lub nowszej nie akceptuje znaków o wysokich kodach ASCII w hasłach chroniących pliki certyfikatów PKCS12. Środowisko Java jest wykorzystywane przez narzędzia programistyczne AIR do tworzenia podpisanych pakietów AIR. Jeśli certyfikat ma być eksportowany w pliku P12 lub PFX, należy w hasła używać tylko zwykłych znaków ASCII.

Certyfikat samopodpisany można wygenerować za pomocą narzędzia ADT (Air Development Tool) używanego do pakowania plików instalacyjnych AIR. Możliwe jest również korzystanie z niektórych narzędzi innych firm.

Instrukcje generowania certyfikatu podpisanego automatycznie i podpisywania plików AIR zawiera sekcja „[Narzędzie ADT](#)” na stronie 175. Pliki AIR można eksportować i podpisywać za pomocą programów Flash Builder, Dreamweaver oraz za pomocą aktualizacji AIR dla programu Flash.

Poniższy przykład przedstawia sposób uzyskania certyfikatu AIR Developer Certificate z ośrodka certyfikacji Thawte, a także sposób przygotowania tego certyfikatu do użytku z narzędziem ADT.

Przykład: uzyskanie certyfikatu AIR Developer Certificate z ośrodka Thawte

***Uwaga:** Niniejszy przykład ilustruje tylko jeden z wielu sposobów uzyskania i przygotowania certyfikatu podpisującego kod do użytku. Każdy ośrodek certyfikacji ma własne zasady i procedury.*

W celu zakupu certyfikatu AIR Developer Certificate w witrynie sieci Web Thawte należy korzystać z przeglądarki Mozilla Firefox. Klucz prywatny dla certyfikatu jest zapisany w magazynie kluczy przeglądarki. Należy się upewnić, że magazyn kluczy Firefox jest zabezpieczony hasłem głównym oraz że komputer jest zabezpieczony fizycznie. (Po zakończeniu procesu uzyskiwania możliwe będzie wyeksportowanie i usunięcie klucza prywatnego z magazynu kluczy przeglądarki).

W ramach procesu rejestrowania generowana jest para kluczy: prywatny/publiczny. Klucz prywatny jest automatycznie zapisywany w magazynie kluczy Firefox. Do żądania i pobierania certyfikatu z witryny sieci Web Thawte należy używać tego samego komputera.

- 1 Odwiedź witrynę sieci Web Thawte i przejdź do strony [certyfikatów podpisujących kod dla produktów](#).
- 2 Z listy certyfikatów podpisujących kod wybierz certyfikat Adobe AIR Developer Certificate.
- 3 Wykonaj trzykrokový proces rejestracji. Wymagane jest wprowadzenie informacji o organizacji oraz danych adresowych. Następnie Thawte przeprowadzi weryfikację tożsamości i może zażądać dodatkowych informacji. Po zakończeniu weryfikacji Thawte wyśle wiadomość e-mail z instrukcjami dotyczącymi pobrania certyfikatu.

***Uwaga:** Dodatkowe informacje o typie wymaganej dokumentacji zawiera dokument: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.*

- 4 Pobierz wydany certyfikat z serwisu Thawte. Certyfikat zostanie automatycznie zapisany w magazynie kluczy Firefox.
- 5 Wyeksportuj plik magazynu kluczy zawierający klucz prywatny i certyfikat z magazynu kluczy Firefox, wykonując poniższe czynności:

***Uwaga:** Prywatny klucz/certyfikat z Firefox jest eksportowany w formacie .p12 (pfx), z którego może korzystać program ADT, Flex, Flash i Dreamweaver.*

- a Otwórz okno dialogowe *Menedżer certyfikatów* przeglądarki Firefox:
- b W systemie Windows: wybierz kolejno opcje Narzędzia -> Opcje -> Zaawansowane -> Szyfrowanie -> Wyświetl certyfikaty
- c W systemie Mac OS: otwórz program Firefox -> Preferencje -> Zaawansowane -> Szyfrowanie -> Wyświetl certyfikaty
- d W systemie Linux: wybierz kolejno opcje Edycja -> Preferencje -> Zaawansowane -> Szyfrowanie -> Wyświetl certyfikaty
- e Z listy certyfikatów wybierz certyfikat Adobe AIR Code Signing Certificate i kliknij przycisk **Kopia zapasowa**.
- f Wprowadź nazwę i lokalizację pliku, do którego zostanie wyeksportowany plik magazynu kluczy, a następnie kliknij przycisk **Zapisz**.
- g Jeśli używane jest hasło główne Firefox, wymagane jest wprowadzenie hasła dla urządzenia zabezpieczenia oprogramowania w celu wyeksportowania pliku. (To hasło jest używane tylko przez program Firefox).
- h W oknie dialogowym *Wybierz hasło kopii zapasowej certyfikatu* utwórz hasło dla pliku magazynu kluczy.

***Ważne:** To hasło chroni plik magazynu kluczy oraz jest wymagane, gdy plik jest używany do podpisywania aplikacji AIR. Należy wybrać hasło bezpieczne.*

- i Kliknij przycisk OK. Powinien pojawić się komunikat informujący o pomyślnym zakończeniu wykonywania kopii zapasowej. Plik magazynu kluczy zawierający prywatny klucz i certyfikat jest zapisywany z rozszerzeniem .p12 (w formacie PKCS12)
- 6 Wyeksportowany plik magazynu kluczy można użyć z programem ADT, Flash Builder, Flash Professional lub Dreamweaver. Hasło utworzone dla pliku jest wymagane zawsze przy podpisywaniu aplikacji AIR.

***Ważne:** Klucz prywatny i certyfikat są nadal zapisane w magazynie kluczy Firefox. Dzięki temu możliwe jest wyeksportowanie dodatkowej kopii pliku certyfikatu, a ponadto jest to dodatkowy punkt dostępu, który musi być chroniony, aby możliwe było zachowanie bezpieczeństwa certyfikatu i klucza prywatnego.*

Zmiana certyfikatów

W pewnych sytuacjach należy zmienić używany certyfikat w celu podpisania aktualizacji dla danej aplikacji AIR. Do takich sytuacji należą:

- Odnowienie oryginalnego certyfikatu podpisującego.
- Aktualizacja z certyfikatu samopodpisanego do certyfikatu wydanego przez ośrodek certyfikacji
- Zmiana certyfikatu samopodpisanego, który wkrótce utraci ważność, na inny certyfikat
- Zmiana certyfikatu komercyjnego np. w przypadku zmiany tożsamości korporacyjnej

Aby środowisko AIR rozpoznało plik AIR jako aktualizację, należy podpisać oryginalne pliki AIR i pliki aktualizacji tym samym certyfikatem lub zastosować podpis migracji certyfikatu do aktualizacji. Podpis migracji jest drugim podpisem stosowanym względem pakietu aktualizacji AIR przy wykorzystaniu oryginalnego certyfikatu. Podpis migracji korzysta z certyfikatu oryginalnego w celu wskazania, że autor podpisu jest oryginalnym wydawcą aplikacji.

Po zainstalowaniu pliku AIR z podpisem migracji nowy certyfikat staje się certyfikatem głównym. Kolejne aktualizacje nie wymagają podpisu migracji. Jednak w miarę możliwości należy stosować podpisy migracji jak najdłużej, aby ułatwić korzystanie z aplikacji użytkownikom, którzy pomijają niektóre aktualizacje.

***Ważne:** Należy zmienić certyfikat i zastosować podpis migracji do aktualizacji z oryginalnym certyfikatem przed utratą jego ważności. W przeciwnym razie użytkownicy będą musieli odinstalować istniejącą wersję aplikacji przed zainstalowaniem nowej wersji. W środowisku AIR 1.5.3 lub nowszym można zastosować podpis migracji przy użyciu certyfikatu, który utracił ważność, w ciągu 365-dniowego okresu prolongaty od momentu utraty ważności. Nie można jednak użyć takiego certyfikatu w celu zastosowania głównego podpisu aplikacji.*

W celu zmiany certyfikatów:

- 1 Utwórz aktualizację aplikacji
- 2 Spakuj i podpisz plik AIR aktualizacji **nowym** certyfikatem
- 3 Podpisz ponownie plik AIR **oryginalnym** certyfikatem (korzystając z polecenia `-migrate` narzędzia ADT)

Pod innymi względami plik AIR z podpisem migracji jest normalnym plikiem AIR. Jeśli aplikacja zostanie zainstalowana w systemie, który nie zawiera wersji oryginalnej, wówczas środowisko AIR zainstaluje nową wersję w standardowy sposób.

***Uwaga:** W wersjach wcześniejszych niż AIR 1.5.3 podpisanie aplikacji AIR odnowionym certyfikatem nie zawsze wymagało podpisu migracji. Począwszy od wersji AIR 1.5.3 podpis migracji jest zawsze wymagany dla odnowionych certyfikatów.*

Aby zastosować podpis migracji, należy użyć „[Polecenie migrate narzędzia ADT](#)” na stronie 184 zgodnie z opisem zawartym w rozdziale „[Podpisywanie zaktualizowanej wersji aplikacji AIR](#)” na stronie 211.

Uwaga: Polecenia migrate programu ADT nie można używać z aplikacjami komputerowymi AIR, które zawierają rozszerzenia natywne, ponieważ takie aplikacje są spakowane jako instalatory natywne, a nie jako pliki AIR. Aby zmienić certyfikaty aplikacji komputerowej AIR, która zawiera rozszerzenie natywne, należy spakować aplikację przy użyciu „Polecenie package narzędzia ADT” na stronie 176 z flagą -migrate.

Zmiany tożsamości aplikacji

Począwszy od wersji AIR 1.5.3 tożsamość aplikacji AIR uległa zmianie po zainstalowaniu aktualizacji z podpisem migracji. Zmiana tożsamości aplikacji ma między innymi następujące konsekwencje:

- Wersja nowej aplikacji nie może uzyskać dostępu do danych w istniejącym zaszyfrowanym magazynie lokalnym.
- Zmienia się lokalizacja katalogu zapisu aplikacji. Dane ze starej lokalizacji nie są kopiowane do nowego katalogu. (Nowa aplikacja może zlokalizować katalog oryginalny na podstawie starego identyfikatora wydawcy).
- Aplikacja nie może otworzyć lokalnych połączeń za pomocą starego identyfikatora wydawcy.
- Zmienia się ciąg znaków tożsamości używany w celu uzyskania dostępu do aplikacji ze strony internetowej.
- Zmienia się OSID aplikacji. (OSID jest używany w przypadku pisania niestandardowych programów instalujących/deinstalujących).

W przypadku publikowania aktualizacji ze środowiskiem AIR 1.5.3 lub nowszym tożsamość aplikacji nie może ulec zmianie. Identyfikator oryginalnej aplikacji oraz identyfikator wydawcy muszą zostać określone w deskrypcji aplikacji pliku AIR aktualizacji. W przeciwnym wypadku nowy pakiet nie zostanie rozpoznany jako aktualizacja.

Uwaga: W przypadku publikacji nowej aplikacji AIR w wersji AIR 1.5.3 lub późniejszej nie należy określać identyfikatora wydawcy.

Terminologia

Niniejsza sekcja zawiera słowniczek terminów, które należy znać podczas podejmowania decyzji dotyczących sposobu podpisania aplikacji dla dystrybucji publicznej.

Termin	Opis
Ośrodek certyfikacji	Obiekt w sieci infrastruktury kluczy publicznych, który służy jako zaufana strona trzecia i w sposób ostateczny poświadcza tożsamość właściciela klucza publicznego. Ośrodek zwykle wydaje certyfikaty elektroniczne podpisywane własnymi kluczami prywatnymi, co stanowi dowód sprawdzenia tożsamości posiadacza certyfikatu.
Oświadczenie CPS (Certificate Practice Statement)	Określa praktyki i strategię ośrodka certyfikacji dotyczące wydawania i weryfikacji certyfikatów. Oświadczenie CPS jest częścią umowy między ośrodkiem certyfikacji a subskrybentami oraz stronami zależnymi. Określa również strategię weryfikacji tożsamości i poziom zabezpieczeń oferowanych przez udostępniane certyfikaty.
Lista CRL (Certificate Revocation List)	Lista wydanych certyfikatów, które zostały odrzucone i nie należy im ufać. AIR sprawdza CRL podczas podpisywania aplikacji AIR, a także w przypadku braku znacznika czasu przy instalowaniu aplikacji.
Łańcuch certyfikatów	Łańcuch certyfikatu jest sekwencją certyfikatów, w której każdy certyfikat z łańcucha został podpisany przez kolejny certyfikat.
Certyfikat elektroniczny	Dokument elektroniczny, który zawiera informacje o tożsamości właściciela, kluczu publicznym właściciela oraz o tożsamości samego certyfikatu. Certyfikat wydany przez ośrodek certyfikacji jest podpisany przez certyfikat należący do wydającego ośrodka.

Termin	Opis
Podpis elektroniczny	Zaszyfrowany komunikat lub zaszyfrowane streszczenie, które można odszyfrować tylko za pomocą klucza prywatnego lub za pomocą klucza prywatnego, będącego połową pary klucz publiczny - klucz prywatny. W infrastrukturze PKI podpis elektroniczny zawiera jeden lub większą liczbę certyfikatów elektronicznych, które ostatecznie prowadzą do ośrodka tożsamości. Podpis elektroniczny może służyć do sprawdzenia, czy komunikat (lub plik) nie został zmodyfikowany od czasu podpisania (z uwzględnieniem ograniczeń określonych przez algorytm kryptograficzny), oraz w celu sprawdzenia tożsamości autora podpisu — przy założeniu, że wydający ośrodek certyfikacji jest zaufany.
Magazyn kluczy	Baza danych zawierająca certyfikaty elektroniczne, a w niektórych przypadkach, powiązane klucze prywatne.
Java Cryptography Architecture (JCA)	Rozszerzalna architektura przeznaczona do zarządzania i uzyskiwania dostępu do magazynów kluczy. Więcej informacji zawiera dokumentacja Java Cryptography Architecture Reference Guide .
PKCS nr 11	Standard Cryptographic Token Interface Standard określony przez RSA Laboratories. Magazyn kluczy oparty na tokenie sprzętowym.
PKCS nr 12	Standard Personal Information Exchange Syntax Standard określony przez RSA Laboratories. Magazyn kluczy oparty na plikach, który zwykle zawiera klucz prywatny oraz skojarzony z nim certyfikat elektroniczny.
Klucz prywatny	Prywatna część dwuczęściowego, asymetrycznego systemu kryptografii obejmującego klucz prywatny i publiczny. Klucz prywatny należy chronić i nigdy nie należy przesyłać go przez sieć. Komunikaty podpisane elektronicznie są szyfrowane kluczem prywatnym przez autora podpisu.
Klucz publiczny	Publiczna część dwuczęściowego, asymetrycznego systemu kryptografii obejmującego klucz prywatny i publiczny. Klucz publiczny jest dostępny dla wszystkich i służy do deszyfrowania komunikatów zaszyfrowanych kluczem prywatnym.
Infrastruktura PKI (Public Key Infrastructure)	System zaufania, w którym ośrodki certyfikacji poświadczają prawdziwość tożsamości właściciela klucza publicznego. Klienci sieci polegają na certyfikatach elektronicznych wydawanych przez zaufany ośrodek certyfikacji w celu weryfikacji tożsamości autora komunikatu elektronicznego (lub pliku).
Znacznik czasu	Elektronicznie podpisany element danych, który zawiera datę i godzinę wystąpienia zdarzenia. Narzędzie ADT może dołączać znaczniki czasu z serwera czasu zgodnego ze standardem RFC 3161 w pakiecie AIR. Jeśli znacznik czasu jest dostępny, środowisko AIR korzysta z niego w celu ustalenia poprawności certyfikatu w czasie podpisywania. Dzięki temu aplikacja AIR może zostać zainstalowana po upływie ważności jej certyfikatu podpisującego.
Ośrodek wystawiający znacznik czasu	Ośrodek, który wystawia znaczniki czasu. Znacznik, który ma zostać rozpoznany przez środowisko AIR, musi być zgodny ze standardem RFC 3161, a podpis znacznika czasu musi prowadzić do zaufanego certyfikatu głównego na komputerze instalacji.

Certyfikaty w systemie iOS

Certyfikaty do podpisywania kodu wydawane przez firmę Apple umożliwiają podpisywanie aplikacji dla systemu iOS, w tym aplikacji opracowanych z użyciem środowiska Adobe AIR. W celu zainstalowania aplikacji na urządzeniach testowych jest wymagane zastosowanie podpisu przy użyciu certyfikatu programisty wydanego przez firmę Apple. Do rozpowszechniania ukończonej aplikacji jest wymagane zastosowanie podpisu przy użyciu certyfikatu rozpowszechniania.

Aby można było podpisać aplikację, zarówno certyfikat do podpisywania kodu, jak i powiązany z nim klucz prywatny muszą być dostępne dla programu ADT. Plik certyfikatu nie zawiera klucza prywatnego. Należy utworzyć magazyn kluczy jako plik w formacie Personal Information Exchange (p12 lub pfx), zawierający certyfikat i klucz prywatny. Więcej informacji zawiera rozdział „[Konwertowanie certyfikatu programisty na plik magazynu kluczy P12](#)” na stronie 209.

Generowanie wniosku o podpisanie certyfikatu

Aby uzyskać certyfikat programisty, należy wygenerować wniosek o podpisanie certyfikatu i złożyć go w witrynie Apple iOS Provisioning Portal.

Podczas obsługi wniosku o podpisanie certyfikatu jest generowana para kluczy (publiczny i prywatny). Klucz prywatny pozostaje na komputerze użytkownika. Użytkownik wysyła wniosek o podpisanie (zawierający klucz publiczny i informacje identyfikacyjne użytkownika) do firmy Apple, która pełni rolę urzędu certyfikacji. Firma Apple podpisuje certyfikat użytkownika za pomocą własnego certyfikatu World Wide Developer Relations.

Generowanie wniosku o podpisanie certyfikatu w systemie Mac OS

W systemie Mac OS możliwe jest użycie aplikacji Dostęp do pęku kluczy do wygenerowania wniosku o podpisanie certyfikatu. Aplikacja Dostęp do pęku kluczy stanowi podkatalog katalogu Narzędzia w katalogu Programy. Instrukcje umożliwiające wygenerowanie wniosku o podpisanie certyfikatu są dostępne w witrynie Apple iOS Provisioning Portal.

Generowanie wniosku o podpisanie certyfikatu w systemie Windows

W przypadku programistów pracujących w systemie Windows może okazać się łatwiejsze uzyskanie certyfikatu programisty iPhone na komputerze z systemem Mac. Możliwe jest jednak uzyskanie certyfikatu na komputer z systemem Windows. Najpierw utwórz wniosek o podpisanie certyfikatu (plik CSR), korzystając z warstwy OpenSSL:

- 1 Zainstaluj warstwę OpenSSL na komputerze z systemem Windows. (Przejdź do witryny <http://www.openssl.org/related/binaries.html>.)

Może być również potrzebne zainstalowanie plików redystrybuowalnych Visual C++ 2008, wymienionych na stronie pobierania protokołu Open SSL. (Nie ma potrzeby instalowania programu Visual C++ na posiadanym komputerze.)

- 2 Otwórz sesję wiersza poleceń Windows i przejdź (za pomocą polecenia CD) do podkatalogu bin katalogu OpenSSL (np. c:\OpenSSL\bin\).
- 3 Utwórz klucz prywatny, wprowadzając w wierszu poleceń:

```
openssl genrsa -out mykey.key 2048
```

Zapisz ten plik klucza prywatnego. Będzie on potrzebny później.

Korzystając z protokołu OpenSSL, nie ignoruj komunikatów o błędach. Mimo że protokół OpenSSL wygeneruje komunikat o błędzie, nadal może on generować pliki. Plik te mogą jednak okazać się bezużyteczne. W przypadku wyświetlenia błędów należy sprawdzić składnię, a następnie uruchomić polecenie ponownie.

- 4 Utwórz plik CSR, wprowadzając w wierszu poleceń:

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest -subj  
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

Zastąp adres e-mail, CN (nazwę certyfikatu) oraz C (kraj) własnymi wartościami.

- 5 Załaduj plik CSR do witryny Apple [iPhone Dev Center](#). (Patrz „Składanie wniosku o certyfikat programisty iPhone i tworzenie profilu informacyjnego”.)

Konwertowanie certyfikatu programisty na plik magazynu kluczy P12

Aby utworzyć magazyn kluczy P12, należy połączyć certyfikat programisty wydany przez firmę Apple i powiązany z nim klucz prywatny, umieszczając je w jednym pliku. Proces tworzenia pliku magazynu kluczy zależy od metody użytej do wygenerowania pierwotnego wniosku o podpisanie certyfikatu, a także od miejsca przechowywania klucza prywatnego.

Konwertowanie certyfikatu programisty telefonu iPhone na plik P12 w systemie Mac OS

Po pobraniu certyfikatu telefonu iPhone od firmy Apple należy wyeksportować go do formatu magazynu kluczy P12. Aby dokonać tego w systemie Mac OS:

- 1 Otwórz aplikację Dostęp do pęku kluczy (w folderze Aplikacje/Narzędzia).
- 2 Jeśli wcześniej nie wykonano tej czynności, dodaj plik certyfikatu programisty do pęku kluczy, wybierając opcję Plik > Importuj. Następnie przejdź do pliku certyfikatu (plik .cer) uzyskanego od firmy Apple.
- 3 Wybierz kategorię Klucze w aplikacji Dostęp do pęku kluczy.
- 4 Wybierz klucz prywatny skojarzony z certyfikatem instalacyjnym iPhone.
Klucz prywatny jest identyfikowany przy użyciu skojarzonego z nim certyfikatu publicznego wystawionego dla podmiotu „iPhone Developer: <imię> <nazwisko>”.
- 5 Naciśnij klawisz Command i kliknij certyfikat programisty telefonu iPhone, a następnie wybierz opcję *Eksportuj „iPhone Developer: nazwa...”*.
- 6 Zapisz magazyn kluczy w formacie pliku Personal Information Exchange (p12).
- 7 Zostanie wyświetlony monit o utworzenie hasła używanego podczas korzystania z magazynu kluczy w celu podpisywania aplikacji lub przekazywania klucza i certyfikatu z tego magazynu kluczy do innego magazynu kluczy.

Konwertowanie certyfikatu programisty firmy Apple na plik P12 w systemie Windows

Podczas opracowywania aplikacji AIR dla systemu iOS należy używać pliku certyfikatu P12. Certyfikat ten generuje się w oparciu o plik certyfikatu programisty Apple iPhone otrzymany od firmy Apple.

- 1 Plik certyfikatu programisty otrzymany od firmy Apple konwertuje się do pliku certyfikatu PEM. Uruchom następującą instrukcję wiersza poleceń z podkatalogu bin katalogu OpenSSL:

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```
- 2 W przypadku używania klucza z pęku kluczy na komputerze z systemem Mac dokonaj jego konwersji na klucz PEM:

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```
- 3 Możesz teraz wygenerować prawidłowy plik P12, bazujący na kluczu oraz wersji PEM certyfikatu programisty iPhone:

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

W przypadku używania klucza z pęku kluczy Mac OS użyj wersji PEM wygenerowanej w poprzednim kroku. W przeciwnym przypadku użyj klucza OpenSSL wygenerowanego wcześniej (w systemie Windows).

Tworzenie niepodpisanego pośredniego pliku AIR za pomocą narzędzia ADT

W celu utworzenia niepodpisanego pośredniego pliku AIR należy użyć polecenia `-prepare`. W celu wygenerowania poprawnego pliku instalacyjnego AIR należy podpisać plik pośredni AIR za pomocą polecenia ADT `-sign`.

Polecenie `-prepare` ma takie same flagi i parametry, jak polecenie `-package` (z wyjątkiem opcji podpisywania). Jedyną różnicą jest to, że plik wyjściowy nie jest podpisany. Plik pośredni jest wygenerowany z rozszerzeniem: `airi`.

W celu podpisania pliku pośredniego AIR należy użyć polecenia ADT `-sign`. Więcej informacji zawiera sekcja „[Polecenie prepare narzędzia ADT](#)” na stronie 183.

Przykład polecenia ADT -prepare

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Podpisywanie pliku pośredniego AIR za pomocą narzędzia ADT

W celu podpisania pliku pośredniego AIR za pomocą narzędzia ADT należy użyć polecenia `-sign`. Polecenie `sign` działa tylko z plikami pośrednimi AIR (rozszerzenie `airi`). Plik AIR nie może zostać podpisany po raz drugi.

W celu utworzenia pliku pośredniego AIR należy użyć polecenia `adt -prepare`. Więcej informacji zawiera sekcja „[Polecenie prepare narzędzia ADT](#)” na stronie 183.

Podpisywanie pliku AIRI

❖ Należy użyć polecenia ADT `-sign` z następującą składnią:

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS Opcje podpisywania identyfikują klucz prywatny oraz certyfikat służący do podpisywania pliku AIR. Te opcje zostały opisane w rozdziale „[Opcje podpisywania kodu ADT](#)” na stronie 190.

airi_file Ścieżka do niepodpisanego pośredniego pliku AIR przeznaczonego do podpisania.

air_file Nazwa pliku AIR, który jest tworzony.

Przykład polecenia ADT -sign

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Więcej informacji zawiera sekcja „[Polecenie sign narzędzia ADT](#)” na stronie 183.

Podpisywanie zaktualizowanej wersji aplikacji AIR

Podczas tworzenia zaktualizowanej wersji istniejącej aplikacji AIR zawsze jest podpisywana zaktualizowana aplikacja. W najlepszej sytuacji można podpisać zaktualizowaną wersję przy użyciu tego samego certyfikatu, który był używany z poprzednią wersją. Podpisywanie przebiega wtedy dokładnie tak samo jak podpisywanie aplikacji po raz pierwszy.

Jeśli certyfikat użyty do podpisania poprzedniej wersji aplikacji wygasł i został wznowiony lub zastąpiony, można podpisać zaktualizowaną wersję za pomocą wznowionego lub nowego (zastępczego) certyfikatu. Aby to zrobić, należy podpisać aplikację za pomocą nowego certyfikatu *oraz* zastosować podpis migracji przy użyciu oryginalnego certyfikatu. Podpis migracji poświadcza, że aktualizację opublikował właściciel oryginalnego certyfikatu.

Przed zastosowaniem podpisu migracji należy wziąć pod uwagę następujące kwestie:

- Podpis migracji można zastosować pod warunkiem, że oryginalny certyfikat jest nadal ważny lub utracił ważność w ciągu ostatnich 365 dni. Ten okres jest nazywany „okresem prolongaty”. Jego długość może ulec zmianie w przyszłości.

Uwaga: Do wersji AIR 2.6 okres prolongaty miał 180 dni.

- Nie można zastosować podpisu migracji po utracie ważności certyfikatu i upływie 365-dniowego okresu prolongaty. W takiej sytuacji użytkownicy muszą odinstalować istniejącą wersję przed zainstalowaniem wersji zaktualizowanej.

Podpisywanie aplikacji AIR

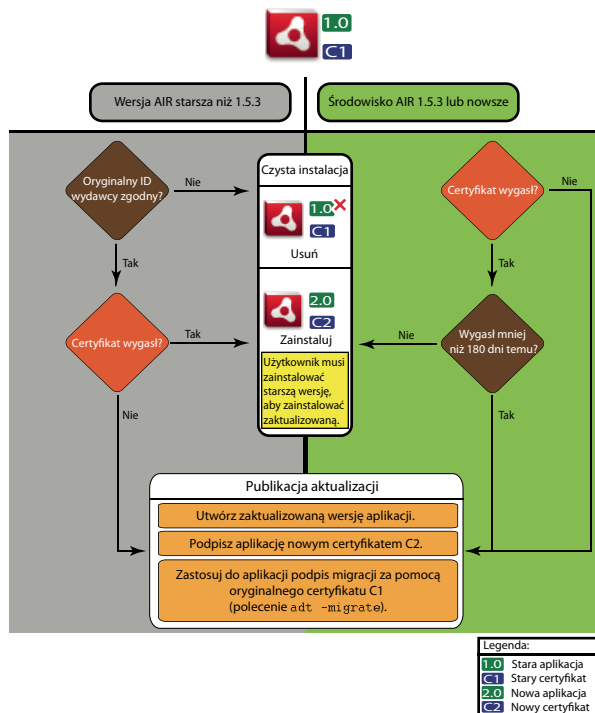
- 365-dniowy okres karencji dotyczy tylko aplikacji, które w przestrzeni nazw deskryptora mają określone środowisko AIR w wersji 1.5.3 lub nowszej.

Ważne: Podpisywanie aktualizacji za pomocą podpisów migracji z certyfikatów, które utraciły ważność, jest rozwiązaniem tymczasowym. Kompleksowe rozwiązanie polega na utworzeniu standardowego obiegu pracy dotyczącego podpisywania, który pozwoli zarządzać wdrażaniem aktualizacji aplikacji. Można na przykład podpisać wszystkie aktualizacje przy użyciu najnowszego certyfikatu i zastosować certyfikat migracji przy użyciu certyfikatu, za pomocą którego podpisano poprzednie aktualizacje (jeśli istniały). Należy wysłać każdą aktualizację na inny adres URL, z którego użytkownicy będą mogli pobrać aplikację. Więcej informacji zawiera rozdział „[Obieg pracy podpisywania aktualizacji aplikacji](#)” na stronie 275.

W tabeli i na rysunku poniżej podsumowano obieg pracy związany z podpisami migracji.

Scenariusz	Stan oryginalnego certyfikatu	Czynność programisty	Czynność użytkownika
Aplikacja oparta na środowisku wykonawczym Adobe AIR w wersji 1.5.3 lub nowszej	Ważny	Opublikowanie najnowszej wersji aplikacji AIR	Nie jest wymagana żadna czynność. Aplikacja jest uaktualniana automatycznie.
	Wygasł. Trwa 365-dniowy okres prolongaty.	Należy podpisać aplikację przy użyciu nowego certyfikatu. Należy zastosować podpis migracji przy użyciu wygasłego certyfikatu.	Nie jest wymagana żadna czynność. Aplikacja jest uaktualniana automatycznie.
	Wygasł. Minął okres prolongaty.	Nie można zastosować podpisu migracji do aktualizacji aplikacji AIR. Zamiast tego należy opublikować kolejną wersję aplikacji AIR przy użyciu nowego certyfikatu. Użytkownicy mogą zainstalować nową wersję po odinstalowaniu istniejącej wersji aplikacji AIR.	Należy odinstalować bieżącą wersję aplikacji AIR i zainstalować najnowszą.

Scenariusz	Stan oryginalnego certyfikatu	Czynność programisty	Czynność użytkownika
<ul style="list-style-type: none"> Aplikacja oparta na środowisku wykonawczym Adobe AIR 1.5.2 lub starszym Identyfikator wydawcy w deskrypcji aplikacji zgodny z identyfikatorem wydawcy poprzedniej wersji 	Ważny	Należy opublikować najnowszą wersję aplikacji AIR.	Nie jest wymagana żadna czynność. Aplikacja jest uaktualniana automatycznie.
	Wygaś. Minął okres prolongaty.	Nie można zastosować podpisu migracji do aktualizacji aplikacji AIR. Zamiast tego należy opublikować kolejną wersję aplikacji AIR przy użyciu nowego certyfikatu. Użytkownicy mogą zainstalować nową wersję po odinstalowaniu istniejącej wersji aplikacji AIR.	Należy odinstalować bieżącą wersję aplikacji AIR i zainstalować najnowszą.
<ul style="list-style-type: none"> Aplikacja oparta na środowisku wykonawczym Adobe AIR 1.5.2 lub starszym Identyfikator wydawcy w deskrypcji aplikacji niezgodny z identyfikatorem wydawcy poprzedniej wersji 	Dowolny	Należy podpisać aplikację AIR przy użyciu ważnego certyfikatu i opublikować najnowszą wersję.	Należy odinstalować bieżącą wersję aplikacji AIR i zainstalować najnowszą.



Obieg pracy podpisywania aktualizacji

Migracja aplikacji AIR w celu użycia nowego certyfikatu

Aby przeprowadzić migrację aplikacji AIR do nowego certyfikatu wraz z aktualizacją aplikacji:

- 1 Utwórz aktualizację aplikacji
- 2 Spakuj i podpisz plik AIR aktualizacji **nowym** certyfikatem
- 3 Podpisz ponownie plik AIR certyfikatem **original**, korzystając z polecenia `-migrate`

Plik AIR podpisany przy użyciu polecenia `-migrate` może posłużyć do zainstalowania nowej wersji aplikacji, a nie tylko do aktualizacji poprzedniej wersji podpisanej przy użyciu starego certyfikatu.

Uwaga: W przypadku aktualizowania aplikacji opublikowanej dla wersji środowiska AIR wcześniejszej niż 1.5.3 należy określić oryginalny identyfikator wydawcy w deskrytorze aplikacji. W przeciwnym wypadku przed zainstalowaniem aktualizacji użytkownicy aplikacji będą musieli odinstalować wcześniejszą wersję.

Należy użyć polecenia ADT `-migrate` z poniższą składnią:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** — opcje podpisywania identyfikują klucz prywatny oraz certyfikat służący do podpisywania pliku AIR. Te opcje muszą identyfikować **oryginalny** certyfikat podpisujący i zostały opisane w rozdziale „[Opcje podpisywania kodu ADT](#)” na stronie 190.
- **air_file_in** Plik AIR dla aktualizacji podpisany certyfikatem **new**.
- **air_file_out** Plik AIR do utworzenia.

Uwaga: Nazwy plików używane dla wejściowego i wyjściowego pliku środowiska AIR muszą być różne.

W poniższym przykładzie przedstawiono wywołanie programu ADT z flagą `-migrate` w celu zastosowania podpisu migracji do zaktualizowanej wersji aplikacji AIR.

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Uwaga: Polecenie `-migrate` zostało dodane do narzędzia ADT w środowisku AIR 1.1.

Migracja aplikacji AIR z instalatorem natywnym w celu użycia nowego certyfikatu

Aplikacji AIR opublikowanej w formie instalatora natywnego (na przykład aplikacji korzystającej z interfejsu API rozszerzeń natywnych) nie można podpisać przy użyciu polecenia `-migrate` programu ADT, ponieważ jest to aplikacja natywna przeznaczona dla określonej platformy, a nie plik AIR. W celu przeprowadzenia migracji aplikacji AIR opublikowanej w formie rozszerzenia natywnego należy wykonać następujące czynności:

- 1 Utwórz aktualizację aplikacji.
- 2 Upewnij się, że znacznik `<supportedProfiles>` w pliku deskryptora aplikacji (`app.xml`) zawiera zarówno profil `desktop`, jak i `extendedDesktop` (lub usuń znacznik `<supportedProfiles>` z deskryptora aplikacji).
- 3 Spakuj i podpisz aplikację aktualizującą **jako plik AIR** przy użyciu polecenia `-package` programu ADT z **nowym** certyfikatem.
- 4 Zastosuj certyfikat migracji do pliku AIR, używając polecenia `-migrate` programu ADT z **oryginalnym** certyfikatem (zgodnie z opisem w rozdziale „[Migracja aplikacji AIR w celu użycia nowego certyfikatu](#)” na stronie 214).
- 5 Spakuj plik AIR do instalatora natywnego, używając polecenia `-package` programu ADT z flagą `-target native`. Aplikacja jest już podpisana, dzięki czemu w tym kroku nie trzeba podawać certyfikatu podpisu.

Poniższy przykład ilustruje kroki 3–5 tego procesu. Kod wywołuje narzędzie ADT kolejno z poleceniami `-package`, `-migrate` i `-package`, aby spakować zaktualizowaną wersję aplikacji AIR w formie instalatora natywnego.

```
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
```

Migracja aplikacji AIR korzystającej z rozszerzenia natywnego w celu użycia nowego certyfikatu

Aplikacji AIR korzystającej z rozszerzenia natywnego nie można podpisać za pomocą polecenia `-migrate` programu ADT. Nie można też przeprowadzić dla niej procedury migracji aplikacji AIR spakowanej w formie instalatora natywnego, ponieważ nie można opublikować takiej aplikacji jako pośredniego pliku AIR. W celu przeprowadzenia migracji aplikacji AIR korzystającej z rozszerzenia natywnego należy wykonać następujące czynności:

1. Utwórz aktualizację aplikacji.
2. Spakuj i podpisz instalator natywny aktualizacji za pomocą polecenia `-package` programu ADT. Spakuj aplikację przy użyciu **nowego** certyfikatu i użyj flagi `-migrate` w celu określenia **oryginalnego** certyfikatu.

Wywołaj polecenie `-package` programu ADT z flagą `-migrate`, korzystając z następującej składni:

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** Opcje podpisywania identyfikują klucz prywatny oraz certyfikat służący do podpisywania pliku AIR. Te opcje muszą identyfikować **nowy** certyfikat podpisu i zostały opisane w rozdziale „[Opcje podpisywania kodu ADT](#)” na stronie 190.
- **MIGRATION_SIGNING_OPTIONS** Opcje podpisywania identyfikują klucz prywatny oraz certyfikat służący do podpisywania pliku AIR. Te opcje muszą identyfikować **oryginalny** certyfikat podpisu i zostały opisane w rozdziale „[Opcje podpisywania kodu ADT](#)” na stronie 190.
- Pozostałe opcje dotyczą pakowania aplikacji AIR w formie instalatora natywnego i zostały opisane w rozdziale „[Polecenie package narzędzia ADT](#)” na stronie 176.

Poniższy przykład ilustruje wywołanie programu ADT z poleceniem `-package` i flagą `-migrate` w celu spakowania zaktualizowanej wersji aplikacji AIR, która korzysta z rozszerzenia natywnego, a także zastosowania podpisu migracji do aktualizacji.

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore
original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Uwaga: Flaga `-migrate` polecenia `-package` jest dostępna w programie ADT w środowisku AIR 3.6 lub nowszym.

Tworzenie certyfikatu samopodpisanego za pomocą narzędzia ADT

Istnieje możliwość wygenerowania poprawnego pliku instalacyjnego AIR przy użyciu certyfikatu samopodpisanego. Jednak certyfikaty samopodpisane dają użytkownikom jedynie ograniczoną gwarancję bezpieczeństwa. Nie ma możliwości zweryfikowania autentyczności certyfikatu samopodpisanego. Podczas instalowania samopodpisanego pliku AIR informacje o wydawcy są wyświetlane jako nieznanne. Certyfikat wygenerowany przez ADT jest ważny przez pięć lat.

W przypadku utworzenia aktualizacji dla aplikacji AIR, która została podpisana certyfikatem samopodpisany, należy użyć tego samego certyfikatu w celu podpisania oryginalnych i zaktualizowanych plików AIR. Certyfikaty wygenerowane przez ADT są zawsze unikalne, nawet jeśli używane są te same parametry. Dlatego jeśli wymagane jest samopodpisanie aktualizacji za pomocą certyfikatu wygenerowanego przez ADT należy zachować oryginalny certyfikat w bezpiecznej lokalizacji. Ponadto wygenerowanie zaktualizowanego pliku AIR po utracie ważności oryginalnego certyfikatu wygenerowanego przez ADT nie będzie możliwe. (Możliwe jest publikowanie nowych aplikacji z innym certyfikatem, ale nie nowych wersji tej samej aplikacji).

Ważne: Z powodu ograniczeń certyfikatów samopodpisanych, firma Adobe zaleca korzystanie z komercyjnych certyfikatów wydawanych przez zaufane ośrodki certyfikacji do podpisywania publicznie wydawanych aplikacji AIR.

Certyfikat i skojarzony klucz prywatny wygenerowany przez ADT są zapisane w pliku magazynu kluczy typu PKCS12. Określone hasło jest ustawione w kluczu, a nie w magazynie kluczy.

Przykłady generowania certyfikatu

```
adt -certificate -cn SelfSign -ou QE -o "Example, Co" -c US 2048-RSA newcert.p12 39#wnetx3t1
adt -certificate -cn ADigitalID 1024-RSA SigningCert.p12 39#wnetx3t1
```

W celu użycia tych certyfikatów do podpisywania plików AIR należy użyć następujących opcji podpisywania z poleceniami ADT `-package` i `-prepare`:

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3t1
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3t1
```

Uwaga: Środowisko Java w wersji 1.5 lub nowszej nie akceptuje znaków o wysokich kodach ASCII w hasłach chroniących pliki certyfikatów PKCS12. W hasłach należy używać tylko zwykłych znaków ASCII.

Rozdział 14: Pliki deskryptora aplikacji AIR

Każda aplikacja AIR wymaga pliku deskryptora aplikacji. Plik deskryptora aplikacji jest dokumentem XML, który definiuje podstawowe właściwości aplikacji.

Wiele środowisk programistycznych, które obsługują środowisko AIR, automatycznie generuje deskryptor aplikacji, gdy użytkownik tworzy projekt. Jeśli deskryptor nie zostanie wygenerowany automatycznie, należy utworzyć własny. W katalogu `samples` pakietu SDK dla środowisk AIR i Flex znajduje się przykładowy plik deskryptora `descriptor-sample.xml`.

Dla pliku deskryptora aplikacji można użyć dowolnej nazwy. Po spakowaniu aplikacji nazwa pliku deskryptora aplikacji zostaje zmieniona na `application.xml` i plik zostaje umieszczony w specjalnym katalogu w pakiecie.

Przykładowy deskryptor aplikacji

Poniższy dokument deskryptora aplikacji ustawia podstawowe właściwości stosowane przez większość aplikacji 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>
```

Jeśli głównym plikiem treści aplikacji jest plik HTML, a nie plik SWF, różnica występuje tylko w elemencie `<content>`.

```
<content>
  HelloWorld.html
</content>
```

Zmiany deskryptora aplikacji

Deskryptor aplikacji AIR został zmieniony w poniższych wersjach środowiska AIR.

Zmiany deskryptora w środowisku AIR 1.1

Dozwolone elementy `name` i `description` są lokalizowane za pomocą elementu `text`.

Zmiany deskryptora w środowisku AIR 1.5

Właściwość `contentType` stała się wymaganym elementem potomnym właściwości `fileType`.

Zmiany deskryptora w środowisku AIR 1.5.3

W celu umożliwienia aplikacjom określania wartości identyfikatora wydawcy dodano element `publisherID`.

Zmiany deskryptora w środowisku AIR 2.0

Dodano elementy:

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (zobacz `imageNxN`)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

Zmiany deskryptora w środowisku AIR 2.5

Usunięto element `version`.

Dodano elementy:

- `android`
- `extensionID`
- `extensions`
- `image36x36` (zobacz `imageNxN`)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

Zmiany deskryptora w środowisku AIR 2.6

Dodano elementy:

- `image114x114` (zobacz `imageNxN`)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Zmiany deskryptora w środowisku AIR 3.0

Dodano elementy:

- `colorDepth`
- `direct` — prawidłowa wartość dla elementu `renderMode`
- `renderMode` — zakończenie ignorowania na platformach komputerowych
- Można określić element `<uses-sdk>` w systemie Android. (Dotąd nie był on dozwolony).

Zmiany deskryptora w środowisku AIR 3.1

Dodane elementy:

- „`Entitlements`” na stronie 232

Zmiany deskryptora w środowisku AIR 3.2

Dodano elementy:

- `depthAndStencil`
- `supportedLanguages`

Zmiany deskryptora w środowisku AIR 3.3

Dodano elementy:

- `aspectRatio` — zawiera teraz opcję `ANY` (dowolne).

Zmiany deskryptora w środowisku AIR 3.4

Dodane elementy:

- `image50x50` (zobacz „`imageNxN`” na stronie 240)
- `image58x58` (zobacz „`imageNxN`” na stronie 240)
- `image100x100` (zobacz „`imageNxN`” na stronie 240)
- `image1024x1024` (zobacz „`imageNxN`” na stronie 240)

Zmiany deskryptora w środowisku AIR 3.6

Dodane elementy:

- Element „[requestedDisplayResolution](#)” na stronie 250 w elemencie „[iPhone](#)” na stronie 244 zawiera teraz atrybut `excludeDevices`, który umożliwia wybranie docelowych urządzeń z systemem iOS używających wysokiej lub standardowej rozdzielczości.
- Nowy element „[requestedDisplayResolution](#)” na stronie 250 w elemencie „[initialWindow](#)” na stronie 242 określa, czy na platformach komputerowych z wyświetlaczami o wysokiej rozdzielczości (na przykład na komputerach Mac) ma być używana wysoka czy standardowa rozdzielczość.

Zmiany deskryptora w środowisku AIR 3.7

Dodane elementy:

- Element „[iPhone](#)” na stronie 244 udostępnia obecnie element „[externalSwfs](#)” na stronie 234, który pozwala określić listę plików SWF wczytywanych w czasie uruchamiania.
- Element „[iPhone](#)” na stronie 244 oferuje obecnie element „[forceCPURenderModeForDevices](#)” na stronie 237, za pomocą którego można wymusić tryb renderowania z użyciem procesora dla określonego zbioru urządzeń.

Struktura pliku deskryptora aplikacji

Plik deskryptora aplikacji jest dokumentem XML o następującej strukturze:

```
<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” na stronie 252>...</„supportedLanguages” na stronie 252>
<supportedProfiles>...</supportedProfiles>
<versionNumber>...</versionNumber>
<versionLabel>...</versionLabel>
</application>
```

Elementy deskryptora aplikacji AIR

W poniższym słowniku elementów opisano każdy z poprawnych elementów pliku deskryptora aplikacji AIR.

allowBrowserInvocation

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Umożliwia interfejsowi API środowiska AIR w przeglądarce wykrywanie i uruchamianie aplikacji.

Jeśli dla tej wartości zostanie ustawiona wartość `true`, wówczas należy rozważyć zagadnienia związane z bezpieczeństwem. Zostały one opisane w sekcji [Wywoływanie aplikacji AIR z przeglądarki](#) (dla programistów ActionScript) oraz w sekcji [Wywoływanie aplikacji AIR z przeglądarki](#) (dla programistów HTML).

Więcej informacji zawiera sekcja „[Uruchamianie zainstalowanej aplikacji AIR z przeglądarki](#)” na stronie 271.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

`true` lub `false` (domyślnie)

Przykład

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 i nowsze wersje — element opcjonalny

Umożliwia dodawanie elementów do pliku manifestu systemu Android. Środowisko AIR tworzy plik `AndroidManifest.xml` dla każdego pakietu APK. Korzystając z elementu `android` w deskrytorze aplikacji AIR, można dodawać do tego pliku dodatkowe elementy. Ten element jest ignorowany na wszystkich platformach poza platformą Android.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne:

- „[colorDepth](#)” na stronie 227
- „[manifestAdditions](#)” na stronie 245

Zawartość

Elementy definiujące właściwości specyficzne dla systemu Android, które mają zostać dodane do manifestu aplikacji Android.

Przykład

```
<android>
  <manifestAdditions>
    ...
  </manifestAdditions>
</android>
```

Więcej tematów Pomocy

„[Ustawienia w systemie Android](#)” na stronie 80

[Plik AndroidManifest.xml](#)

application

Adobe AIR 1.0 i nowsze wersje — element wymagany

Główny element dokumentu deskryptora aplikacji AIR.

Element macierzysty: brak

Elementy potomne:

- „[allowBrowserInvocation](#)” na stronie 222
- „[android](#)” na stronie 222
- „[copyright](#)” na stronie 229
- „[customUpdateUI](#)” na stronie 229
- „[description](#)” na stronie 230
- „[extensions](#)” na stronie 233
- „[filename](#)” na stronie 234
- „[fileTypes](#)” na stronie 236
- „[icon](#)” na stronie 239
- „[id](#)” na stronie 239
- „[initialWindow](#)” na stronie 242
- „[installFolder](#)” na stronie 243
- „[iPhone](#)” na stronie 244
- „[name](#)” na stronie 247
- „[programMenuFolder](#)” na stronie 249
- „[publisherID](#)” na stronie 249
- „[supportedLanguages](#)” na stronie 252
- „[supportedProfiles](#)” na stronie 253
- „[version](#)” na stronie 255
- „[versionLabel](#)” na stronie 256
- „[versionNumber](#)” na stronie 256

Atrybuty

Wartość `minimumPatchLevel` oznacza minimalny poziom poprawek środowiska wykonawczego AIR wymagany przez tę aplikację.

Wartość `xmlns` to atrybut przestrzeni nazw XML, który określa wymaganą przez aplikację wersję środowiska wykonawczego AIR.

Przestrzeń nazw zmienia się z każdym głównym wydaniem środowiska AIR (ale nie w przypadku mniejszych poprawek). Ostatni segment przestrzeni nazw, na przykład „3.0”, określa wersję środowiska wykonawczego, jakiej wymaga aplikacja.

Wartości atrybutu `xmlns` dla najważniejszych wersji środowiska AIR:

```
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"
```

Dla aplikacji w formacie SWF wersja środowiska wykonawczego AIR podana w deskrytorze aplikacji określa maksymalną wersję pliku SWF, jaki może zostać załadowany jako początkowa treść aplikacji. Dla aplikacji, dla których została określona wersja środowiska AIR 1.0 lub AIR 1.1, jako początkowej treści można używać jedynie plików SWF9 (Flash Player 9) — nawet jeśli zostały uruchomione za pomocą środowiska wykonawczego AIR 2. Dla aplikacji, dla których została określona wersja środowiska AIR 1.5 (lub nowsza), jako początkowej treści można używać plików SWF9 lub SWF10 (Flash Player 10).

Wersja SWF określa, które wersje interfejsów API środowiska AIR i programu Flash Player są dostępne. Jeśli plik SWF9 jest używany jako początkowa treść aplikacji AIR 1.5, aplikacja będzie miała dostęp jedynie do interfejsów API środowiska AIR 1.1 i programu Flash Player 9. Ponadto zmiany w działaniu dokonane w istniejących interfejsach API środowiska AIR 2.0 lub programu Flash Player 10.1 nie odniosą skutku. (Istotne zmiany w interfejsach API związane z bezpieczeństwem są odstępstwem od tej reguły i mogą być stosowane wstecz w bieżących lub przyszłych poprawkach środowiska wykonawczego).

Dla aplikacji w formacie HTML wersja środowiska wykonawczego podana w deskrytorze aplikacji samodzielnie określa, które wersje interfejsów API środowiska AIR i programu Flash Player są dostępne dla aplikacji. Działanie kodu HTML, CSS i JavaScript jest zawsze określone przez wersję silnika Webkit używanego w zainstalowanym środowisku wykonawczym AIR, a nie przez deskryptor aplikacji.

Podczas ładowania treści SWF przez aplikację AIR wersja interfejsów API środowiska AIR i programu Flash Player dostępnych dla tej treści zależy od sposobu jej ładowania. Niekiedy obowiązująca wersja jest ostatecznie ustalana na podstawie przestrzeni nazw deskryptora aplikacji, niekiedy na podstawie wersji treści ładującej, a niekiedy na podstawie wersji treści ładowanej. W poniższej tabeli został przedstawiony sposób, w jaki wersja interfejsu API jest określana w oparciu o metodę ładowania:

Sposób ładowania treści	Sposób określania wersji interfejsu API
Początkowa treść, aplikacja w formacie SWF	Wersja SWF załadowanego pliku
Początkowa treść, aplikacja w formacie HTML	Przestrzeń nazw deskryptora aplikacji
Plik SWF załadowany przez treść SWF	Wersja treści ładującej

Sposób ładowania treści	Sposób określania wersji interfejsu API
Biblioteka SWF załadowana przez treść HTML za pomocą znacznika <script>	Przestrzeń nazw deskryptora aplikacji
Plik SWF załadowany przez treść HTML za pomocą interfejsów API środowiska AIR lub programu Flash Player (np. flash.display.Loader)	Przestrzeń nazw deskryptora aplikacji
Plik SWF załadowany przez treść HTML za pomocą znaczników <object> lub <embed> (lub równoważnych interfejsów API języka JavaScript)	Wersja SWF wczytanego pliku

Podczas ładowania pliku SWF w wersji innej niż treść ładująca można napotkać na dwa problemy:

- Wczytanie pliku SWF nowszej wersji przez plik SWF starszej wersji. Odniesienia do interfejsów API dodane w nowszych wersjach środowiska AIR i programu Flash Player pozostaną nierozstrzygnięte.
- Wczytanie pliku SWF starszej wersji przez plik SWF nowszej wersji. Zachowania interfejsów API zmienionych w nowszych wersjach środowiska AIR i programu Flash Player mogą odbiegać od tych, których oczekuje wczytana zawartość.

Zawartość

Element application zawiera elementy potomne, które definiują właściwości aplikacji AIR.

Przykład

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/3.0">
  <id>HelloWorld</id>
  <version>2.0</version>
  <filename>Hello World</filename>
  <name>Example Co. AIR Hello World</name>
  <description>
    <text xml:lang="en">This is an example.</text>
    <text xml:lang="fr">C'est un exemple.</text>
    <text xml:lang="es">Esto es un ejemplo.</text>
  </description>
  <copyright>Copyright (c) 2010 Example Co.</copyright>
  <initialWindow>
    <title>Hello World</title>
    <content>
      HelloWorld.swf
    </content>
    <systemChrome>none</systemChrome>
    <transparent>true</transparent>
    <visible>true</visible>
    <minSize>320 240</minSize>
  </initialWindow>
  <installFolder>Example Co/Hello World</installFolder>
  <programMenuFolder>Example Co</programMenuFolder>
  <icon>
    <image16x16>icons/smallIcon.png</image16x16>
    <image32x32>icons/mediumIcon.png</image32x32>
  </icon>
</application>
```

```
<image48x48>icons/bigIcon.png</image48x48>
<image128x128>icons/biggestIcon.png</image128x128>
</icon>
<customUpdateUI>true</customUpdateUI>
<allowBrowserInvocation>>false</allowBrowserInvocation>
<fileTypes>
  <fileType>
    <name>adobe.VideoFile</name>
    <extension>avf</extension>
    <description>Adobe Video File</description>
    <contentType>application/vnd.adobe.video-file</contentType>
    <icon>
      <image16x16>icons/avfIcon_16.png</image16x16>
      <image32x32>icons/avfIcon_32.png</image32x32>
      <image48x48>icons/avfIcon_48.png</image48x48>
      <image128x128>icons/avfIcon_128.png</image128x128>
    </icon>
  </fileType>
</fileTypes>
</application>
```

aspectRatio

Adobe AIR 2.0 i nowsze wersje, Android i iOS — element opcjonalny

Określa proporcje aplikacji.

Jeśli ten element nie został określony, aplikacja jest otwierana w „naturalnych” proporcjach i w naturalnej orientacji urządzenia. Naturalna orientacja zależy od urządzenia. Zazwyczaj na urządzeniach z niewielkim wyświetlaczem, takich jak telefony, są to proporcje pionowe. Na niektórych urządzeniach, takich jak tablet iPad, aplikacja jest otwierana w bieżącej orientacji. W środowisku AIR 3.3 lub nowszym ma to zastosowanie do całej aplikacji, nie tylko do początkowego formatu wyświetlania.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

portrait, landscape lub any

Przykład

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

autoOrients

Adobe AIR 2.0 i nowsze wersje, Android i iOS — element opcjonalny

Określa, czy orientacja bieżącej zawartości aplikacji jest automatycznie zmieniana wraz ze zmianą orientacji fizycznej urządzenia. Więcej informacji zawiera artykuł [Orientacja stołu montażowego](#).

W przypadku używania automatycznej orientacji należy wziąć pod uwagę ustawienie właściwości `align` i `scaleMode` stołu montażowego w następujący sposób:

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Te ustawienia pozwalają aplikacji na obracanie dookoła górnego lewego rogu i uniemożliwiają automatyczne skalowanie zawartości aplikacji. W innych trybach skalowania zawartość aplikacji jest dopasowywana do obróconych wymiarów stołu montażowego, ale jest ona również przycinana, zniekształcana lub nadmiernie zmniejszana. Prawie zawsze można osiągnąć lepsze wyniki, samodzielnie ponownie rysując zawartość lub zmieniając jej układ.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

true lub false (domyślnie)

Przykład

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

colorDepth

Adobe AIR 3 i nowsze wersje — element opcjonalny

Określa, czy mają być używane kolory 16-bitowe, czy 32-bitowe.

Stosowanie kolorów 16-bitowych może zwiększyć wydajność renderowania, lecz obniża jakość kolorów. W wersjach wcześniejszych niż AIR 3 kolory 16-bitowe są zawsze używane w systemie Android. W środowisku AIR 3 są domyślnie używane kolory 32-bitowe.

***Uwaga:** Jeśli aplikacja korzysta z klasy StageVideo, należy używać kolorów 32-bitowych.*

Element macierzysty: „[android](#)” na stronie 222

Elementy potomne: brak

Zawartość

Jedna z następujących wartości:

- 16bit
- 32bit

Przykład

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

containsVideo

Określa, czy aplikacja zawiera elementy wideo.

Element macierzysty: „[android](#)” na stronie 222

Elementy potomne: brak

Zawartość

Jedna z następujących wartości:

- true
- false

Przykład

```
<android>
  <containsVideo>true</containsVideo>
  <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 i nowsze wersje — element wymagany

Wartość określona dla elementu `content` jest adresem URL głównego pliku zawartości aplikacji. Może to być plik SWF lub HTML. Adres URL jest określony względem głównego katalogu folderu instalacyjnego aplikacji. (W przypadku uruchomienia aplikacji AIR za pomocą programu ADL adres URL jest określony względem folderu zawierającego plik deskryptora aplikacji. W celu określenia innego katalogu głównego należy określić parametr `root-dir` programu ADL).

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Adres URL podany względem katalogu aplikacji. Wartość elementu `content` jest traktowana jako adres URL, dlatego znaki w nazwie pliku `content` muszą być zapisane w kodzie URL zgodnie z regułami zdefiniowanymi w normie [RFC 1738](#). Na przykład: znaki spacji muszą być zakodowane jako `%20`.

Przykład

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR w wersji od 1.0 do 1.1 — element opcjonalny; AIR 1.5 i nowsze wersje — element wymagany

Właściwość `contentType` jest wymagana w środowisku AIR 1.5 lub nowszym. (Była ona właściwością opcjonalną w środowisku AIR 1.0 i 1.1). Ta właściwość pomaga niektórym systemom operacyjnym zlokalizować najlepszą aplikację do otwarcia pliku. Wartość powinna być typem MIME dla treści pliku. Należy zauważyć, że wartość jest ignorowana w systemie Linux, jeśli typ pliku jest już zarejestrowany i ma przypisany typ MIME.

Element macierzysty: „[fileType](#)” na stronie 235

Elementy potomne: brak

Zawartość

Typ i podtyp MIME. Więcej informacji na temat typów MIME można znaleźć w dokumencie [RFC2045](#).

Przykład

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Informacje o prawach autorskich dotyczących aplikacji AIR. W systemie Mac OS tekst dotyczący praw autorskich pojawia się w oknie dialogowym informacji dla zainstalowanej aplikacji. W systemie Mac OS informacje o prawach autorskich są również wykorzystywane w polu NSHumanReadableCopyright w pliku Info.plist aplikacji.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Ciąg zawierający informacje o prawach autorskich dotyczących aplikacji.

Przykład

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy aplikacja zapewnia własne okna dialogowe aktualizacji. W przypadku wartości `false` środowisko AIR przedstawia użytkownikowi standardowe okna dialogowe aktualizacji. Z wbudowanego systemu aktualizacji środowiska AIR mogą korzystać wyłącznie aplikacje rozpowszechniane w postaci plików AIR.

Jeśli element `customUpdateUI` zainstalowanej wersji aplikacji jest ustawiony na `true`, a użytkownik kliknie dwukrotnie plik AIR dla nowej wersji lub zainstaluje aktualizację aplikacji za pomocą funkcji instalacji bezproblemowej, wówczas środowisko operacyjne otworzy zainstalowaną wersję aplikacji. Środowisko wykonawcze nie otwiera domyślnego instalatora aplikacji AIR. Logika aplikacji może wówczas określić sposób kontynuowania operacji aktualizacji. (Identyfikator aplikacji i identyfikator wydawcy w pliku AIR muszą być zgodne z identyfikatorami w zainstalowanej aplikacji — tylko wówczas możliwe będzie kontynuowanie aktualizacji).

***Uwaga:** Mechanizm `customUpdateUI` jest uwzględniany tylko wówczas, gdy aplikacja jest już zainstalowana, a użytkownik kliknie dwukrotnie plik instalacyjny AIR zawierający aktualizację lub zainstaluje aktualizację aplikacji przy użyciu funkcji instalowania bezproblemowego. Aktualizację można pobrać i uruchomić za pośrednictwem logiki aplikacji, wyświetlając w razie potrzeby własny interfejs użytkownika, niezależnie od tego, czy dla `customUpdateUI` ustawiona jest wartość `true`.*

Więcej informacji zawiera sekcja „[Aktualizowanie aplikacji AIR](#)” na stronie 273.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

`true` lub `false` (domyślnie)

Przykład

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 i nowsze wersje — element opcjonalny

Wskazuje, że aplikacja wymaga zastosowania bufora głębi lub bufora szablonu. Te bufory są zazwyczaj używane podczas pracy z zawartością 3D. Domyślnie ten element ma wartość `false` w celu wyłączenia bufora głębi i bufora szablonu. Ten element jest potrzebny, gdyż bufory należy przydzielić podczas uruchamiania aplikacji — przed wczytaniem zawartości.

Ustawienie tego elementu musi odpowiadać wartości przekazanej w argumencie `enableDepthAndStencil` do metody `Context3D.configureBackBuffer()`. Jeśli te wartości nie są zgodne, środowisko AIR generuje błąd.

Ten element ma zastosowanie jedynie w sytuacji, gdy `renderMode = direct`. Jeśli właściwość `renderMode` nie ma wartości `direct`, narzędzie ADT generuje błąd 118.

```
<depthAndStencil> element unexpected for render mode cpu. It requires "direct" render mode.
```

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

`true` lub `false` (domyślnie)

Przykład

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

description

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Opis aplikacji wyświetlany w instalatorze aplikacji AIR.

Po określeniu pojedynczego węzła `text` (zamiast wielu elementów `text`) instalator aplikacji AIR korzysta z tego opisu bez względu na język systemu. W przeciwnym razie instalator aplikacji AIR korzysta z opisu, który jest najbliższy językowi interfejsu systemu operacyjnego użytkownika. Załóżmy na przykład, że w instalacji element `description` pliku deskryptora aplikacji zawiera wartość ustawień regionalnych `en` (angielski). Instalator aplikacji AIR korzysta z opisu `en` pod warunkiem, że system użytkownika identyfikuje `en` (angielski) jako język interfejsu użytkownika. Ten instalator korzysta z opisu `en` również wówczas, gdy język interfejsu systemu operacyjnego to `en-US` (angielski USA). Jeśli jednak język interfejsu użytkownika to `en-US`, a plik deskryptora aplikacji definiuje nazwy `en-US` i `en-GB`, wówczas instalator aplikacji AIR korzysta z wartości `en-US`. Jeśli aplikacja nie definiuje opisu zgodnego z językiem interfejsu użytkownika systemu, wówczas instalator aplikacji AIR korzysta z pierwszej wartości `description` zdefiniowanej w pliku deskryptora aplikacji.

Więcej informacji na temat programowania aplikacji wielojęzycznych zawiera sekcja „[Lokalizowanie aplikacji AIR](#)” na stronie 309.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: „[text](#)” na stronie 254

Zawartość

Schemat deskryptora aplikacji AIR 1.0 zezwala na zdefiniowanie tylko jednego prostego węzła `text` dla nazwy (zamiast wielu elementów `text`).

W środowisku AIR 1.1 (lub nowszym) można określić wiele języków w elemencie `description`. Atrybut `xml:lang` dla każdego elementu `text` określa kod języka zdefiniowany w normie [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Przykład

Opis prostego węzła `text`:

```
<description>This is a sample AIR application.</description>
```

Opis ze zlokalizowanymi elementami `text` dla języka angielskiego, francuskiego i hiszpańskiego (obowiązuje w środowisku AIR 1.1 lub nowszym):

```
<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 i nowsze wersje — element wymagany

Opis typu pliku jest wyświetlany użytkownikowi przez system operacyjny. Opisu typu pliku nie można lokalizować.

Dodatkowe informacje zawiera sekcja Element „[description](#)” na stronie 230 jako element potomny elementu `application`.

Element macierzysty: „[fileType](#)” na stronie 235

Elementy potomne: brak

Zawartość

Ciąg opisujący zawartość pliku.

Przykład

```
<description>PNG image</description>
```

embedFonts

Umożliwia zastosowanie czcionek niestandardowych do obiektu `StageText` w aplikacji AIR. Ten element jest opcjonalny.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: „[czcionka](#)” na stronie 236

Zawartość

Element `embedFonts` może zawierać dowolną liczbę elementów czcionek.

Przykład

```
<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

AIR 3.1 i nowsze wersje — wyłącznie iOS, element opcjonalny

W systemie iOS są używane właściwości określane jako uprawnienia, które pozwalają aplikacji uzyskać dostęp do dodatkowych zasobów i funkcji. W aplikacji dla urządzeń przenośnych z systemem iOS można podać te informacje za pomocą elementu Entitlements.

Element macierzysty: „[iPhone](#)” na stronie 244

Elementy potomne: Elementy pliku Entitlements.plist systemu iOS

Zawartość

Zawiera elementy potomne określające pary klucz-wartość do użycia jako ustawienia Entitlements.plist na potrzeby aplikacji. Zawartość elementu Entitlements powinna być zawarta w bloku CDATA. Więcej informacji znajduje się w materiale [Dokumentacja klucza Entitlement](#) w bibliotece dla programistów systemu iOS.

Przykład

```
<iphone>
...
  <Entitlements>
    <![CDATA[
      <key>aps-environment</key>
      <string>development</string>
    ]]>
  </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 i nowsze wersje — element wymagany

Ciąg rozszerzenia typu pliku.

Element macierzysty: „[fileType](#)” na stronie 235

Elementy potomne: brak

Zawartość

Ciąg określający znaki rozszerzenia pliku (bez kropki, „”).

Przykład

```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 i nowsze wersje

Określa identyfikator rozszerzenia języka ActionScript używanego przez aplikację. Ten identyfikator jest definiowany w dokumencie deskryptora aplikacji.

Element macierzysty: „[extensions](#)” na stronie 233

Elementy potomne: brak

Zawartość

Ciąg określający identyfikator rozszerzenia języka ActionScript.

Przykład

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

extensions

Adobe AIR 2.5 i nowsze wersje — element opcjonalny

Określa rozszerzenia języka ActionScript używane przez aplikację.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: „[extensionID](#)” na stronie 233

Zawartość

Elementy potomne `extensionID` zawierające identyfikatory rozszerzeń języka ActionScript z pliku deskryptora rozszerzeń.

Przykład

```
<extensions>
  <extensionID>extension.first</extensionID>
  <extensionID>extension.next</extensionID>
  <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

AIR 3.7 i nowsze wersje (tylko iOS) — element opcjonalny

Określa nazwę pliku tekstowego, który zawiera listę plików SWF do skonfigurowania przez ADT na potrzeby zdalnego przechowywania. Aby zminimalizować wstępny rozmiar pobieranej aplikacji, można spakować podzbiór plików SWF używanych przez aplikację, a do wczytywania pozostałych zewnętrznych plików SWF (zawierających tylko zasoby) stosować metodę `Loader.load()` w czasie wykonywania. W celu skorzystania z tej funkcji trzeba spakować aplikację w taki sposób, aby narzędzie ADT przeniosło cały kod bajtowy ActionScript (ABC) z zewnętrznych plików SWF do głównego pliku SWF aplikacji, co pozwoli uzyskać plik SWF zawierający tylko zasoby. Sposób ten zapewnia zgodność z obowiązującą w sklepie Apple Store regułą, która zabrania pobierania jakiegokolwiek kodu do zainstalowanej aplikacji.

Więcej informacji zawiera dokument „[Minimalizowanie ilości pobieranych danych przez wczytywanie zewnętrznych plików SWF zawierających tylko zasoby](#)” na stronie 92.

Element macierzysty: „[iPhone](#)” na stronie 244, „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Nazwa pliku tekstowego zawierającego rozdzieloną znakami końca wiersza listę plików SWF, które mają być przechowywane na zdalnym serwerze.

Atrybuty:

Brak.

Przykłady

iOS:

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

filename

Adobe AIR 1.0 i nowsze wersje — element wymagany

Ciąg znaków używany jako nazwa pliku aplikacji (bez rozszerzenia) po zainstalowaniu aplikacji. Plik aplikacji uruchamia aplikację AIR w środowisku wykonawczym. Jeśli nie określono wartości `name`, wówczas wartość `filename` jest używana również jako nazwa folderu instalacyjnego.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Właściwość `filename` może zawierać dowolne znaki Unicode (UTF-8) z wyjątkiem poniższych, których stosowanie jest zabronione w nazwach plików w różnych systemach plików:

Znak	Kod szesnastkowy
różne	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
	x7C

Wartość filename nie może kończyć się kropką.

Przykład

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Opisuje pojedynczy typ pliku, który może zostać zarejestrowany dla aplikacji.

Element macierzysty: „fileTypes” na stronie 236

Elementy potomne:

- „contentType” na stronie 228
- „description” na stronie 231
- „extension” na stronie 232
- „icon” na stronie 239
- „name” na stronie 248

Zawartość

Elementy opisujące typ pliku.

Przykład

```
<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</image48x48>
  </icon>
</fileType>
```


fileTypes

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Element `fileTypes` umożliwia deklarowanie typów plików, z którymi można kojarzyć aplikacje AIR.

Po zainstalowaniu aplikacji AIR wszystkie zadeklarowane typy plików zostają zarejestrowane w systemie operacyjnym. Jeśli te typy plików nie są jeszcze skojarzone z inną aplikacją, wówczas są kojarzone z aplikacją AIR. W celu zastąpienia istniejącego skojarzenia między typem pliku a inną aplikacją należy użyć metody `NativeApplication.setAsDefaultApplication()` w środowisku wykonawczym (najlepiej za zgodą użytkownika).

Uwaga: Metody środowiska wykonawczego mogą zarządzać tylko skojarzeniami dla typów plików zadeklarowanych w deskrypcji aplikacji.

Element `fileTypes` jest opcjonalny.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: „[fileType](#)” na stronie 235

Zawartość

Element `fileTypes` może zawierać dowolną liczbę elementów `fileType`.

Przykład

```
<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>
```

czcionka

Opisuje pojedynczą czcionkę niestandardową, jakiej można użyć w aplikacji AIR.

Element macierzysty: „[embedFonts](#)” na stronie 231

Elementy potomne: „[fontName](#)” na stronie 237, „[fontPath](#)” na stronie 237

Zawartość

Elementy określające nazwę oraz ścieżkę czcionki niestandardowej.

Przykład

```
<font>  
  <fontPath>ttf/space age.ttf</fontPath>  
  <fontName>space age</fontName>  
</font>
```

fontName

Określa nazwę czcionki niestandardowej.

element macierzysty: „[czcionka](#)” na stronie 236

Elementy potomne: brak

Zawartość

Nazwa czcionki niestandardowej do zdefiniowania we właściwości StageText.fontFamily

Przykład

```
<fontName>space age</fontName>
```

fontPath

Podaje lokalizację pliku niestandardowych czcionek.

element macierzysty: „[czcionka](#)” na stronie 236

Elementy potomne: brak

Zawartość

Ścieżka pliku czcionki niestandardowej (w odniesieniu do źródła).

Przykład

```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

AIR 3.7 i nowsze wersje (tylko iOS) — element opcjonalny

Wymusza tryb renderowania z użyciem procesora dla określonego zbioru urządzeń. Ta funkcja umożliwia selektywne włączenie trybu renderowania GPU dla pozostałych urządzeń z systemem iOS.

Ten znacznik można dodać jako obiekt potomny znacznika iPhone, wskazując rozdzieloną spacjami listę nazw modeli urządzeń. Prawidłowe nazwy modeli urządzeń:

iPad1,1	iPhone1,1	iPod1,1
iPad2,1	iPhone1,2	iPod2,1
iPad2,2	iPhone2,1	iPod3,3
iPad2,3	iPhone3,1	iPod4,1
iPad2,4	iPhone3,2	iPod5,1
iPad2,5	iPhone4,1	

iPad3,1	iPhone5,1	
iPad3,2		
iPad3,3		
iPad3,4		

Element macierzysty: „[iPhone](#)” na stronie 244, „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Rozdzielona spacjami lista nazw modeli urządzeń.

Atrybuty:

Brak.

Przykłady

iOS:

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

fullScreen

Adobe AIR 2.0 i nowsze wersje, Android i iOS — element opcjonalny

Określa, czy aplikacja jest uruchamiana w trybie pełnoekranowym.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

true lub false (domyślnie)

Przykład

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Początkowa wysokość głównego okna aplikacji.

Jeśli wysokość nie zostanie ustawiona, zostanie określona przez ustawienia w głównym pliku SWF, a w przypadku aplikacji AIR w języku HTML przez system operacyjny.

Maksymalna wysokość okna została zmieniona w środowisku AIR 2 z 2048 pikseli na 4096 pikseli.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Dodatnia liczba całkowita o maksymalnej wartości wynoszącej 4095.

Przykład

```
<height>4095</height>
```

icon

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Właściwość `icon` określa jeden lub większą liczbę plików ikon, jakie mogą być używane dla aplikacji. Dołączenie ikony jest opcjonalne. Jeśli właściwość `icon` nie zostanie określona, wówczas system operacyjny wyświetla ikonę domyślną.

Ścieżka jest określona względem katalogu głównego aplikacji. Pliki ikon muszą być w formacie PNG. Możliwe jest określenie ikon we wszystkich następujących rozmiarach:

Jeśli istnieje element określonego rozmiaru, obraz w pliku musi być dokładnie zgodny z określonym rozmiarem. Jeśli nie są dostępne wszystkie rozmiary, wówczas zmieniana jest skala rozmiaru najbliższego dla określonego zastosowania ikony przez system operacyjny.

Uwaga: *Określone ikony nie są automatycznie dodawane do pakietu AIR. Podczas pakowania aplikacji pliki ikon muszą być dołączane do odpowiednich lokalizacji względnych.*

W celu zapewnienia najlepszych rezultatów należy udostępnić obraz dla każdego z dostępnych rozmiarów. Ponadto należy się upewnić, że ikony wyglądają odpowiednio w trybach koloru 16- i 32-bitowego.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: „[imageNxN](#)” na stronie 240

Zawartość

Element `imageNxN` dla każdego wymaganego rozmiaru ikony.

Przykład

```
<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 i nowsze wersje — element wymagany

Ciąg znaków identyfikatora aplikacji, określany jako identyfikator aplikacji. Często jest używany identyfikator w odwrotnej notacji DNS, ale ten styl nie jest wymagany.

Element macierzysty: „application” na stronie 223

Elementy potomne: brak

Zawartość

Wartość identyfikatora jest ograniczona do następujących znaków:

- 0–9
- a–z
- A–Z
- . (kropka)
- - (kreska)

Wartość musi zawierać od 1 do 212 znaków. Ten element jest wymagany.

Przykład

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Definiuje ścieżkę do ikony względem katalogu aplikacji.

Można używać następujących rozmiarów ikon, przy czym każdy określa inny rozmiar ikony:

- image16x16
- image29x29 (AIR 2 i nowsze wersje)
- image32x32
- image36x36 (AIR 2.5 i nowsze wersje)
- image48x48
- image50x50 (AIR 3.4 i nowsze wersje)
- image57x57 (AIR 2 i nowsze wersje)
- image58x58 (AIR 3.4 i nowsze wersje)
- image72x72 (AIR 2 i nowsze wersje)
- image100x100 (AIR 3.4 i nowsze wersje)
- image114x114 (AIR 2.6 i nowsze wersje)
- image128x128
- image144x144 (AIR 3.4 i nowsze wersje)
- image512x512 (AIR 2 i nowsze wersje)
- image1024x1024 (AIR 3.4 i nowsze wersje)

Ikona musi być grafiką PNG o rozmiarze dokładnie takim, jaki określono w elemencie image. Pliki ikon muszą być dołączone do pakietu aplikacji. Ikony, których dotyczy odniesienia w dokumencie deskryptora aplikacji, nie są automatycznie dołączane.

Element macierzysty: „application” na stronie 223

Elementy potomne: brak

Zawartość

Ścieżka pliku ikony może zawierać dowolne znaki Unicode (UTF-8) z wyjątkiem poniższych, których stosowanie jest zabronione w nazwach plików w różnych systemach plików:

Znak	Kod szesnastkowy
<i>różne</i>	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
	x7C

Przykład

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Umożliwia określanie dodatkowych właściwości aplikacji systemu iOS.

Element macierzysty: „iPhone” na stronie 244

Elementy potomne: Elementy pliku Info.plist systemu iOS

Zawartość

Zawiera elementy potomne określające pary klucz-wartość do użycia jako ustawienia Info.plist na potrzeby aplikacji. Zawartość elementu InfoAdditions powinna być zawarta w bloku CDATA.

Informacje na temat par kluczy i wartości oraz sposobu wyrażania ich w języku XML zawiera dokument [Information Property List Key Reference](#) (Dokumentacja kluczy listy właściwości informacji).

Przykład

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

Więcej tematów Pomocy

„[Ustawienia w systemie iOS](#)” na stronie 87

initialWindow

Adobe AIR 1.0 i nowsze wersje — element wymagany

Definiuje główny plik zawartości i początkowy wygląd aplikacji.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: Jako elementy potomne elementu initialWindow mogą występować wszystkie poniższe elementy. Jednak niektóre elementy są ignorowane w zależności od tego, czy środowisko AIR obsługuje okna na danej platformie.

Element	Komputer	Urządzenie przenośne
„ aspectRatio ” na stronie 226	Ignorowany	Stosowany
„ autoOrients ” na stronie 226	Ignorowany	Stosowany
„ content ” na stronie 228	Stosowany	Stosowany
„ depthAndStencil ” na stronie 230	Stosowany	Stosowany
„ fullScreen ” na stronie 238	Ignorowany	Stosowany
„ height ” na stronie 238	Stosowany	Ignorowany
„ maximizable ” na stronie 246	Stosowany	Ignorowany
„ maxSize ” na stronie 246	Stosowany	Ignorowany
„ minimizable ” na stronie 247	Stosowany	Ignorowany
„ minSize ” na stronie 247	Stosowany	Ignorowany
„ renderMode ” na stronie 250	używane (AIR 3.0 i nowsze wersje)	Stosowany
„ requestedDisplayResolution ” na stronie 250	Używane (AIR 3.6 i nowsze wersje)	Ignorowany
„ resizable ” na stronie 251	Stosowany	Ignorowany
„ softKeyboardBehavior ” na stronie 252	Ignorowany	Stosowany
„ systemChrome ” na stronie 254	Stosowany	Ignorowany
„ title ” na stronie 255	Stosowany	Ignorowany
„ transparent ” na stronie 255	Stosowany	Ignorowany
„ visible ” na stronie 257	Stosowany	Ignorowany
„ width ” na stronie 257	Stosowany	Ignorowany
„ x ” na stronie 257	Stosowany	Ignorowany
„ y ” na stronie 258	Stosowany	Ignorowany

Zawartość

Elementy potomne definiujące zachowanie i wygląd aplikacji.

Przykład

```
<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 i nowsze wersje — element opcjonalny

Identyfikuje podkatalog domyślnego katalogu instalacyjnego.

W systemie Windows domyślnym folderem instalacyjnym jest katalog Program Files. W systemie Mac OS jest to katalog /Applications. W systemie Linux jest to katalog /opt/. Na przykład: jeśli właściwość `installFolder` jest ustawiona na "Acme", a aplikacja ma nazwę "ExampleApp", wówczas aplikacja zostanie zainstalowana w folderze C:\Program Files\Acme\ExampleApp w systemie Windows, w folderze /Applications/Acme/Example.app w systemie MacOS i w folderze /opt/Acme/ExampleApp w systemie Linux.

Właściwość `installFolder` jest opcjonalna. Jeśli właściwość `installFolder` nie zostanie określona, wówczas aplikacja zostanie zainstalowana w domyślnym katalogu instalacyjnym, zgodnie z właściwością `name`.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Właściwość `installFolder` może zawierać dowolne znaki Unicode (UTF-8) z wyjątkiem znaków, których użycie jest zabronione w nazwach folderów w różnych systemach plików. (Listę wyjątków przedstawiono w opisie właściwości `filename`).

Znak ukośnika (/) powinien być używany jako znak separatora katalogu, jeśli jest wymagane określenie podkatalogu zagnieżdżonego.

Przykład

```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0 i nowsze wersje, wyłącznie iOS — element opcjonalny

Definiuje właściwości aplikacji specyficzne dla systemu iOS.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne:

- „[Entitlements](#)” na stronie 232
- „[externalSwfs](#)” na stronie 234
- „[forceCPURenderModeForDevices](#)” na stronie 237
- „[InfoAdditions](#)” na stronie 241
- „[requestedDisplayResolution](#)” na stronie 250

Więcej tematów Pomocy

„[Ustawienia w systemie iOS](#)” na stronie 87

manifest

AIR 2.5 i nowsze wersje, wyłącznie urządzenia z systemem Android — element opcjonalny

Określa informacje, które mają zostać dodane do pliku manifestu systemu Android dla aplikacji.

Element macierzysty: „[manifestAdditions](#)” na stronie 245

Elementy potomne: Elementy zdefiniowane przez zestaw SDK systemu Android.

Zawartość

Element manifestu nie należy w rzeczywistości do schematu deskryptora aplikacji AIR. Jest on głównym elementem dokumentu XML manifestu w systemie Android. Cała zawartość, jaka ma zostać umieszczona w elemencie manifestu, musi być zgodna ze schematem `AndroidManifest.xml`. Podczas tworzenia pliku APK za pomocą narzędzi AIR informacje w elemencie manifestu są kopiowane do odpowiedniej części tworzonego pliku `AndroidManifest.xml` aplikacji.

Jeśli zostaną określone wartości manifestu dostępne tylko w wersji zestawu SDK nowszej niż bezpośrednio obsługiwana przez środowisko AIR, należy ustawić flagę `-platformSDK` na ADT podczas pakowania aplikacji. Flagę należy ustawić na ścieżkę w systemie plików, która wskazuje wersję zestawu SDK systemu Android obsługującą dodawane wartości.

Sam element manifestu musi być ujęty w blok CDATA w deskrytorze aplikacji AIR.

Przykład

```
<![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>  
]]>
```

Więcej tematów Pomocy

„Ustawienia w systemie Android” na stronie 80

[Plik AndroidManifest.xml](#)

manifestAdditions

AIR 2.5 i nowsze wersje, wyłącznie urządzenia z systemem Android

Określa informacje, które mają zostać dodane do pliku manifestu systemu Android.

Każda aplikacja dla systemu Android zawiera plik manifestu, który definiuje podstawowe właściwości aplikacji. Manifest systemu Android jest oparty na podobnym pomysłu co deskryptor aplikacji AIR. Aplikacja AIR dla systemu Android ma zarówno deskryptor aplikacji, jak i automatycznie tworzony plik manifestu systemu Android. Gdy aplikacja AIR dla systemu Android jest pakowana, informacje w tym elemencie `manifestAdditions` są dodawane do odpowiednich części dokumentu manifestu Android.

Element macierzysty: „[android](#)” na stronie 222

Elementy potomne: „[manifest](#)” na stronie 244

Zawartość

Informacje w elemencie `manifestAdditions` są dodawane do dokumentu XML `AndroidManifest`.

W celu zapewnienia prawidłowego działania aplikacji i środowiska wykonawczego wiele wpisów manifestu w tworzonym dokumencie manifestu systemu Android jest ustawianych przez środowisko AIR. Nie można nadpisywać następujących ustawień.

Nie można ustawiać następujących atrybutów elementu `manifest`:

- `package`
- `android:versionCode`
- `android:versionName`

Nie można ustawiać następujących atrybutów dla głównego elementu `activity`:

- `android:label`
- `android:icon`

Nie można ustawiać następujących atrybutów elementu `application`:

- `android:theme`
- `android:name`

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

Przykład

```
<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>
```

Więcej tematów Pomocy

„Ustawienia w systemie Android” na stronie 80

[Plik AndroidManifest.xml](#)

maximizable

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy można maksymalizować okno.

Uwaga: W systemach operacyjnych, w których maksymalizowanie jest operacją zmiany rozmiaru (takich jak Mac OS X), dla elementów `maximizable` i `resizable` należy ustawić wartość `false`, aby zapobiec powiększaniu okna lub zmianie jego wielkości.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

`true` (domyślnie) lub `false`

Przykład

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Maksymalne rozmiary okna. Jeśli rozmiar maksymalny nie zostanie ustawiony, określi go system operacyjny.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Dwie liczby całkowite przedstawiające maksymalną szerokość i wysokość, oddzielone odstępami.

Uwaga: W środowisku AIR 2 maksymalny rozmiar okna obsługiwany przez środowisko AIR został zwiększony z 2048 x 2048 pikseli do 4096 x 4096 pikseli. (Współrzędne ekranu zaczynają się od zera, dlatego maksymalną wartością szerokości lub wysokości, której można użyć, jest 4095).

Przykład

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy można minimalizować okno.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

true (domyślnie) lub false

Przykład

```
<minimizable>>false</minimizable>
```

minSize

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa minimalny rozmiar dozwolony dla okna.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Dwie liczby całkowite przedstawiające minimalną szerokość i wysokość, oddzielone odstępami. Minimalny rozmiar narzucany przez system operacyjny ma pierwszeństwo w stosunku do wartości ustawionej w deskrypcji aplikacji.

Przykład

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Tytuł aplikacji wyświetlany w instalatorze aplikacji AIR.

Jeśli nie określono elementu name, instalator aplikacji AIR wyświetla wartość `filename` jako nazwę pliku.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: „[text](#)” na stronie 254

Zawartość

Po określeniu pojedynczego węzła `text` (zamiast wielu elementów `<text>`) instalator aplikacji AIR korzysta z tej nazwy bez względu na język systemu.

Schemat deskryptora aplikacji AIR 1.0 zezwala na zdefiniowanie tylko jednego prostego węzła `text` dla nazwy (zamiast wielu elementów `text`). W środowisku AIR 1.1 (lub nowszym) można określić wiele języków w elemencie `name`.

Atrybut `xml:lang` dla każdego elementu `text` określa kod języka zdefiniowany w normie [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Instalator aplikacji AIR korzysta z nazwy, która jest najbliższa językowi interfejsu systemu operacyjnego użytkownika. Na przykład: należy rozważyć instalację, w której element `name` pliku deskryptora aplikacji zawiera wartość ustawień narodowych `en` (angielski). Instalator aplikacji AIR korzysta z nazwy `en` pod warunkiem, że system operacyjny identyfikuje `en` (angielski) jako język interfejsu użytkownika. Korzysta z nazwy `en` również wówczas, gdy język interfejsu systemu operacyjnego to `en-US` (angielski USA). Jeśli jednak język interfejsu użytkownika to `en-US`, a plik deskryptora aplikacji definiuje nazwy `en-US` i `en-GB`, wówczas instalator aplikacji AIR korzysta z wartości `en-US`. Jeśli aplikacja nie definiuje nazwy zgodnej z językami interfejsu użytkownika systemu, wówczas instalator aplikacji AIR korzysta z pierwszej wartości `name` zdefiniowanej w pliku deskryptora aplikacji.

Element `name` definiuje tylko tytuł aplikacji używany w instalatorze aplikacji AIR. Instalator aplikacji AIR obsługuje wiele języków: chiński tradycyjny, chiński uproszczony, czeski, holenderski, angielski, francuski, niemiecki, włoski, japoński, koreański, polski, portugalski (Brazylia), rosyjski, hiszpański, szwedzki i turecki. Instalator aplikacji AIR wybiera język wyświetlany (dla tekstu innego niż tytuł i opis aplikacji) na podstawie języka interfejsu użytkownika systemu. Wybór języka jest uzależniony od ustawień w pliku deskryptora aplikacji.

Element `name` *nie* definiuje ustawień narodowych dostępnych dla aplikacji działającej, zainstalowanej. Szczegółowe informacje na temat programowania aplikacji wielojęzycznych zawiera sekcja „[Lokalizowanie aplikacji AIR](#)” na stronie 309.

Przykład

W poniższym przykładzie nazwę zdefiniowano za pomocą prostego węzła `text`.

```
<name>Test Application</name>
```

W poniższym przykładzie dotyczącym środowiska AIR 1.1 (i nowszych wersji) nazwę określono w trzech językach (angielskim, francuskim i hiszpańskim) za pomocą węzłów elementu `<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 i nowsze wersje — element wymagany

Określa nazwę typu pliku.

Element macierzysty: „[fileType](#)” na stronie 235

Elementy potomne: brak

Zawartość

Ciąg przedstawiający nazwę typu pliku.

Przykład

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa lokalizację, w której należy umieścić skróty do aplikacji, jakie będą dostępne w menu Wszystkie programy systemu operacyjnego Windows lub Applications (Aplikacje) systemu Linux. (To ustawienie jest aktualnie ignorowane w innych systemach operacyjnych).

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Ciąg użyty dla właściwości `programMenuFolder` może zawierać dowolne znaki Unicode (UTF-8) z wyjątkiem znaków, których użycie jest zabronione w nazwach folderów w różnych systemach plików. (Listę wyjątków przedstawiono w opisie elementu `filename`). Znak ukośnika (`/`) *nie może* być używany jako ostatni znak w tej wartości.

Przykład

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 i nowsze wersje — element opcjonalny

Określa identyfikator wydawcy w przypadku aktualizacji aplikacji AIR utworzonej pierwotnie za pomocą środowiska AIR 1.5.2 lub starszego.

W przypadku tworzenia aktualizacji aplikacji określa wyłącznie identyfikator wydawcy. Wartość elementu `publisherID` musi być zgodna z identyfikatorem wydawcy utworzonym przez środowisko AIR dla wcześniejszej wersji tej aplikacji. W przypadku zainstalowanej aplikacji identyfikator wydawcy można znaleźć w pliku `META-INF/AIR/publisherid` w folderze, w którym zainstalowano aplikację.

Nowe aplikacje utworzone za pomocą środowiska AIR 1.5.3 lub nowszego nie powinny określać identyfikatora wydawcy.

Więcej informacji zawiera sekcja „[Informacje o identyfikatorach wydawców AIR](#)” na stronie 202.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Ciąg identyfikatora wydawcy.

Przykład

```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 i nowsze wersje — element opcjonalny

Określa, czy jeśli w systemie jest obsługiwane przyspieszanie za pomocą GPU, to ma ono być stosowane.

Element macierzysty: „initialWindow” na stronie 242

Elementy potomne: brak

Zawartość

Jedna z następujących wartości:

- Wartość `auto` (domyślna) obecnie oznacza stosowanie trybu procesora.
- Wartość `cpu` powoduje, że przyspieszanie sprzętowe nie jest stosowane.
- Wartość `direct` oznacza, że tworzenie kompozycji podczas renderowania jest obsługiwane przez procesor, natomiast kopiowanie bitowe korzysta z GPU. Dostępne w środowisku AIR 3 i nowszych wersjach.

Uwaga: Aby można było stosować przyspieszanie GPU zawartości Flash w środowisku AIR dla platform przenośnych, firma Adobe zaleca, aby zamiast opcji `renderMode="direct"` (obiektu `Stage3D`) użyć opcji `renderMode="gpu"`. Firma Adobe oficjalnie oferuje pomoc techniczną dla następujących architektur opartych na obiekcie `Stage3D` i zaleca ich stosowanie: *Starling (2D)* i *Away3D (3D)*. Więcej informacji o architekturach `Stage3D` oraz `Starling/Away3D` znajduje się w dokumencie <http://gaming.adobe.com/getstarted/>.

- Wartość `gpu` powoduje, że przyspieszanie sprzętowe jest stosowane, jeśli jest dostępne.

Ważne: Nie należy używać trybu renderowania GPU w przypadku aplikacji *Flex*.

Przykład

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

requestedDisplayResolution

Adobe AIR 2.6 i nowsze wersje — tylko iOS; Adobe AIR 3.6 i nowsze wersje — OS X, opcjonalnie

Określa, czy na urządzeniu lub komputerze wyposażonym w wyświetlacz o wysokiej rozdzielczości aplikacja dąży do stosowania rozdzielczości standardowej czy wysokiej. Po ustawieniu wartości *standard*, która jest wartością domyślną, ekran jest widoczny dla aplikacji jako ekran o standardowej rozdzielczości. Po ustawieniu wartości *high* aplikacja może odwoływać się do każdego piksela w wysokiej rozdzielczości.

Jeśli na przykład na telefonie iPhone z ekranem o rozdzielczości 640x960 jest ustawiona wartość *standard*, stół montażowy zajmujący cały ekran ma wymiary 320x480, a każdy piksel aplikacji jest renderowany jako cztery piksele ekranu. W przypadku ustawienia *high* stół montażowy zajmujący cały ekran ma wymiary 640x960.

Na urządzeniach z ekranem o standardowej rozdzielczości wymiary stołu montażowego są zgodne z wymiarami ekranu niezależnie od tego, które ustawienie jest używane.

Jeśli element `requestedDisplayResolution` jest zagnieżdżony w elemencie `iPhone`, jest stosowany dla urządzeń z systemem iOS. W takiej sytuacji za pomocą atrybutu `excludeDevices` można wskazać urządzenia, dla których nie jest stosowane to ustawienie.

Jeśli element `requestedDisplayResolution` jest zagnieżdżony w elemencie `initialWindow`, jest stosowany w przypadku aplikacji komputerowych AIR na komputerach MacBook Pro, które obsługują wyświetlacze o wysokiej rozdzielczości. Określona wartość dotyczy wszystkich okien natywnych używanych w aplikacji. Zagnieżdżenie elementu `requestedDisplayResolution` w elemencie `initialWindow` jest obsługiwane od wersji 3.6 środowiska AIR.

Element macierzysty: „iPhone” na stronie 244, „initialWindow” na stronie 242

Elementy potomne: brak

Zawartość

standard (domyślnie) lub *high*

Atrybut:

`excludeDevices` — rozdzielona spacjami lista nazw lub prefiksów nazw modeli urządzeń z systemem iOS. Umożliwia programistom ustawianie wysokiej lub standardowej rozdzielczości zależnie od urządzenia. Ten atrybut jest dostępny tylko w systemie iOS (element `requestedDisplayResolution` jest zagnieżdżony w elemencie `iPhone`). Atrybut `excludeDevices` jest dostępny w środowisku AIR 3.6 lub nowszym.

Na każdym urządzeniu, którego model jest wymieniony w tym atrybucie, wartość `requestedDisplayResolution` jest odwrotna od ustawionej. Jeśli element `requestedDisplayResolution` ma wartość *high*, na wykluczonych urządzeniach jest używana standardowa rozdzielczość. Jeśli element `requestedDisplayResolution` ma wartość *standard*, na wykluczonych urządzeniach jest używana wysoka rozdzielczość.

Wartościami są nazwy i prefiksy nazw modeli urządzeń z systemem iOS. Wartość `iPad3,1` oznacza na przykład tablet iPad Wi-Fi 3. generacji (ale nie modele GSM i CDMA tabletu iPad 3. generacji). Wartość `iPad3` oznacza wszystkie tablety iPad 3. generacji. Nieoficjalna lista modeli urządzeń z systemem iOS jest dostępna na [stronie modeli witryny wiki dotyczącej telefonu iPhone](#).

Przykłady

Komputer:

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

iOS:

```
<iPhone>
  <requestedDisplayResolution excludeDevices="iPad3
iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy można zmieniać rozmiar okna.

Uwaga: W systemach operacyjnych, w których maksymalizowanie jest operacją zmiany rozmiaru (takich jak Mac OS X), dla elementów `maximizable` i `resizable` należy ustawić wartość `false`, aby zapobiec powiększaniu okna lub zmianie jego wielkości.

Element macierzysty: „initialWindow” na stronie 242

Elementy potomne:

Zawartość

`true` (domyślnie) lub `false`

Przykład

```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 i nowsze wersje, profil urządzeń przenośnych — element opcjonalny

Określa domyślny sposób działania aplikacji, gdy jest wyświetlana klawiatura wirtualna. Domyślnym zachowaniem jest przesunięcie aplikacji do góry. Środowisko wykonawcze utrzymuje aktywne pole tekstowe lub obiekt interaktywny na ekranie. Jeśli aplikacja nie posiada własnej logiki obsługi klawiatury, można użyć opcji *pan*.

Można również wyłączyć domyślne zachowanie, ustawiając dla elementu `softKeyboardBehavior` wartość *none*. W takiej sytuacji uniesienie klawiatury programowej spowoduje wywołanie zdarzenia `SoftKeyboardEvent` przez pola tekstowe i obiekty interaktywne, ale środowisko wykonawcze nie przesuwa aplikacji ani nie zmienia jej rozmiaru. To aplikacja będzie odpowiedzialna za zachowanie widoczności obszaru wprowadzania tekstu.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Wartość *none* lub *pan*. Wartością domyślną jest *pan*.

Przykład

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Więcej tematów Pomocy

[SoftKeyboardEvent](#)

supportedLanguages

Adobe AIR 3.2 i nowsze wersje — element opcjonalny

Identyfikuje języki obsługiwane przez aplikację. Ten element jest używany tylko w systemie iOS, środowisku dołączanym dla komputerów Mac i aplikacjach dla systemu Android. Ten element jest ignorowany w aplikacjach wszystkich innych typów.

Jeśli ten element nie zostanie określony, pakowacz domyślnie wykona następujące operacje zależnie od typu aplikacji:

- System iOS: Na liście języków obsługiwanych przez aplikację w sklepie App Store dla systemu iOS zostaną umieszczone wszystkie języki obsługiwane przez środowisko wykonawcze AIR.
- Środowisko dołączane dla komputerów Mac: Aplikacja spakowana razem ze środowiskiem nie będzie zawierać informacji o tłumaczeniach.
- Android: Pakiet aplikacji zawiera zasoby dla wszystkich języków obsługiwanych przez środowisko wykonawcze AIR.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Lista obsługiwanych języków (rozdzielana spacjami). Prawidłowe wartości języków to wartości ISO 639-1 dla języków obsługiwanych przez środowisko wykonawcze AIR: en, de, es, fr, it, ja, ko, pt, ru, cs, nl, pl, sv, tr, zh, da, nb, iw.

Pakowacz generuje błąd w przypadku pustych wartości elementu `<supportedLanguages>`.

Uwaga: Przetłumaczone znaczniki (na przykład znacznik `name`) ignorują wartość języka, jeśli jest używany znacznik `<supportedLanguages>` niezawierający danego języka. Jeśli rozszerzenie natywne zawiera zasoby dla języka, którego nie określono przy użyciu znacznika `<supportedLangauges>`, jest generowane ostrzeżenie i takie zasoby są ignorowane dla danego języka.

Przykład

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 i nowsze wersje — element opcjonalny

Określa profile obsługiwane dla danej aplikacji.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

W elemencie `supportedProfiles` można podać dowolne z tych wartości.

- `desktop` — profil komputera przeznaczony dla aplikacji AIR instalowanych na komputerach przy użyciu plików AIR. Takie aplikacje nie mają dostępu do klasy `NativeProcess` (która służy do komunikacji z aplikacjami macierzystymi).
- `extendedDesktop` — rozszerzony profil komputera przeznaczony dla aplikacji AIR instalowanych na komputerach przy użyciu macierzystego programu instalacyjnego. Takie aplikacje mają dostęp do klasy `NativeProcess` (która służy do komunikacji z aplikacjami macierzystymi).
- `mobileDevice` — profil urządzeń przenośnych przeznaczony dla aplikacji na urządzenia przenośne.
- `extendedMobileDevice` — profil rozszerzony dla urządzenia przenośnego nie jest obecnie używany.

Właściwość `supportedProfiles` jest opcjonalna. Gdy ten element nie zostanie uwzględniony w pliku deskryptora aplikacji, aplikację można skompilować i wdrożyć z dowolnym profilem.

Aby określić wiele profili, należy rozdzielić je spacjami. Na przykład następujące ustawienie określa, że aplikacja może być kompilowana tylko w profilu lokalnym i rozszerzonym:

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

Uwaga: Jeśli aplikacja jest uruchamiana za pomocą narzędzia ADL, a dla opcji `-profile` narzędzia ADL nie określono wartości, wówczas jest używany pierwszy profil w deskrytorze aplikacji. (Jeśli w deskrytorze aplikacji również nie zostały określone żadne profile, wówczas jest używany profil komputera).

Przykład

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

Więcej tematów Pomocy

„[Profil urządzeń](#)” na stronie 259

„[Obsługiwane profile](#)” na stronie 79

systemChrome

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy tworzone początkowe okno aplikacji zawiera standardowy pasek tytułu, krawędzie i elementy sterujące oferowane przez system operacyjny.

W środowisku wykonawczym nie można zmieniać ustawienia karnacji systemowej okna.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Jedna z następujących wartości:

- `none` — brak karnacji systemowej. Za wyświetlanie karnacji okna odpowiada aplikacja (lub środowisko aplikacji, takie jak Flex).
- `standard` (domyślnie) — karnacja systemowa pochodzi z systemu operacyjnego.

Przykład

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 i nowsze wersje — element opcjonalny

Określa ciąg zlokalizowany.

Atrybut `xml:lang` elementu `text` określa kod języka zdefiniowany w normie [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Instalator aplikacji AIR korzysta z elementu `text` wraz z wartością atrybutu `xml:lang`, która najbardziej pasuje do języka interfejsu użytkownika systemu operacyjnego tego użytkownika.

Na przykład w pewnej instalacji element `text` zawiera wartość dla ustawienia regionalnego `en` (angielski). Instalator aplikacji AIR korzysta z nazwy `en` pod warunkiem, że system operacyjny identyfikuje `en` (angielski) jako język interfejsu użytkownika. Korzysta z nazwy `en` również wówczas, gdy język interfejsu systemu operacyjnego to `en-US` (angielski USA). Jeśli jednak język interfejsu użytkownika to `en-US`, a plik deskryptora aplikacji definiuje nazwy `en-US` i `en-GB`, wówczas instalator aplikacji AIR korzysta z wartości `en-US`.

Jeśli aplikacja nie definiuje elementu `text` zgodnego z językami interfejsu użytkownika systemu, wówczas instalator aplikacji AIR korzysta z pierwszej wartości `name` zdefiniowanej w pliku deskryptora aplikacji.

Elementy macierzyste:

- „[name](#)” na stronie 247
- „[description](#)” na stronie 230

Elementy potomne: brak

Zawartość

Atrybut `xml:lang` określający ustawienie regionalne i ciąg zlokalizowanego tekstu.

Przykład

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa tytuł wyświetlany na pasku tytułu początkowego okna aplikacji.

Tytuł jest wyświetlany wyłącznie wówczas, gdy dla elementu `systemChrome` jest ustawiona wartość `standard`.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Ciąg zawierający tytuł okna.

Przykład

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy dla początkowego okna aplikacji jest stosowane mieszanie alfa z pulpitem.

Okno, dla którego włączono przezroczystość, może być wyświetlane wolniej i wymagać większej ilości pamięci. Ustawienia przezroczystości nie można zmienić w środowisku wykonawczym.

***Ważne:** Dla elementu `transparent` można ustawić wartość `true`, gdy element `systemChrome` ma wartość `none`.*

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

`true` lub `false` (domyślnie)

Przykład

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

version

Adobe AIR w wersji od 1.0 do 2.0 — element wymagany; element niedozwolony w środowisku AIR 2.5 lub nowszym

Określa informacje o wersji aplikacji.

Ciąg znaków wersji jest specyfikatorem zdefiniowanym w aplikacji. Środowisko AIR w żaden sposób nie interpretuje ciągu znaków wersji. Dlatego wersja „3.0” nie jest w żaden sposób traktowana jako bardziej aktualna niż wersja „2.0”. Przykłady: „1.0”, „.4”, „0.5”, „4.9”, „1.3.4a”.

W środowisku AIR 2.5 i nowszych wersjach element `version` został zastąpiony elementami `versionNumber` i `versionLabel`.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Ciąg zawierający wersję aplikacji.

Przykład

```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 i nowsze wersje — element opcjonalny

Określa ciąg wersji przeznaczony do odczytu przez użytkowników.

W oknach dialogowych instalacji zamiast wartości elementu `versionNumber` jest wyświetlana wartość etykiety wersji. Jeśli element `versionLabel` nie jest używany, wówczas w obu tych przypadkach jest stosowana wartość elementu `versionNumber`.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Ciąg zawierający wyświetlany publicznie tekst wersji.

Przykład

```
<versionLabel>0.9 Beta</versionLabel>
```

versionNumber

Adobe AIR 2.5 i nowsze wersje — element wymagany

Numer wersji aplikacji.

Element macierzysty: „[application](#)” na stronie 223

Elementy potomne: brak

Zawartość

Numer wersji może zawierać sekwencję składającą się maksymalnie z trzech liczb całkowitych oddzielonych przecinkami. Każda liczba całkowita musi być liczbą z przedziału od 0 do 999 (włącznie).

Przykłady

```
<versionNumber>1.0.657</versionNumber>
```

```
<versionNumber>10</versionNumber>
```

```
<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Określa, czy początkowe okno aplikacji jest widoczne od razu po utworzeniu.

Domyślnie okna aplikacji AIR, łącznie z oknami początkowymi, są tworzone w stanie niewidocznym. Okno można wyświetlić przez wywołanie metody `activate()` obiektu `NativeWindow` lub ustawienie dla właściwości `visible` wartości `true`. Okno główne może być początkowo ukryte — wówczas zmiany położenia i wielkości okna oraz układu składników w oknie nie będą widoczne od razu.

Składnik `mx:WindowedApplication` środowiska Flex automatycznie wyświetla i aktywuje okno bezpośrednio przed wywołaniem zdarzenia `applicationComplete`, chyba że atrybut `visible` w definicji MXML ma wartość `false`.

Na urządzeniach z profilami `mobile`, które nie obsługują okien, ustawienie `visible` jest ignorowane.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

`true` lub `false` (domyślnie)

Przykład

```
<visible>true</visible>
```

width

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Początkowa szerokość głównego okna aplikacji.

Jeśli szerokość nie zostanie ustawiona, zostanie określona przez ustawienia w głównym pliku SWF, a w przypadku aplikacji AIR opartych na języku HTML — przez system operacyjny.

Maksymalna szerokość okna została zmieniona w środowisku AIR 2 z 2048 pikseli na 4096 pikseli.

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Dodatnia liczba całkowita o maksymalnej wartości wynoszącej 4095.

Przykład

```
<width>1024</width>
```

X

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Pozioma pozycja początkowego okna aplikacji.

W większości przypadków lepiej jest pozwolić systemowi operacyjnemu na określenie początkowej pozycji okna, niż przypisać stałą wartość.

Początek układu współrzędnych ekranu (0,0) znajduje się w lewym górnym rogu ekranu (tak jak określa to system operacyjny).

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Wartość całkowita.

Przykład

```
<x>120</x>
```

y

Adobe AIR 1.0 i nowsze wersje — element opcjonalny

Pionowa pozycja początkowego okna aplikacji.

W większości przypadków lepiej jest pozwolić systemowi operacyjnemu na określenie początkowej pozycji okna, niż przypisać stałą wartość.

Początek układu współrzędnych ekranu (0,0) znajduje się w lewym górnym rogu ekranu (tak jak określa to system operacyjny).

Element macierzysty: „[initialWindow](#)” na stronie 242

Elementy potomne: brak

Zawartość

Wartość całkowita.

Przykład

```
<y>250</y>
```

Rozdział 15: Profile urządzeń

Adobe AIR 2 i wersje późniejsze

Profile stanowią mechanizm definiowania klas urządzeń komputerowych, na których działa aplikacja. Profil definiuje zestaw interfejsów API i funkcji obsługiwanych typowo przez określoną klasę urządzeń. Dostępne profile:

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Profile dla aplikacji są definiowane w deskrytorze aplikacji. Użytkownicy komputerów i urządzeń należących do uwzględnionych profili mogą zainstalować daną aplikację, natomiast użytkownicy innych komputerów i urządzeń nie mogą tego zrobić. Jeśli na przykład w deskrytorze aplikacji został uwzględniony tylko profil desktop, użytkownicy mogą instalować i uruchamiać daną aplikację tylko na komputerach stacjonarnych.

Uwzględnienie profilu, który nie jest w rzeczywistości obsługiwany przez aplikację, może spowodować nieodpowiednie działanie aplikacji w pewnych środowiskach. Jeśli w deskrytorze aplikacji nie zostaną określone żadne profile, środowisko AIR nie będzie stosować ograniczeń dla danej aplikacji. Pakiet aplikacji można utworzyć w dowolnym obsługiwanej formie. Użytkownicy urządzeń z dowolnego profilu będą mogli ją zainstalować, lecz aplikacja może działać niewłaściwie.

O ile jest to możliwe, podczas tworzenia pakietu aplikacji są wymuszane ograniczenia związane z profilami. Jeśli na przykład został uwzględniony tylko profil extendedDesktop, nie można utworzyć pakietu aplikacji w postaci pliku AIR, a jedynie pakiet instalatora natywnego. Jeśli nie został uwzględniony profil mobileDevice, nie można utworzyć pakietu aplikacji w postaci pliku APK systemu Android.

Jedno urządzenie komputerowe może obsługiwać więcej niż jeden profil. Na przykład środowisko AIR na komputerach stacjonarnych obsługuje zarówno aplikacje o profilu desktop, jak i extendedDesktop. Jednak aplikacja o profilu extendedDesktop może komunikować się z procesami natywnymi i MUSI być umieszczona w pakiecie instalatora natywnego (exe, dmg, deb lub rpm). Natomiast aplikacja o profilu desktop nie może komunikować się z procesem macierzystym. Taka aplikacja może być umieszczona w pakiecie będącym plikiem AIR lub pakietem instalatora natywnego.

Uwzględnienie określonej funkcji w profilu wskazuje, że jest ona zazwyczaj obsługiwana przez klasę urządzeń, dla której ten profil jest zdefiniowany. Nie oznacza to jednak, że wszystkie urządzenia o danym profilu obsługują każdą funkcję. Na przykład większość telefonów komórkowych (ale nie wszystkie) zawiera przyspieszeniometer. Dla klas i funkcji, które nie są obsługiwane przez wszystkie urządzenia, zazwyczaj jest dostępna właściwość logiczna, którą można sprawdzić przed użyciem danej funkcji. Na przykład w przypadku przyspieszeniometera można sprawdzić właściwość statyczną `Accelerometer.isSupported` w celu ustalenia, czy dane urządzenie zawiera obsługiwany przyspieszeniometer.

Poniższe profile można przypisać do aplikacji AIR za pomocą elementu `supportedProfiles` w deskrytorze aplikacji:

Komputer Profil komputerowy definiuje zestaw funkcji dla aplikacji AIR, które są instalowane jako pliki AIR na komputerach stacjonarnych. Te aplikacje są instalowane i uruchamiane na obsługiwanych platformach stacjonarnych (Mac OS, Windows i Linux). Aplikacje AIR tworzone w środowisku AIR w wersjach wcześniejszych niż AIR 2 mogą być traktowane jako aplikacje o profilu Stacjonarny. Niektóre interfejsy API nie działają w tym profilu. Na przykład: aplikacje o profilu komputerowym nie mogą się komunikować z procesami rodzinnymi.

Rozszerzony komputerowy Profil rozszerzony komputerowy definiuje zestaw funkcji dla aplikacji AIR, które są uwzględniane w pakiecie instalatora natywnego i instalowane przy jego użyciu. Te instalatory macierzyste to pliki EXE w systemie Windows, pliki DMG w systemie Mac OS oraz pliki DEB lub RPM w systemie Linux. Aplikacje o rozszerzonym profilu komputerowym oferują dodatkowe funkcje niedostępne w aplikacjach o profilu komputerowym. Więcej informacji zawiera sekcja „[Pakowanie instalatora natywnego dla komputerów](#)” na stronie 59.

Dla urządzeń przenośnych W profilu dla urządzeń przenośnych zdefiniowano zestaw funkcji aplikacji zainstalowanych na urządzeniach przenośnych, między innymi telefonach komórkowych i tabletach. Te aplikacje można instalować i uruchamiać na obsługiwanych platformach przenośnych, łącznie z systemami Android, BlackBerry Tablet OS oraz iOS.

Rozszerzony dla urządzeń przenośnych W profilu rozszerzonym dla urządzeń przenośnych zdefiniowano rozszerzony zestaw funkcji przeznaczonych dla aplikacji instalowanych na urządzeniach przenośnych. Aktualnie nie ma urządzeń obsługujących ten profil.

Ograniczanie profili docelowych w pliku deskryptora aplikacji

Adobe AIR 2 i wersje późniejsze

W środowisku AIR 2 plik deskryptora aplikacji zawiera element `supportedProfiles`, który umożliwia ograniczanie profili docelowych. Na przykład następujące ustawienie określa, że aplikacja może być kompilowana tylko w profilu komputerowym:

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

Jeśli ten element zostanie ustawiony, pakiety aplikacji będzie można tworzyć tylko dla profili z listy. Należy użyć następujących wartości:

- `desktop` — profil komputerowy
- `extendedDesktop` — profil rozszerzony komputerowy
- `mobileDevice` — profil dla urządzeń przenośnych

Element `supportedProfiles` jest opcjonalny. Gdy ten element nie zostanie uwzględniony w pliku deskryptora aplikacji, pakiet aplikacji będzie można utworzyć dla dowolnego profilu i możliwe będzie wdrożenie aplikacji w dowolnym profilu.

W celu określenia wielu profili w elemencie `supportedProfiles` należy oddzielić każdy znakiem spacji, jak poniżej:

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

Możliwości różnych profili

Adobe AIR 2 i wersje późniejsze

Poniższa tabela zawiera listę klas i funkcji, które są obsługiwane tylko w niektórych profilach.

Klasa lub funkcja	desktop	extendedDesktop	mobileDevice
Accelerometer (Accelerometer.isSupported)	Nie	Nie	Sprawdź
Accessibility (Capabilities.hasAccessibility)	Tak	Tak	Nie
Eliminowanie echa akustycznego (Microphone.getEnhancedMicrophone())	Tak	Tak	Nie
ActionScript 2	Tak	Tak	Nie
Macierz CacheAsBitmap	Nie	Nie	Tak
Camera (Camera.isSupported)	Tak	Tak	Tak
CameraRoll	Nie	Nie	Tak
CameraUI (CameraUI.isSupported)	Nie	Nie	Tak
Dedykowane pakiety ze środowiskiem wykonawczym	Tak	Tak	Tak
ContextMenu (ContextMenu.isSupported)	Tak	Tak	Nie
DatagramSocket (DatagramSocket.isSupported)	Tak	Tak	Tak
DockIcon (NativeApplication.supportsDockIcon)	Sprawdź	Sprawdź	Nie
Przeciąganie i upuszczanie (NativeDragManager.isSupported)	Tak	Tak	Sprawdź
EncryptedLocalStore (EncryptedLocalStore.isSupported)	Tak	Tak	Tak
Moduł Flash Access (DRMManager.isSupported)	Tak	Tak	Nie
GameInput (GameInput.isSupported)	Nie	Nie	Nie
Geolocation (Geolocation.isSupported)	Nie	Nie	Sprawdź
HTMLLoader (HTMLLoader.isSupported)	Tak	Tak	Nie
IME (IME.isSupported)	Tak	Tak	Sprawdź
LocalConnection (LocalConnection.isSupported)	Tak	Tak	Nie
Mikrofon (Microphone.isSupported)	Tak	Tak	Sprawdź
Dźwięk wielokanałowy (Capabilities.hasMultiChannelAudio())	Nie	Nie	Nie
Rozszerzenia natywne	Nie	Tak	Tak
NativeMenu (NativeMenu.isSupported)	Tak	Tak	Nie
NativeProcess (NativeProcess.isSupported)	Nie	Tak	Nie
NativeWindow (NativeWindow.isSupported)	Tak	Tak	Nie
NetworkInfo (NetworkInfo.isSupported)	Tak	Tak	Sprawdź

Klasa lub funkcja	desktop	extendedDesktop	mobileDevice
Otwieranie plików za pomocą aplikacji domyślnej	Ograniczone	Tak	Nie
PrintJob (PrintJob.isSupported)	Tak	Tak	Nie
SecureSocket (SecureSocket.isSupported)	Tak	Tak	Sprawdź
ServerSocket (ServerSocket.isSupported)	Tak	Tak	Tak
Shader	Tak	Tak	Ograniczone
Stage3D (Stage.stage3Ds.length)	Tak	Tak	Tak
Orientacja stołu montażowego (Stage.supportsOrientationChange)	Nie	Nie	Tak
StageVideo	Nie	Nie	Sprawdź
StageWebView (StageWebView.isSupported)	Tak	Tak	Tak
Uruchamianie aplikacji podczas logowania (NativeApplication.supportsStartAtLogin)	Tak	Tak	Nie
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Tak	Tak	Nie
Tryb bezczynności systemu	Nie	Nie	Tak
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Sprawdź	Sprawdź	Nie
Wprowadzanie przy użyciu architektury Text Layout Framework	Tak	Tak	Nie
Updater (Updater.isSupported)	Tak	Nie	Nie
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Tak	Tak	Nie

Znaczenie pozycji w tabeli:

- *Sprawdź* — funkcja jest obsługiwana przez niektóre, ale nie wszystkie urządzenia o określonym profilu. Przed użyciem danej funkcji w czasie wykonywania należy sprawdzić, czy jest ona obsługiwana.
- *Ograniczone* — funkcja jest obsługiwana, lecz występują znaczące ograniczenia. Więcej informacji zawiera odpowiednia dokumentacja.
- *Nie* — funkcja nie jest obsługiwana w określonym profilu.
- *Tak* — funkcja jest obsługiwana w określonym profilu. Poszczególne urządzenia komputerowe mogą nie zawierać sprzętu wymaganego dla danej funkcji. Na przykład nie wszystkie telefony są wyposażone w kamerę.

Określanie profili podczas debugowania za pomocą narzędzia ADL

Adobe AIR 2 i wersje późniejsze

ADL sprawdza, czy w elemencie `supportedProfiles` pliku deskryptora aplikacji określono obsługiwane profile. Jeśli tak, wówczas podczas debugowania ADL domyślnie korzysta z pierwszego obsługiwanego profilu z listy.

Profil dla sesji debugowania ADL można określić za pomocą argumentu `-profile` wiersza poleceń. Więcej informacji zawiera sekcja „[AIR Debug Launcher \(ADL\)](#)” na stronie 169. Ten argument może być używany bez względu na to, czy profil określono w elemencie `supportedProfiles` w pliku deskryptora aplikacji. Jeśli jednak element `supportedProfiles` zostanie określony, musi zawierać profil określony w wierszu poleceń. W przeciwnym wypadku ADL wygeneruje błąd.

Rozdział 16: Interfejs API środowiska w przeglądarce — AIR.SWF

Dostosowywanie pliku badge.swf instalacji bezproblemowej

Oprócz korzystania z pliku badge.swf udostępnionego w SDK możliwe jest również utworzenie własnego pliku SWF do użytku na stronie przeglądarki. Niestandardowy plik SWF może wchodzić w interakcje ze środowiskiem wykonawczym na następujące sposoby:

- Może zainstalować aplikację AIR. Patrz „[Instalowanie aplikacji AIR z przeglądarki](#)” na stronie 270.
- Może sprawdzić, czy konkretna aplikacja AIR została zainstalowana. Patrz „[Sprawdzanie ze strony internetowej, czy aplikacja AIR została zainstalowana](#)” na stronie 269.
- Może sprawdzić, czy zainstalowane jest środowisko wykonawcze. Patrz „[Sprawdzanie, czy środowisko wykonawcze jest zainstalowane](#)” na stronie 268.
- Może uruchomić zainstalowaną aplikację AIR w systemie użytkownika. Patrz „[Uruchamianie zainstalowanej aplikacji AIR z przeglądarki](#)” na stronie 271.

Wszystkie te funkcje stają się dostępne po wywołaniu interfejsów API w pliku SWF, który znajduje się w witrynie adobe.com: air.swf. Istnieje możliwość dostosowania pliku badge.swf i wywoływania elementów interfejsu API pliku air.swf z własnego pliku SWF użytkownika.

Ponadto plik SWF działający w przeglądarce może komunikować się z działającą aplikacją AIR za pomocą klasy LocalConnection. Więcej informacji zawiera sekcja [Komunikacja z innymi wystąpieniami programu Flash Player i środowiska AIR](#) (dla programistów ActionScript) oraz sekcja [Komunikacja z innymi wystąpieniami programu Flash Player i środowiska AIR](#) (dla programistów HTML).

Ważne: Funkcje opisane w tej sekcji (a także interfejsy API w pliku air.swf) wymagają zainstalowania w przeglądarce internetowej w systemie Windows lub Mac OS aktualizacji numer 3 lub nowszej programu Adobe® Flash® Player 9. W systemie Linux funkcja instalacji bezproblemowej wymaga programu Flash Player 10 (wersja 10,0,12,36 lub nowsza). Możliwe jest napisanie kodu w celu sprawdzenia zainstalowanej wersji programu Flash Player, a także udostępnienie alternatywnego interfejsu dla użytkownika, jeśli wymagana wersja programu Flash Player nie jest zainstalowana. Na przykład: jeśli zainstalowana jest starsza wersja programu Flash Player, można udostępnić odsyłacz do wersji pliku AIR przeznaczonej do pobrania (zamiast używania pliku badge.swf lub interfejsu API air.swf w celu zainstalowania aplikacji).

Korzystanie z pliku badge.swf w celu zainstalowania aplikacji

Pakiety AIR SDK i Flex SDK zawierają plik badge.swf, który umożliwia łatwe korzystanie z funkcji instalacji bezproblemowej. Plik badge.swf może zainstalować środowisko wykonawcze i aplikację AIR za pośrednictwem odsyłacza na stronie internetowej. Plik badge.swf oraz jego kod źródłowy są dostępne do dystrybucji na stronie internetowej.

Osadzanie pliku badge.swf na stronie internetowej

- 1 Zlokalizuj poniższe pliki, dostępne w katalogu samples/badge pakietu Flex SDK lub AIR SDK, a następnie dodaj je do serwera sieci Web.
 - badge.swf
 - default_badge.html
 - AC_RunActiveContent.js
- 2 Otwórz plik default_badge.html w edytorze tekstu.
- 3 Na stronie default_badge.html, w funkcji `AC_FL_RunContent()` JavaScript, dostosuj definicje parametru `FlashVars` dla następujących parametrów:

Parametr	Opis
appname	Nazwa aplikacji wyświetlana przez plik SWF, gdy środowisko wykonawcze nie jest zainstalowane.
appurl	(Wymagane). Adres URL pliku AIR, który ma zostać pobrany. Należy użyć adresu URL bezwzględnego, a nie względnego.
airversion	(Wymagane). Dla wersji 1.0 środowiska wykonawczego, należy ustawić tę wartość na 1.0.
imageurl	Adres URL obrazu (opcjonalnie) do wyświetlenia w identyfikatorze.
buttoncolor	Kolor przycisku pobierania (określony jako wartość szesnastkowa, na przykład <code>FFCC00</code>).
messagecolor	Kolor komunikatu tekstowego wyświetlanego poniżej przycisku, gdy środowisko wykonawcze nie zostało zainstalowane (określony jako wartość szesnastkowa, na przykład <code>FFCC00</code>).

- 4 Minimalna wielkość pliku badge.swf to 217 pikseli szerokości i 180 pikseli wysokości. Dostosuj wartości parametrów `width` i `height` funkcji `AC_FL_RunContent()` w celu dostosowania ich do własnych potrzeb.
- 5 Zmień nazwę pliku default_badge.html i dostosuj jego kod (albo dołącz go do innej strony HTML).

Uwaga: Dla znacznika HTML `embed`, który ładuje plik badge.swf, nie należy ustawiać atrybutu `wmode`; należy pozostawić ustawienie domyślne ("`window`"). Inne ustawienia `wmode` uniemożliwią przeprowadzenie instalacji w niektórych systemach. Ponadto użycie innych ustawień `wmode` wywołuje błąd: „Błąd nr 2044: nieobsługiwane zdarzenie `ErrorEvent::text=Błąd nr 2074: Stół montażowy jest za mały, by zmieścić się na nim interfejs użytkownika służący do pobierania.`”

Możliwa jest także edycja i ponowna kompilacja pliku badge.swf. Szczegółowe informacje zawiera rozdział „[Modyfikowanie pliku badge.swf](#)” na stronie 266.

Instalowanie aplikacji AIR za pośrednictwem odsyłacza instalacji bezproblemowej ze strony internetowej

Po dodaniu do strony odsyłacza do instalacji bezproblemowej użytkownik może zainstalować aplikację AIR poprzez kliknięcie odsyłacza w pliku SWF.

- 1 W przeglądarce internetowej, w której jest zainstalowany program Flash Player (wersja 9 z aktualizacją 3 lub nowszą w systemach Windows i Mac OS albo wersja 10 w systemie Linux), przejdź na stronę HTML.
- 2 Na stronie internetowej kliknij odsyłacz w pliku badge.swf.
 - Jeśli zainstalowano środowisko wykonawcze, przejdź do kolejnego kroku.
 - Jeśli środowisko wykonawcze nie zostało zainstalowane, pojawi się okno dialogowe z zapytaniem o to, czy wymagane jest zainstalowanie środowiska. Zainstaluj środowisko wykonawcze (patrz „[Instalowanie środowiska Adobe AIR](#)” na stronie 3), a następnie przejdź do kolejnego kroku.

3 W oknie instalacji pozostaw ustawienia domyślne i kliknij przycisk kontynuowania.

W systemie Windows środowisko AIR automatycznie wykonuje następujące operacje:

- Instaluje aplikację w katalogu `c:\Program Files\`
- Tworzy skrót pulpituowy dla aplikacji
- Tworzy skrót w menu Start
- Dodaje pozycję dotyczącą aplikacji do obszaru Dodaj/Usuń Programy w panelu sterowania

W systemie Mac OS instalator dodaje aplikację do katalogu Applications (na przykład do katalogu /Applications w systemie Mac OS).

W systemie Linux środowisko AIR automatycznie wykonuje następujące operacje:

- Instaluje aplikację w katalogu `/opt`.
- Tworzy skrót pulpituowy dla aplikacji
- Tworzy skrót w menu Start
- Dodaje wpis aplikacji do systemowego menedżera pakietów.

4 Wybierz żądane opcje, a następnie kliknij przycisk Instaluj.

5 Po zakończeniu instalowania kliknij przycisk Zakończ.

Modyfikowanie pliku `badge.swf`

Zestawy SDK środowisk Flex i AIR zawierają pliki źródłowe dla pliku `badge.swf`. Te pliki znajdują się w folderze `samples/badge` pakietu SDK:

Pliki źródłowe	Opis
<code>badge fla</code>	Plik źródłowy Flash, który służy do kompilowania pliku <code>badge.swf</code> . Wynikiem kompilacji pliku <code>badge fla</code> jest plik SWF 9 (który można załadować w programie Flash Player).
<code>AIRBadge.as</code>	Klasa ActionScript 3.0, która definiuje klasę podstawową używaną w pliku <code>badge fla</code> .

W celu zmiany interfejsu pliku `badge fla` można użyć programu Flash Professional.

Funkcja konstruktora `AIRBadge()` zdefiniowana w klasie `AIRBadge` wczytuje plik `air.swf` dostępny na stronie <http://airdownload.adobe.com/air/browserapi/air.swf>. Plik `air.swf` zawiera kod, który umożliwia korzystanie z funkcji instalacji bezproblemowej.

Metoda `onInit()` (w klasie `AIRBadge`) zostaje wywołana po pomyślnym załadowaniu pliku `air.swf`:

```
private function onInit(e:Event):void {
    _air = e.target.content;
    switch (_air.getStatus()) {
        case "installed" :
            root.statusMessage.text = "";
            break;
        case "available" :
            if (_appName && _appName.length > 0) {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                    + _messageColor + "'>In order to run " + _appName +
                    ", this installer will also set up Adobe® AIR®.</font></p>";
            } else {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                    + _messageColor + "'>In order to run this application, "
                    + "this installer will also set up Adobe® AIR®.</font></p>";
            }
            break;
        case "unavailable" :
            root.statusMessage.htmlText = "<p align='center'><font color='#"
                + _messageColor
                + "'>Adobe® AIR® is not available for your system.</font></p>";
            root.buttonBg_mc.enabled = false;
            break;
    }
}
```

Kod ustawia dla globalnej zmiennej `_air` główną klasę załadowanego pliku `air.swf`. Klasa zawiera następujące metody publiczne, do których plik `badge.swf` uzyskuje dostęp w celu wywołania funkcji instalacji bezproblemowej:

Metoda	Opis
<code>getStatus()</code>	<p>Określa, czy na komputerze zainstalowane jest środowisko wykonawcze (lub czy może zostać zainstalowane). Szczegółowe informacje zawiera sekcja „Sprawdzenie, czy środowisko wykonawcze jest zainstalowane” na stronie 268.</p> <ul style="list-style-type: none"> <code>runtimeVersion</code> — ciąg znaków oznaczający wersję środowiska wykonawczego (na przykład "1.0.M6") wymaganego do zainstalowania aplikacji.
<code>installApplication()</code>	<p>Instaluje określoną aplikację na komputerze użytkownika. Szczegółowe informacje zawiera sekcja „Instalowanie aplikacji AIR z przeglądarki” na stronie 270.</p> <ul style="list-style-type: none"> <code>url</code> — ciąg znaków definiujący adres URL. Należy użyć adresu URL bezwzględnego, a nie względnego. <code>runtimeVersion</code> — ciąg znaków oznaczający wersję środowiska wykonawczego (na przykład "2.5.") wymaganego do zainstalowania aplikacji. <code>arguments</code> — argumenty, jakie zostaną przekazane do aplikacji, jeśli zostanie uruchomiona po zainstalowaniu. Aplikacja jest uruchamiana po zainstalowaniu, jeśli element <code>allowBrowserInvocation</code> ma wartość <code>true</code> w pliku deskryptora aplikacji. (Więcej informacji na temat pliku deskryptora aplikacji zawiera rozdział „Pliki deskryptora aplikacji AIR” na stronie 217). Jeśli aplikacja zostanie uruchomiona w wyniku bezproblemowej instalacji z przeglądarki (użytkownik wybrał opcję uruchomienia po instalacji), wówczas obiekt <code>NativeApplication</code> aplikacji wywołuje zdarzenie <code>BrowserInvokeEvent</code> tylko po przekazaniu argumentów. Należy rozważyć, czy dane przekazywane do aplikacji nie spowodują zagrożenia. Szczegółowe informacje zawiera sekcja „Uruchamianie zainstalowanej aplikacji AIR z przeglądarki” na stronie 271.

Ustawienia `url` i `runtimeVersion` są przekazywane do pliku SWF za pośrednictwem ustawień FlashVars na stronie HTML kontenera.

Jeśli aplikacja zostanie uruchomiona bezpośrednio po zainstalowaniu, można skorzystać z komunikacji LocalConnection, aby wywołana zainstalowana aplikacja skontaktowała się z plikiem badge.swf. Więcej informacji zawiera sekcja [Komunikacja z innymi wystąpieniami programu Flash Player i środowiska AIR](#) (dla programistów ActionScript) oraz sekcja [Komunikacja z innymi wystąpieniami programu Flash Player i środowiska AIR](#) (dla programistów HTML).

Możliwe jest również wywołanie metody `getApplicationVersion()` pliku `air.swf` w celu sprawdzenia, czy aplikacja jest zainstalowana. Tę metodę można wywołać przed zainstalowaniem aplikacji lub po uruchomieniu instalowania. Szczegółowe informacje zawiera sekcja „[Sprawdzanie ze strony internetowej, czy aplikacja AIR została zainstalowana](#)” na stronie 269.

Ładowanie pliku air.swf

Możliwe jest utworzenie własnego pliku SWF, który będzie korzystał z interfejsów API pliku `air.swf` w celu interakcji ze środowiskiem wykonawczym i aplikacjami AIR ze strony internetowej w przeglądarce. Plik `air.swf` jest dostępny na stronie <http://airdownload.adobe.com/air/browserapi/air.swf>. W celu utworzenia odwołania do interfejsów API `air.swf` z pliku SWF należy załadować plik `air.swf` do tej samej domeny aplikacji, co plik SWF. Poniższy kod stanowi przykład ładowania pliku `air.swf` do domeny aplikacji ładującego pliku SWF:

```
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;
}
```

Po załadowaniu pliku `air.swf` (gdzie obiekt `contentLoaderInfo` obiektu `Loader` wywoła zdarzenie `init`) można wywołać dowolny element interfejsu API pliku `air.swf`, tak jak opisano to w następnych sekcjach.

Uwaga: Plik `badge.swf`, dostępny w pakiecie AIR SDK i Flex SDK, automatycznie ładuje plik `air.swf`. Patrz „[Korzystanie z pliku badge.swf w celu zainstalowania aplikacji](#)” na stronie 264. Instrukcje w niniejszej sekcji dotyczą tworzenia własnego pliku SWF, który ładuje plik `air.swf`.

Sprawdzanie, czy środowisko wykonawcze jest zainstalowane

Plik SWF może sprawdzić, czy środowisko wykonawcze jest zainstalowane — w tym celu należy wywołać metodę `getStatus()` w pliku `air.swf` załadowanym ze strony <http://airdownload.adobe.com/air/browserapi/air.swf>. Szczegółowe informacje zawiera sekcja „[Ładowanie pliku air.swf](#)” na stronie 268.

Po załadowaniu pliku `air.swf` plik SWF może wywołać metodę `getStatus()` pliku `air.swf`, jak w poniższym kodzie:

```
var status:String = airSWF.getStatus();
```

Metoda `getStatus()` zwraca jeden z następujących ciągów znaków w zależności od statusu środowiska wykonawczego na komputerze:

Ciąg znaków	Opis
"available"	Środowisko wykonawcze może zostać zainstalowane na tym komputerze, ale aktualnie nie jest zainstalowane.
"unavailable"	Środowisko wykonawcze nie może zostać zainstalowane na komputerze.
"installed"	Środowisko wykonawcze jest już zainstalowane na komputerze.

Metoda `getStatus()` generuje błąd, jeśli w przeglądarce nie jest zainstalowana wymagana wersja programu Flash Player (wersja 9 z aktualizacją 3 lub nowszą w systemach Windows i Mac OS albo wersja 10 w systemie Linux).

Sprawdzanie ze strony internetowej, czy aplikacja AIR została zainstalowana

Plik SWF może sprawdzić, czy zainstalowana jest aplikacja AIR (ze zgodnym identyfikatorem aplikacji i wydawcy) — w tym celu musi wywołać metodę `getApplicationVersion()` w pliku `air.swf` załadowanym ze strony <http://airdownload.adobe.com/air/browserapi/air.swf>. Szczegółowe informacje zawiera sekcja „[Ładowanie pliku air.swf](#)” na stronie 268.

Po załadowaniu pliku `air.swf` plik SWF może wywołać metodę `getApplicationVersion()` pliku `air.swf`, jak w poniższym kodzie:

```
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.
    }
}
```

Metoda `getApplicationVersion()` ma następujące parametry:

Parametry	Opis
appID	Identyfikator aplikacji. Szczegółowe informacje zawiera opis właściwości „id” na stronie 239.
pubID	Identyfikator wydawcy dla aplikacji. Szczegółowe informacje zawiera opis właściwości „publisherID” na stronie 249. Jeśli konkretna aplikacja nie zawiera identyfikatora wydawcy, ustaw parametr pubID na pusty ciąg znaków (“”).
callback	Funkcja wywołania zwrotnego, która służy jako funkcja modułu obsługi. Metoda <code>getApplicationVersion()</code> działa asynchronicznie, a po wykryciu zainstalowanej wersji (lub braku zainstalowanej wersji) wywoływana jest ta metoda wywołania zwrotnego. Definicja metody wywołania zwrotnego musi zawierać jeden parametr — ciąg znaków — który określa wersję zainstalowanej aplikacji. Jeśli aplikacja nie została zainstalowana, do funkcji przekazywana jest wartość null, co ilustruje poprzedni fragment kodu.

Metoda `getApplicationVersion()` generuje błąd, jeśli w przeglądarce nie jest zainstalowana wymagana wersja programu Flash Player (wersja 9 z aktualizacją 3 lub nowszą w systemach Windows i Mac OS albo wersja 10 w systemie Linux).

Uwaga: *Począwszy od wersji AIR 1.5.3 identyfikator wydawcy nie jest używany. Identyfikatory wydawców nie są przypisywane do aplikacji automatycznie. W celu zapewnienia zgodności wstecz aplikacje nadal mogą określać identyfikator wydawcy.*

Instalowanie aplikacji AIR z przeglądarki

Plik SWF może zainstalować aplikację AIR poprzez wywołanie metody `installApplication()` w pliku `air.swf` załadowanym ze strony <http://airdownload.adobe.com/air/browserapi/air.swf>. Szczegółowe informacje zawiera sekcja „Ładowanie pliku `air.swf`” na stronie 268.

Po załadowaniu pliku `air.swf` plik SWF może wywołać metodę `installApplication()` pliku `air.swf`, jak w poniższym kodzie:

```
var url:String = "http://www.example.com/myApplication.air";  
var runtimeVersion:String = "1.0";  
var arguments:Array = ["launchFromBrowser"]; // Optional  
airSWF.installApplication(url, runtimeVersion, arguments);
```

Metoda `installApplication()` zainstaluje określoną aplikację na komputerze użytkownika. Ta metoda ma następujące parametry:

Parametr	Opis
url	Ciąg znaków określający adres URL pliku AIR do zainstalowania. Należy użyć adresu URL bezwzględnego, a nie względnego.
runtimeVersion	Ciąg znaków oznaczający wersję środowiska wykonawczego (np. „1.0”) wymaganego do zainstalowania aplikacji.
arguments	Tablica argumentów, jakie zostaną przekazane do aplikacji, jeśli zostanie uruchomiona po zainstalowaniu. W argumentach są rozpoznawane tylko znaki alfanumeryczne. Jeśli zachodzi potrzeba przekazania innych wartości, należy rozważyć użycie odpowiedniego systemu kodowania. Aplikacja jest uruchamiana po zainstalowaniu, jeśli element <code>allowBrowserInvocation</code> ma wartość <code>true</code> w pliku deskryptora aplikacji. (Więcej informacji na temat pliku deskryptora aplikacji zawiera rozdział „ Pliki deskryptora aplikacji AIR ” na stronie 217). Jeśli aplikacja zostanie uruchomiona w wyniku bezproblemowej instalacji z przeglądarki (użytkownik wybrał opcję uruchomienia po instalacji), wówczas obiekt <code>NativeApplication</code> aplikacji wywołuje zdarzenie <code>BrowserInvokeEvent</code> tylko po przekazaniu argumentów. Szczegółowe informacje zawiera sekcja „ Uruchamianie zainstalowanej aplikacji AIR z przeglądarki ” na stronie 271.

Metoda `installApplication()` może działać wyłącznie wówczas, gdy zostanie wywołana w module obsługi zdarzeń, np. w przypadku zdarzenia myszy.

Metoda `installApplication()` generuje błąd, jeśli w przeglądarce nie jest zainstalowana wymagana wersja programu Flash Player (wersja 9 z aktualizacją 3 lub nowszą w systemach Windows i Mac OS albo wersja 10 w systemie Linux).

W celu zainstalowania zaktualizowanej wersji aplikacji w systemie Mac OS użytkownik powinien mieć odpowiednie uprawnienia dostępu do instalowania w katalogu aplikacji (a także uprawnienia administracyjne, jeśli aplikacja aktualizuje środowisko wykonawcze). W systemie Windows użytkownik potrzebuje uprawnień administratora.

Możliwe jest również wywołanie metody `getApplicationVersion()` pliku `air.swf` w celu sprawdzenia, czy aplikacja jest już zainstalowana. Tę metodę można wywołać przed zainstalowaniem aplikacji lub po uruchomieniu instalowania. Szczegółowe informacje zawiera sekcja „[Sprawdzenie ze strony internetowej, czy aplikacja AIR została zainstalowana](#)” na stronie 269. Uruchomiona aplikacja może się komunikować z treścią pliku SWF w przeglądarce — w tym celu może korzystać z klasy `LocalConnection`. Więcej informacji zawiera sekcja [Komunikacja z innymi instancjami Flash Player i AIR](#) (dla programistów ActionScript) oraz sekcja [Komunikacja z innymi instancjami Flash Player i AIR](#) (dla programistów HTML).

Uruchamianie zainstalowanej aplikacji AIR z przeglądarki

W celu korzystania z trybu wywołania z przeglądarki (która umożliwia uruchomienie aplikacji z przeglądarki) plik deskryptora aplikacji docelowej musi zawierać następujące ustawienia:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Więcej informacji na temat pliku deskryptora aplikacji zawiera rozdział „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Plik SWF w przeglądarce może uruchamiać aplikację AIR poprzez wywołanie metody `launchApplication()` w pliku `air.swf` załadowanym ze strony <http://airdownload.adobe.com/air/browserapi/air.swf>. Szczegółowe informacje zawiera sekcja „[Ładowanie pliku air.swf](#)” na stronie 268.

Po załadowaniu pliku `air.swf` plik SWF może wywołać metodę `launchApplication()` pliku `air.swf`, jak w poniższym kodzie:

```
var appID:String = "com.example.air.myTestApplication";  
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";  
var arguments:Array = ["launchFromBrowser"]; // Optional  
airSWF.launchApplication(appID, pubID, arguments);
```

Metoda `launchApplication()` jest zdefiniowana na najwyższym poziomie pliku `air.swf` (który jest załadowany do domeny aplikacji pliku SWF interfejsu użytkownika). Wywołanie tej metody sprawia, że środowisko AIR uruchamia określoną aplikację (jeśli jest zainstalowana i możliwe jest wywołanie przeglądarki, co definiuje ustawienie `allowBrowserInvocation` w pliku deskryptora aplikacji). Ta metoda ma następujące parametry:

Parametr	Opis
<code>appID</code>	Identyfikator dla aplikacji przeznaczonej do uruchomienia. Szczegółowe informacje zawiera opis właściwości „ <code>id</code> ” na stronie 239.
<code>pubID</code>	Identyfikator wydawcy dla aplikacji przeznaczonej do uruchomienia. Szczegółowe informacje zawiera opis właściwości „ <code>publisherID</code> ” na stronie 249. Jeśli konkretna aplikacja nie zawiera identyfikatora wydawcy, ustaw parametr <code>pubID</code> na pusty ciąg znaków (“”).
<code>arguments</code>	Tablica argumentów, które mają zostać przekazane do aplikacji. Obiekt <code>NativeApplication</code> aplikacji wywołuje zdarzenie <code>BrowserInvokeEvent</code> , którego właściwość <code>arguments</code> jest ustawiona na tę tablicę. W argumentach są rozpoznawane tylko znaki alfanumeryczne. Jeśli zachodzi potrzeba przekazania innych wartości, należy rozważyć użycie odpowiedniego systemu kodowania.

Metoda `launchApplication()` może działać wyłącznie wówczas, gdy zostanie wywołana w module obsługi zdarzeń, np. w przypadku zdarzenia myszy.

Metoda `launchApplication()` generuje błąd, jeśli w przeglądarce nie jest zainstalowana wymagana wersja programu Flash Player (wersja 9 z aktualizacją 3 lub nowszą w systemach Windows i Mac OS albo wersja 10 w systemie Linux).

Jeśli dla elementu `allowBrowserInvocation` ustawiono wartość `false` w pliku deskryptora aplikacji, wywołanie metody `launchApplication()` nie przynosi żadnego skutku.

Przed wyświetleniem interfejsu użytkownika do uruchomienia aplikacji konieczne może być wywołanie metody `getApplicationVersion()` w pliku `air.swf`. Szczegółowe informacje zawiera sekcja „[Sprawdzanie ze strony internetowej, czy aplikacja AIR została zainstalowana](#)” na stronie 269.

Gdy aplikacja zostaje wywołana za pośrednictwem funkcji wywołania przeglądarki, wówczas obiekt `NativeApplication` aplikacji wywołuje obiekt `BrowserInvokeEvent`. Szczegółowe informacje zawiera sekcja [Wywoływanie aplikacji AIR z przeglądarki](#) (dla programistów ActionScript) oraz sekcja [Wywoływanie aplikacji AIR z przeglądarki](#) (dla programistów HTML).

Jeśli korzysta się z funkcji wywoływania z przeglądarki, należy uwzględnić kwestie zabezpieczeń. Te kwestie zostały opisane w sekcji [Wywoływanie aplikacji AIR z przeglądarki](#) (dla programistów ActionScript) oraz w sekcji [Wywoływanie aplikacji AIR z przeglądarki](#) (dla programistów HTML).

Uruchomiona aplikacja może się komunikować z treścią pliku SWF w przeglądarce — w tym celu może korzystać z klasy `LocalConnection`. Więcej informacji zawiera sekcja [Komunikacja z innymi instancjami Flash Player i AIR](#) (dla programistów ActionScript) oraz sekcja [Komunikacja z innymi instancjami Flash Player i AIR](#) (dla programistów HTML).

Uwaga: Poczawszy od wersji AIR 1.5.3 identyfikator wydawcy nie jest używany. Identyfikatory wydawców nie są przypisywane do aplikacji automatycznie. W celu zapewnienia zgodności wstecz aplikacje nadal mogą określać identyfikator wydawcy.

Rozdział 17: Aktualizowanie aplikacji AIR

Użytkownicy mogą instalować lub aktualizować aplikację AIR, klikając dwukrotnie plik AIR na komputerze lub używając przeglądarki (funkcji instalacji bezproblemowej). Instalator środowiska Adobe® AIR® zarządza procesem instalacji i ostrzega użytkownika, jeśli ma nastąpić aktualizacja już zainstalowanej aplikacji.

Można jednak również spowodować, że zainstalowana aplikacja będzie samoczynnie aktualizować się do nowej wersji. W mechanizmie tym korzysta się z klasy Updater. (Zainstalowana aplikacja może wykryć dostępność nowej wersji do pobrania i zainstalowania). Klasa Updater zawiera metodę `update()`, która umożliwia wskazanie pliku AIR na komputerze użytkownika i zaktualizowanie aplikacji do wersji zawartej w pliku. Aby można było korzystać z klasy Updater, aplikacja musi być umieszczona w pakiecie będącym plikiem AIR. Aplikacje umieszczone w natywnych pakietach wykonywalnych lub w innych pakietach natywnych powinny używać funkcji aktualizowania udostępnianych przez platformę natywną.

Zarówno identyfikator aplikacji, jak i identyfikator wydawcy pliku z aktualizacją muszą być identyczne z odpowiednimi identyfikatorami aktualizowanej aplikacji. Identyfikator wydawcy jest wyznaczany na podstawie certyfikatu podpisującego, co oznacza, że zarówno aktualizacja, jak i aktualizowana aplikacja muszą być podpisane tym samym certyfikatem.

W środowisku AIR 1.5.3 i w późniejszych wersjach plik deskryptora aplikacji zawiera element `<publisherID>`. Ten element musi być używany, jeśli wersja aplikacji została utworzona w środowisku AIR 1.5.2 lub wcześniejszej wersji. Więcej informacji zawiera sekcja „[publisherID](#)” na stronie 249.

W wersji AIR 1.1 i nowszych istnieje możliwość przeprowadzenia migracji aplikacji w taki sposób, aby używany był dla niej nowy certyfikat do podpisywania kodu. Migracja aplikacji do nowego certyfikatu wymaga podpisania pliku AIR aktualizacji zarówno nowym, jak i oryginalnym certyfikatem. Migracja do nowego certyfikatu jest procesem jednokierunkowym. Po migracji wyłącznie pliki AIR podpisane nowym certyfikatem będą rozpoznawane jako aktualizacje istniejącej instalacji.

Zarządzanie aktualizowaniem aplikacji może być skomplikowanym zagadnieniem. Środowisko AIR 1.5 zawiera nową *architekturę aktualizacji aplikacji Adobe AIR*. Architektura ta obejmuje elementy interfejsu API pomagające programistom w tworzeniu sprawnych mechanizmów aktualizacji aplikacji AIR.

Migracja certyfikatu umożliwia zmianę certyfikatu samopodpisanego na komercyjny certyfikat podpisujący kod, a także zmianę jednego certyfikatu samopodpisanego lub certyfikatu komercyjnego na inny. Jeśli migracja do nowego certyfikatu nie zostanie przeprowadzona, użytkownicy muszą usunąć swoje bieżące wersje aplikacji przed zainstalowaniem nowej wersji. Więcej informacji zawiera sekcja „[Zmiana certyfikatów](#)” na stronie 206.

Dobłą metodą jest dołączenie mechanizmu aktualizowania do aplikacji. Jeśli zostanie utworzona nowa wersja aplikacji, mechanizm powiadomi użytkownika o możliwości zainstalowania nowej wersji.

Instalator aplikacji AIR tworzy pliki dziennika podczas instalowania, aktualizowania oraz usuwania aplikacji AIR. Dzięki tym dziennikom można określić przyczyny problemów z instalowaniem. Więcej informacji zawiera artykuł [Dzienniki instalacji](#).

Uwaga: Nowe wersje środowiska wykonawczego Adobe AIR mogą zawierać zaktualizowane wersje pakietu WebKit. Zaktualizowana wersja pakietu WebKit może powodować nieoczekiwane zmiany treści HTML we wdrożonej aplikacji AIR. Te zmiany mogą powodować konieczność aktualizowania aplikacji. Mechanizm aktualizacji może informować użytkownika o nowej wersji aplikacji. Więcej informacji zawiera sekcja [Informacje o środowisku HTML \(dla programistów ActionScript\)](#) oraz sekcja [Informacje o środowisku HTML \(dla programistów HTML\)](#).

Informacje o aktualizowaniu aplikacji

Klasa [Updater](#) (w pakiecie `flash.desktop`) zawiera jedną metodę, `update()`, której można użyć w celu zaktualizowania obecnie uruchomionej aplikacji do innej wersji. Na przykład, jeśli użytkownik ma wersję pliku AIR ("Sample_App_v2.air") umieszczoną na pulpicie, poniższy kod zaktualizuje aplikację.

Przykład w języku `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);
```

Przykład w języku `JavaScript`:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Przed użyciem klasy `Updater` użytkownik lub aplikacja musi pobrać zaktualizowaną wersję pliku AIR do komputera. Więcej informacji zawiera rozdział „[Pobieranie pliku AIR na komputer użytkownika](#)” na stronie 276.

Wyniki wywołania metody `Updater.update()`

Gdy aplikacja w środowisku wykonawczym wywoła metodę `update()`, środowisko wykonawcze zamyka aplikację, a następnie próbuje zainstalować nową wersję z pliku AIR. Środowisko wykonawcze sprawdza, czy identyfikatory aplikacji i wydawcy określone w pliku AIR zgadzają się z identyfikatorami aplikacji i wydawcy aplikacji wywołującej metodę `update()`. (Informacje na temat identyfikatorów aplikacji i wydawcy zawiera rozdział „[Pliki deskryptora aplikacji AIR](#)” na stronie 217). Środowisko sprawdza także, czy ciąg znaków wersji jest identyczny z ciągiem znaków `version` przekazanym do metody `update()`. Jeśli instalacja zakończy się pomyślnie, środowisko wykonawcze otwiera nową wersję aplikacji. W przeciwnym wypadku (jeśli nie było możliwe zakończenie instalacji), ponownie otwierana jest dotychczasowa (sprzed instalacji) wersja aplikacji.

W celu zainstalowania zaktualizowanej wersji aplikacji w systemie `Mac OS` użytkownik musi mieć odpowiednie uprawnienia dostępu do zainstalowania w katalogu aplikacji. W systemach `Windows` i `Linux` użytkownik musi mieć uprawnienia administracyjne.

Jeśli zaktualizowana wersja aplikacji wymaga zaktualizowanej wersji środowiska wykonawczego, instalowana jest nowa wersja tego środowiska. W celu zaktualizowania środowiska wykonawczego użytkownik musi posiadać uprawnienia administracyjne do komputera.

Podczas testowania aplikacji przy użyciu środowiska `ADL` wywołanie metody `update()` powoduje wygenerowanie wyjątku w czasie wykonywania.

Informacje o ciągu znaków wersji

Ciąg znaków określony jako parametr `version` metody `update()` musi być zgodny z ciągiem znaków w elemencie `version` lub `versionNumber` w pliku deskryptora aplikacji dla pliku AIR przeznaczonego do zainstalowania. Określenie parametru `version` jest wymagane ze względów bezpieczeństwa. Sprawdzanie przez aplikację numeru wersji w pliku AIR zapobiega niezamierzonemu zainstalowaniu starszej wersji. (Starsza wersja aplikacji może zawierać lukę w zabezpieczeniach, która została wyeliminowana w obecnie zainstalowanej wersji.) Aplikacja powinna także porównać ciąg znaków wersji w pliku AIR z ciągiem znaków wersji w instalowanej aplikacji, aby uniemożliwić ataki polegające na podstawieniu starszej wersji.

W wersjach środowiska AIR starszych niż 2.5 ciąg znaków wersji może mieć dowolny format. Na przykład może mieć wartość „2.01” lub „version 2”. W środowisku AIR 2.5 lub nowszym ciąg znaków wersji musi być sekwencją maksymalnie trzech trzycyfrowych liczb rozdzielonych kropkami. Prawidłowy numer wersji może mieć na przykład postać „0”, „1.0” lub „67.89.999”. Przed zaktualizowaniem aplikacji należy sprawdzić poprawność ciągu znaków wersji aktualizacji.

Jeśli aplikacja Adobe AIR pobierze plik AIR z sieci Web, wskazane jest zapewnienie mechanizmu powiadamiania przez usługę Web Service aplikacji Adobe AIR o pobieranej wersji. Aplikacja może następnie wykorzystać odpowiedni ciąg znaków jako parametr `version` metody `update()`. Jeśli plik AIR został uzyskany w inny sposób, a jego wersja jest nieznaną, aplikacja AIR może przeanalizować plik AIR w celu określenia informacji o wersji. (Plik AIR ma postać archiwum w skompresowanego w formacie ZIP, a plik deskryptora jest drugim rekordem w archiwum).

Szczegółowe informacje na temat pliku deskryptora aplikacji zawiera rozdział „[Pliki deskryptora aplikacji AIR](#)” na stronie 217.

Obieg pracy podpisywania aktualizacji aplikacji

Publikowanie aktualizacji metodą „ad hoc” komplikuje zarządzanie wieloma wersjami aplikacji, a także utrudnia śledzenie dat utraty ważności certyfikatów. Certyfikaty mogą utracić ważność, zanim będzie możliwe opublikowanie aktualizacji.

Środowisko wykonawcze Adobe AIR traktuje aktualizację aplikacji opublikowaną bez podpisu migracji jako nową aplikację. Użytkownicy muszą odinstalować bieżącą aplikację AIR, aby móc zainstalować aktualizację aplikacji.

W celu rozwiązania tego problemu należy wysłać każdą zaktualizowaną aplikację z najnowszym certyfikatem na osobny adres URL wdrażania. Należy uwzględnić mechanizm przypominania o stosowaniu podpisów migracji, gdy rozpocznie się 180-dniowy okres przedterminu certyfikatu. Więcej informacji zawiera rozdział „[Podpisywanie zaktualizowanej wersji aplikacji AIR](#)” na stronie 211.

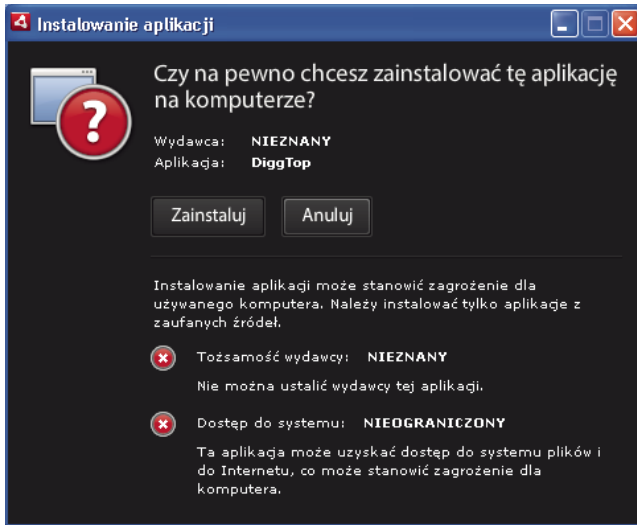
Informacje na temat stosowania podpisów zawiera sekcja „[Polecenia narzędzia ADT](#)” na stronie 175.

Aby usprawnić stosowanie podpisów migracji, należy wykonać następujące zadania:

- Wyślij każdą zaktualizowaną aplikację na osobny adres URL wdrażania.
- Wyślij plik XML deskryptora uaktualnienia i najnowszy certyfikat dla aktualizacji na ten sam adres URL.
- Podpisz zaktualizowaną aplikację za pomocą najnowszego certyfikatu.
- Zastosuj podpis migracji do zaktualizowanej aplikacji za pomocą certyfikatu użytego do podpisania poprzedniej wersji, umieszczonej pod innym adresem URL.

Prezentowanie niestandardowego interfejsu aktualizacji aplikacji

Środowisko AIR zawiera domyślny interfejs aktualizacji:



Ten interfejs jest używany zawsze, gdy użytkownik po raz pierwszy instaluje aplikację na komputerze. Można jednak zdefiniować własny interfejs, który będzie używany przy instalowaniu następnych wersji aplikacji. Jeśli aplikacja definiuje niestandardowy interfejs aktualizacji, należy dla obecnie zainstalowanej aplikacji określić element `customUpdateUI` w pliku deskryptora aplikacji:

```
<customUpdateUI>true</customUpdateUI>
```

Gdy aplikacja jest już zainstalowana, a użytkownik otworzy plik AIR z identyfikatorem aplikacji i wydawcy identycznymi z odpowiednimi identyfikatorami zainstalowanej aplikacji, środowisko wykonawcze AIR otwiera aplikację, a nie domyślny instalator aplikacji AIR. Więcej informacji zawiera sekcja „`customUpdateUI`” na stronie 229.

Aplikacja w chwili uruchomienia (tj. gdy obiekt `NativeApplication.nativeApplication` wywoła zdarzenie `load`) może zdecydować, czy ma dokonać aktualizacji (przy użyciu klasy `Updater`). Jeśli aktualizacja ma być dokonana, aplikacja może zaprezentować użytkownikowi własny interfejs instalacji (który będzie się różnił od standardowego interfejsu do obsługi aplikacji).

Pobieranie pliku AIR na komputer użytkownika

Aby użyć klasy `Updater`, użytkownik lub aplikacja musi uprzednio zapisać plik AIR lokalnie na dysku swojego komputera.

Uwaga: W skład środowiska AIR 1.5 wchodzi architektura aktualizacji, która pomaga programistom w realizacji sprawnych mechanizmów aktualizacji aplikacji AIR. Korzystanie z tej architektury może okazać się znacznie łatwiejsze niż bezpośrednie używanie metody `update()` klasy `Update`. Szczegółowe informacje zawiera sekcja „[Korzystanie z architektury aktualizacji](#)” na stronie 280.

Poniższy kod odczytuje plik AIR z adresu URL (http://example.com/air/updates/Sample_App_v2.air) i zapisuje ten plik w katalogu magazynu aplikacji.

Przykład w języku 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.");
}
```

Przykład w języku 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.");
}
```

Więcej informacji:

- [Przepływ pracy w celu odczytywania i zapisywania plików](#) (dla programistów ActionScript)
- [Przepływ pracy w celu odczytywania i zapisywania plików](#) (dla programistów HTML)

Sprawdzanie, czy aplikacja została uruchomiona po raz pierwszy

Często po zaktualizowaniu aplikacja wyświetla komunikat powitalny lub instrukcję „pierwsze kroki”. Po uruchomieniu aplikacja sprawdza, czy została uruchomiona po raz pierwszy, i na tej podstawie może wyświetlić taki komunikat lub nie.

Uwaga: W skład środowiska AIR 1.5 wchodzi architektura aktualizacji, która pomaga programistom w realizacji sprawnych mechanizmów aktualizacji aplikacji AIR. Architektura ta udostępnia łatwe metody sprawdzania, czy dana aplikacja jest uruchomiona po raz pierwszy. Szczegółowe informacje zawiera sekcja „[Korzystanie z architektury aktualizacji](#)” na stronie 280.

Jedną z metod sprawdzenia tego warunku polega na zapisaniu pliku w katalogu magazynu aplikacji po jej zainicjowaniu. Po każdym uruchomieniu aplikacja powinna sprawdzać, czy plik istnieje. Brak pliku oznacza, że aplikacja została uruchomiona po raz pierwszy dla bieżącego użytkownika. Obecność pliku oznacza, że aplikacja była już co najmniej jeden raz uruchamiana. Jeśli plik istnieje i zawiera numer wersji starszej niż bieżąca, można przyjąć, że użytkownik uruchamia nową wersję po raz pierwszy.

Poniższy przykład Flex ilustruje ten sposób postępowania.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    title="Sample Version Checker Application"
    applicationComplete="system extension()">
  <mx:Script>
    <![CDATA [
      import flash.filesystem.*;
      public var file:File;
      public var currentVersion:String = "1.2";
      public function system extension():void {
        file = File.applicationStorageDirectory;
        file = file.resolvePath("Preferences/version.txt");
        trace(file.nativePath);
        if(file.exists) {
          checkVersion();
        } else {
          firstRun();
        }
      }
      private function checkVersion():void {
        var stream:FileStream = new FileStream();
        stream.open(file, FileMode.READ);
        var reversion:String = stream.readUTFBytes(stream.bytesAvailable);
        stream.close();
        if (reversion != currentVersion) {
          log.text = "You have updated to version " + currentVersion + ".\n";
        }
      }
    ]]>
  </mx:Script>
</mx:WindowedApplication>
```

```
        } else {
            saveFile();
        }
        log.text += "Welcome to the application.";
    }
    private function firstRun():void {
        log.text = "Thank you for installing the application. \n"
            + "This is the first time you have run it.";
        saveFile();
    }
    private function saveFile():void {
        var stream:FileStream = new FileStream();
        stream.open(file, FileMode.WRITE);
        stream.writeUTFBytes(currentVersion);
        stream.close();
    }
    ]]>
</mx:Script>
<mx:TextArea ID="log" width="100%" height="100%" />
</mx:WindowedApplication>
```

Poniższy przykład ilustruje to rozwiązanie w języku 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.";
      }
    </script>
  </head>
  <body>
    <div id="log" style="border: 1px solid black; padding: 5px; min-height: 100px;">
```

```
    }  
    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>
```

Jeśli aplikacja zapisuje dane lokalnie (np. w katalogu magazynu aplikacji), niekiedy celowe jest sprawdzenie przy pierwszym uruchomieniu, czy istnieją dane zapisane wcześniej przez poprzednie wersje.

Korzystanie z architektury aktualizacji

Zarządzanie aktualizacjami aplikacji może być pracochłonne. *Infrastruktura aktualizacji dla aplikacji Adobe AIR* oferuje interfejsy API, które umożliwiają programistom udostępnienie niezawodnych funkcji aktualizowania w aplikacjach AIR. Infrastruktura aktualizacji środowiska AIR zwalnia programistów z wykonywania następujących zadań:

- Okresowe sprawdzanie dostępności aktualizacji w odpowiednich interwałach lub na żądanie użytkownika
- Pobieranie plików AIR (aktualizacji) ze źródła w Internecie
- Powiadomianie użytkownika przy pierwszym uruchomieniu nowo zainstalowanej wersji
- Potwierdzanie, że użytkownik chce sprawdzić dostępność aktualizacji
- Wyświetlanie informacji o wersji nowej aktualizacji
- Wyświetlanie informacji o postępie pobierania i błędach

Infrastruktura aktualizacji środowiska AIR zawiera przykładowy interfejs użytkownika dla aplikacji. Interfejs ten udostępnia użytkownikowi podstawowe informacje i opcje konfiguracji związane z aktualizacjami aplikacji. Aplikacja może także definiować własny interfejs użytkownika współpracujący z infrastrukturą aktualizacji.

Infrastruktura aktualizacji aplikacji AIR umożliwia przechowywanie informacji o wersji aktualizacji aplikacji AIR w prostych plikach konfiguracyjnych XML. W przypadku większości aplikacji przygotowanie tych plików konfiguracyjnych przez dodanie podstawowego kodu wystarczy do zapewnienia użytkownikom końcowym odpowiedniej obsługi aktualizacji.

Nawet jeśli programista nie zdecyduje się na użycie infrastruktury aktualizacji, aplikacje mogą korzystać z klasy Updater dostępnej w środowisku Adobe AIR, która wspomaga uaktualnianie do nowych wersji. Klasa Updater umożliwia aplikacji uaktualnienie do wersji zawartej w pliku AIR na komputerze użytkownika. Jednak zarządzanie aktualizacjami zwykle nie sprowadza się do prostej aktualizacji aplikacji na podstawie lokalnego pliku AIR.

Pliki architektury aktualizacji AIR

Architektura aktualizacji AIR jest zawarta w katalogu `frameworks/libs/air` pakietu AIR 2 SDK. Zawiera następujące pliki:

- `applicationupdater.swc` — definiuje podstawowe funkcje biblioteki aktualizacji, do wykorzystania w kodzie ActionScript. Ta wersja nie zawiera interfejsu użytkownika.
- `applicationupdater.swf` — definiuje podstawowe funkcje biblioteki aktualizacji, do wykorzystania w kodzie JavaScript. Ta wersja nie zawiera interfejsu użytkownika.
- `applicationupdater_ui.swc` — definiuje podstawową funkcjonalność biblioteki aktualizacji dla Flex 4 wraz z interfejsem użytkownika, którego aplikacja może używać do wyświetlania opcji aktualizacji.
- `applicationupdater_ui.swf` — definiuje podstawową funkcjonalność biblioteki aktualizacji dla JavaScript wraz z interfejsem użytkownika, którego aplikacja może używać do wyświetlania opcji aktualizacji.

Więcej informacji zawierają poniższe sekcje:

- „[Konfigurowanie środowiska programistycznego Flex](#)” na stronie 281
- „[Uwzględnianie plików architektury w aplikacji AIR opartej na kodzie HTML](#)” na stronie 281
- „[Prosty przykład: korzystanie z wersji ApplicationUpdaterUI](#)” na stronie 282

Konfigurowanie środowiska programistycznego Flex

Pliki SWC w katalogu `frameworks/libs/air` pakietu AIR 2 SDK definiują klasy używane podczas programowania w środowisku Flex i w programie Flash.

Aby skorzystać z architektury aktualizacji przy kompilowaniu aplikacji za pomocą pakietu Flex SDK, należy uwzględnić plik `ApplicationUpdater.swc` albo `ApplicationUpdater_UI.swc` w wywołaniu kompilatora `amxmlc`. W poniższym przykładzie kompilator ładuje plik `ApplicationUpdater.swc` z podkatalogu `lib` w katalogu pakietu Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater.swc -- myApp.mxml
```

W poniższym przykładzie kompilator ładuje plik `ApplicationUpdater_UI.swc` z podkatalogu `lib` pakietu Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc -- myApp.mxml
```

W przypadku tworzenia aplikacji za pomocą kompilatora Flash Builder należy dodać plik SWC na karcie ścieżki biblioteki w ustawieniach ścieżki kompilacji Flex w oknie dialogowym właściwości.

Pliki SWC należy koniecznie skopiować do katalogu, do którego będziemy odwoływać się w kompilatorze `amxmlc` (w przypadku użycia pakietu Flex SDK) lub w programie Flash Builder.

Uwzględnianie plików architektury w aplikacji AIR opartej na kodzie HTML

Katalog `frameworks/html` architektury aktualizacji zawiera następujące pliki:

- `applicationupdater.swf` — definiuje podstawowe funkcje biblioteki aktualizacji, bez interfejsu użytkownika.
- `applicationupdater_ui.swf` — definiuje podstawowe funkcje biblioteki aktualizacji wraz z interfejsem użytkownika, którego aplikacja może używać do wyświetlania opcji aktualizacji.

Kod JavaScript w aplikacjach AIR może korzystać z klas zdefiniowanych w plikach SWF.

Aby skorzystać z infrastruktury aktualizacji, należy umieścić plik `applicationupdater.swf` albo `applicationupdater_ui.swf` w katalogu (lub podkatalogu) aplikacji. Następnie w pliku HTML, w którym infrastruktura będzie stosowana (w kodzie JavaScript) należy umieścić znacznik `script` wczytujący ten plik.

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

Można też użyć poniższego znacznika `script` w celu wczytania pliku `applicationupdater_ui.swf`.

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

Interfejs API zdefiniowany w tych dwóch plikach został opisany w dalszej części niniejszego dokumentu.

Prosty przykład: korzystanie z wersji `ApplicationUpdaterUI`

Wersja architektury aktualizacji zapisana w pliku `ApplicationUpdaterUI` udostępnia prosty interfejs, który łatwo można włączyć do własnych aplikacji. Poniżej przedstawiono prosty przykład.

Najpierw tworzymy aplikację AIR, która wywołuje architekturę aktualizacji:

- 1 Jeśli aplikacja AIR jest oparta na języku HTML, należy wczytać plik `applicationupdater_ui.swf`.

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

- 2 W logice programowej aplikacji AIR tworzymy instancję obiektu `ApplicationUpdaterUI`.

W języku `ActionScript` należałoby użyć następującego kodu:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

W języku `JavaScript` należałoby użyć następującego kodu:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Celowe może być dodanie tego kodu do funkcji inicjującej, wykonywanej po załadowaniu aplikacji.

- 3 Tworzymy plik tekstowy o nazwie `updateConfig.xml` i dodajemy do niego następujący kod:

```
<?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>
```

Modyfikujemy element `URL` pliku `updateConfig.xml` w taki sposób, aby wskazywał docelową lokalizację pliku deskryptora aktualizacji na serwerze WWW (patrz następna procedura).

Wartość `delay` to liczba dni, co jaką aplikacja sprawdza dostępność aktualizacji.

- 4 Dodajemy plik `updateConfig.xml` do katalogu projektu aplikacji AIR.
- 5 Powodujemy, że obiekt `Updater` odwołuje się do pliku `updateConfig.xml`, i wywołujemy metodę `initialize()` obiektu.

W języku `ActionScript` należałoby użyć następującego kodu:

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

W języku `JavaScript` należałoby użyć następującego kodu:

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

- 6 Tworzymy drugą wersję aplikacji AIR, o innym numerze niż pierwsza aplikacja. (Wersja jest podana w pliku deskryptora, w elemencie `version`).

Następnie dodajemy zaktualizowaną wersję aplikacji AIR na serwer WWW:

- 1 Umieszczamy pliku AIR wersji zaktualizowanej na serwerze WWW.
- 2 Należy utworzyć plik tekstowy o nazwie `updateDescriptor.2.5.xml` i dodać do niego następującą treść:

```
<?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>
```

Należy edytować elementy `versionNumber`, URL i `description` pliku `updateDescriptor.xml` w taki sposób, aby odpowiadały plikowi AIR aktualizacji. Ten format deskryptora aktualizacji jest używany przez aplikacje korzystające z infrastruktury aktualizacji dołączonej do zestawu SDK środowiska AIR 2.5 i nowszych wersji.

3 Należy utworzyć plik tekstowy o nazwie `updateDescriptor.1.0.xml` i dodać do niego następującą treść:

```
<?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>
```

Należy edytować elementy `version`, URL i `description` pliku `updateDescriptor.xml` w taki sposób, aby wskazywały na plik AIR aktualizacji. Ten format deskryptora aktualizacji jest używany przez aplikacje korzystające z infrastruktury aktualizacji dołączonej do zestawu SDK środowiska AIR 2 i starszych wersji.

Uwaga: *Utworzenie drugiego z podanych plików deskryptora aktualizacji jest konieczne tylko w przypadku obsługi aktualizacji aplikacji opracowanych dla środowiska AIR w wersji starszej niż 2.5.*

4 Należy dodać pliki `updateDescriptor.2.5.xml` i `updateDescriptor.1.0.xml` do tego samego katalogu serwera internetowego, który zawiera plik AIR aktualizacji.

Choć jest to prosty przykład, zapewnia on funkcjonalność aktualizacji wystarczającą w wielu zastosowaniach. W pozostałej części tego dokumentu opisano sposoby wykorzystania architektury aktualizacji odpowiednio do indywidualnych potrzeb.

Dodatkowe przykłady użycia architektury aktualizacji zawiera następująca aplikacja w serwisie Centrum programowania Adobe AIR:

- [Architektura aktualizacji w aplikacji opartej na środowisku Flash](http://www.adobe.com/go/learn_air_qs_update_framework_flash_pl)
(http://www.adobe.com/go/learn_air_qs_update_framework_flash_pl)

Aktualizowanie do środowiska AIR 2.5

Zasady przydzielania numerów wersji do aplikacji zostały zmienione w środowisku AIR 2.5, dlatego infrastruktura aktualizacji środowiska AIR 2 nie może przeanalizować informacji o wersji w deskrytorze aplikacji środowiska AIR 2.5. Ta niezgodność oznacza, że należy zaktualizować aplikację w celu korzystania z nowej infrastruktury aktualizacji, ZANIM aplikacja zostanie zaktualizowana w celu korzystania z zestawu SDK środowiska AIR 2.5. Dlatego aktualizowanie aplikacji z dowolnej wersji środowiska AIR starszej niż 2.5 w celu korzystania ze środowiska AIR 2.5 lub nowszej wersji wymaga DWÓCH aktualizacji. Pierwsza aktualizacja musi używać przestrzeni nazw środowiska AIR 2 i uwzględniać bibliotekę infrastruktury aktualizacji środowiska AIR 2.5. (Pakiet aplikacji można utworzyć za pomocą zestawu SDK środowiska AIR 2.5). Druga aktualizacja może używać przestrzeni nazw środowiska AIR 2.5 i uwzględniać nowe funkcje aplikacji.

Pośrednia aktualizacja może również nie wykonywać żadnych operacji oprócz zaktualizowania aplikacji do wersji dla środowiska AIR 2.5 przy użyciu klasy `Updater` środowiska AIR w bezpośredni sposób.

Poniższy przykład przedstawia aktualizowanie aplikacji z wersji 1.0 do 2.0. Wersja 1.0 używa starej przestrzeni nazw wersji 2.0. Wersja 2.0 używa przestrzeni nazw wersji 2.5 i zawiera nowe funkcje zaimplementowane przy użyciu interfejsów API środowiska 2.5.

1 Utwórz pośrednią wersję aplikacji (1.0.1) na podstawie wersji 1.0.

a Podczas tworzenia aplikacji użyj infrastruktury aktualizacji aplikacji środowiska AIR 2.5.

Uwaga: W przypadku aplikacji AIR opartej na technologii Flash użyj pliku `applicationupdater.swc` lub `applicationupdater_ui.swc`. W przypadku aplikacji AIR opartej na języku HTML użyj pliku `applicationupdater.swf` lub `applicationupdater_ui.swf`.

b Utwórz plik deskryptora aktualizacji dla wersji 1.0.1, używając starej przestrzeni nazw i wersji, tak jak pokazano poniżej.

```
<?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 Utwórz wersję 2.0 aplikacji, korzystając z interfejsów API i przestrzeni nazw środowiska AIR 2.5.

3 Utwórz deskryptor aktualizacji w celu zaktualizowania aplikacji z wersji 1.0.1 do wersji 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>
```

Definiowanie plików deskryptora aktualizacji i dodawanie pliku AIR do serwera internetowego

Podczas pracy z infrastrukturą aktualizacji środowiska AIR należy zdefiniować podstawowe informacje o dostępnej aktualizacji w plikach deskryptora aktualizacji przechowywanych na serwerze internetowym. Plik deskryptora aktualizacji jest prostym plikiem XML. Infrastruktura aktualizacji dołączona do aplikacji sprawdza zawartość tego pliku, aby stwierdzić, czy na serwerze została umieszczona nowa wersja.

Format pliku deskryptora aktualizacji uległ zmianie w środowisku AIR 2.5. W nowym formacie jest używana inna przestrzeń nazw. Oryginalna przestrzeń nazw to „`http://ns.adobe.com/air/framework/update/description/1.0`”. Przestrzeń nazw środowiska AIR 2.5 to „`http://ns.adobe.com/air/framework/update/description/2.5`”.

Aplikacje AIR utworzone dla wersji środowiska AIR starszych niż 2.5 mogą odczytywać tylko deskryptor aktualizacji dotyczący wersji 1.0. Aplikacje AIR utworzone za pomocą infrastruktury aktualizacji zawartej w środowisku AIR 2.5 lub nowszym mogą odczytywać tylko deskryptor aktualizacji dotyczący wersji 2.5. Z powodu tej niezgodności wersji często trzeba tworzyć dwa pliki deskryptora aktualizacji. Logika obsługująca aktualizowanie w wersjach aplikacji dla środowiska AIR 2.5 musi pobierać deskryptor aktualizacji korzystający z nowego formatu. Starsze wersje aplikacji AIR muszą nadal korzystać z oryginalnego formatu. Dla każdej wydanej aktualizacji należy zmodyfikować oba pliki (do czasu zakończenia obsługi wersji utworzonych dla wersji środowiska AIR starszych niż 2.5).

Plik deskryptora aktualizacji zawiera następujące dane:

- `versionNumber` — nowa wersja aplikacji AIR. Element `versionNumber` należy stosować w deskryptorach aktualizacji używanych do aktualizowania aplikacji AIR 2.5. Podana wartość musi być taka sama jak ciąg znaków użyty w elemencie `versionNumber` w nowym pliku deskryptora aplikacji AIR. Jeśli numer wersji w pliku deskryptora aktualizacji nie jest zgodny z numerem wersji pliku AIR aktualizacji, infrastruktura aktualizacji generuje wyjątek.
- `version` — nowa wersja aplikacji AIR. Element `version` należy stosować w deskryptorach aktualizacji używanych do aktualizowania aplikacji utworzonych dla wersji środowiska AIR starszych niż 2.5. Podana wartość musi być taka sama jak ciąg znaków użyty w elemencie `version` w nowym pliku deskryptora aplikacji AIR. Jeśli wersja w pliku deskryptora aktualizacji nie jest zgodna z wersją pliku AIR aktualizacji, infrastruktura aktualizacji generuje wyjątek.
- `versionLabel` — ciąg znaków w postaci czytelnej dla człowieka, przeznaczony do wyświetlania dla użytkowników. Element `versionLabel` jest opcjonalny, ale można go określać tylko w plikach deskryptora aktualizacji w wersji 2.5. Ten element należy stosować, jeśli w pliku deskryptora aplikacji jest używany element `versionLabel`. (Dla obu elementów należy ustawić tę samą wartość).
- `url` — lokalizacja pliku AIR aktualizacji. Jest to plik zawierający zaktualizowaną wersję aplikacji AIR.
- `description` — opis nowej wersji. Te informacje mogą być wyświetlane dla użytkownika podczas aktualizacji.

Elementy `version` i `url` są wymagane. Element `description` jest opcjonalny.

Oto przykładowy plik deskryptora aktualizacji dla wersji 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>
```

Oto przykładowy plik deskryptora aktualizacji dla wersji 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>
```

Aby zdefiniować znacznik `description` w wielu językach, należy zastosować wiele elementów `text` definiujących atrybut `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>
```

Umieszczamy plik deskryptora aktualizacji wraz z plikiem AIR aktualizacji na serwerze WWW.

Katalog templates dołączony do deskryptora aktualizacji zawiera przykładowe pliki deskryptorów aktualizacji. Zawierają one wersje w jednym języku oraz wersje wielojęzyczne.

Tworzenie instancji klasy Updater

Po załadowaniu architektury aktualizacji AIR w kodzie (patrz „[Konfigurowanie środowiska programistycznego Flex](#)” na stronie 281 i „[Uwzględnianie plików architektury w aplikacji AIR opartej na kodzie HTML](#)” na stronie 281) konieczne jest utworzenie instancji klasy Updater, co ilustruje poniższy przykład.

Przykład w języku ActionScript:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

Przykład w języku JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

W powyższym kodzie zastosowano klasę ApplicationUpdater (która nie udostępnia interfejsu użytkownika). Oto kod, w którym zastosowano klasę ApplicationUpdaterUI (z interfejsem użytkownika).

Przykład w języku ActionScript:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Przykład w języku JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

W przykładach kodu zamieszczonych w dalszej części tego dokumentu przyjęto założenie, że został utworzony obiekt Updater o nazwie appUpdater.

Konfigurowanie ustawień aktualizacji

Zarówno klasę ApplicationUpdater, jak i klasę ApplicationUpdaterUI można skonfigurować za pośrednictwem pliku konfiguracji dostarczonego z aplikacją lub za pomocą kodu ActionScript bądź JavaScript w aplikacji.

Definiowanie ustawień aktualizacji w pliku konfiguracyjnym XML

Plik konfiguracyjny aktualizacji jest plikiem XML. Może zawierać następujące elementy:

- `updateURL` — ciąg znaków (String). Reprezentuje lokalizację deskryptora aktualizacji na serwerze zdalnym. Dozwolone są wszelkie poprawne lokalizacje URLRequest. Właściwość `updateURL` musi być zdefiniowana — albo w pliku konfiguracyjnym, albo za pomocą skryptu (zobacz „[Definiowanie plików deskryptora aktualizacji i dodawanie pliku AIR do serwera internetowego](#)” na stronie 284). Właściwość tę należy zdefiniować przed użyciem obiektu Updater (przed wywołaniem metody `initialize()` obiektu Updater, co opisano w sekcji „[Inicjowanie architektury aktualizacji](#)” na stronie 289).
- `delay` — liczba (Number). Reprezentuje interwał (w dniach, dozwolone są wartości ułamkowe, takie jak 0.25) sprawdzania dostępności aktualizacji. Wartość 0 (domyślna) określa, że obiekt Updater nie będzie okresowo automatycznie sprawdzał dostępności aktualizacji.

Plik konfiguracyjny dla klasy `ApplicationUpdaterUI` może, oprócz elementów `updateURL` i `delay`, zawierać następujący element:

- `defaultUI`: lista elementów `dialog`. Każdy element `dialog` ma atrybut `name` odpowiadający oknu dialogowemu w interfejsie użytkownika. Każdy element `dialog` ma atrybut `visible` określający, czy okno dialogowe jest widoczne. Wartością domyślną jest `true`. Oto możliwe wartości atrybutu `name`:
 - `"checkForUpdate"` — odpowiada oknom dialogowym Sprawdź dostępność aktualizacji, Brak aktualizacji i Błąd aktualizacji.
 - `"downloadUpdate"` — odpowiada oknu dialogowemu Pobierz aktualizację.
 - `"downloadProgress"` — odpowiada oknom dialogowym Postęp pobierania i Błąd pobierania.
 - `"installUpdate"` — odpowiada oknu dialogowemu Zainstaluj aktualizację.
 - `"fileUpdate"` — odpowiada oknom dialogowym Aktualizacja pliku, Plik nie jest aktualizacją i Błąd pliku.
- `"unexpectedError"` — odpowiada oknu dialogowemu Nieoczekiwany błąd.

Ustawienie wartości `false` powoduje, że odpowiednie okna dialogowe nie będą pojawiać się w trakcie procedury aktualizacji.

Oto przykładowy plik konfiguracyjny dla architektury `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>
```

Oto przykładowy plik konfiguracyjny dla architektury `ApplicationUpdaterUI` zawierający definicję elementów `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>
```

Właściwość `configurationFile` powinna wskazywać na położenie tego pliku:

Przykład w języku ActionScript:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

Przykład w języku JavaScript:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

Katalog `templates` architektury aktualizacji zawiera przykładowy plik konfiguracyjny o nazwie `config-template.xml`.

Definiowanie ustawień aktualizacji za pomocą kodu ActionScript lub JavaScript

Te same parametry konfiguracyjne można ustawić za pośrednictwem kodu aplikacji, tak jak w poniższym przykładzie:

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

Obiekt `Updater` ma właściwości `updateURL` i `delay`. Właściwości te opisują te same ustawienia, co elementy `updateURL` i `delay` w pliku konfiguracyjnym: adres URL pliku deskryptora aktualizacji i interwał sprawdzania dostępności aktualizacji. W wypadku określenia ustawień w pliku konfiguracyjnym i w kodzie wszelkie właściwości ustawione w kodzie przesłaniają odpowiednie ustawienia w pliku.

Właściwość `updateURL` musi być zdefiniowana — albo w pliku konfiguracyjnym, albo za pomocą skryptu (zobacz „[Definiowanie plików deskryptora aktualizacji i dodawanie pliku AIR do serwera internetowego](#)” na stronie 284) — przed użyciem obiektu `Updater` (przed wywołaniem metody `initialize()` obiektu `Updater`, co opisano w rozdziale „[Inicjowanie architektury aktualizacji](#)” na stronie 289).

W architekturze `ApplicationUpdaterUI` zdefiniowane są następujące dodatkowe właściwości obiektu `Updater`:

- `isCheckForUpdateVisible` — odpowiada oknom dialogowym Sprawdź dostępność aktualizacji, Brak aktualizacji i Błąd aktualizacji.
- `isDownloadUpdateVisible` — odpowiada oknu dialogowemu Pobierz aktualizację.
- `isDownloadProgressVisible` — odpowiada oknom dialogowym Postęp pobierania i Błąd pobierania.
- `isInstallUpdateVisible` — odpowiada oknu dialogowemu Zainstaluj aktualizację.
- `isFileUpdateVisible` — odpowiada oknom dialogowym Aktualizacja pliku, Plik nie jest aktualizacją i Błąd pliku.
- `isUnexpectedErrorVisible` — odpowiada oknu dialogowemu Nieoczekiwany błąd.

Każda z właściwości odpowiada jednemu lub większej liczbie okien dialogowych w interfejsie użytkownika architektury `ApplicationUpdaterUI`. Wszystkie właściwości są typu `Boolean` i mają wartość domyślną `true`. Ustawienie wartości `false` powoduje, że odpowiednie okna dialogowe nie będą pojawiać się w trakcie procedury aktualizacji.

Właściwości okien dialogowych przesłaniają ustawienia w pliku konfiguracyjnym aktualizacji.

Proces aktualizacji

Architektura aktualizacji AIR realizuje proces aktualizacji w następujących krokach:

- 1 Procedura inicjowania obiektu `Updater` sprawdza, czy w zdefiniowanym okresie dokonano już sprawdzenia dostępności aktualizacji (patrz „[Konfigurowanie ustawień aktualizacji](#)” na stronie 286). Jeśli przypada termin aktualizacji, proces aktualizacji jest kontynuowany.
- 2 Obiekt `Updater` pobiera i interpretuje plik deskryptora aktualizacji.
- 3 Obiekt `Updater` pobiera plik AIR aktualizacji.
- 4 Obiekt `Updater` instaluje zaktualizowaną wersję aplikacji.

Obiekt `Updater` wywołuje zdarzenia po zakończeniu każdego z tych kroków. W architekturze `ApplicationUpdater` możliwe jest anulowanie zdarzeń wskazujących na pomyślne zakończenie kroków procesu. W wypadku anulowania jednego z takich zdarzeń anulowany zostanie następny krok procesu. W architekturze `ApplicationUpdaterUI` obiekt `Updater` wyświetla okno dialogowe umożliwiające użytkownikowi anulowanie lub wykonanie każdego z kroków procesu.

W wypadku anulowania zdarzenia możliwe jest wznowienie procesu poprzez wywołanie metod obiektu `Updater`.

W miarę, jak obiekt `Updater` w wersji `ApplicationUpdater` realizuje kolejne etapy aktualizacji, jego bieżący stan jest zapisywany we właściwości `currentState`. Tej właściwości przypisywany jest ciąg znaków o jednej z następujących możliwych wartości:

- "UNINITIALIZED" — obiekt `Updater` nie został zainicjowany.

- "INITIALIZING" — trwa inicjowanie obiektu Updater.
- "READY" — obiekt Updater został zainicjowany.
- "BEFORE_CHECKING" — obiekt Updater nie sprawdził jeszcze, czy istnieje plik deskryptora aktualizacji.
- "CHECKING" — obiekt Updater sprawdza, czy istnieje plik deskryptora aktualizacji.
- "AVAILABLE" — plik deskryptora jest dostępny dla obiektu Updater.
- "DOWNLOADING" — obiekt Updater pobiera plik AIR.
- "DOWNLOADED" — obiekt Updater pobrał plik AIR.
- "INSTALLING" — obiekt Updater instaluje plik AIR.
- "PENDING_INSTALLING" — obiekt Updater jest zainicjowany i istnieją oczekujące aktualizacje.

Niektóre metody obiektu Updater są wykonywane tylko wtedy, gdy obiekt ten jest w określonym stanie.

Inicjowanie architektury aktualizacji

Po ustawieniu właściwości konfiguracyjnych (zobacz „[Prosty przykład: korzystanie z wersji ApplicationUpdaterUI](#)” na stronie 282) należy wywołać metodę `initialize()` w celu zainicjowania aktualizacji:

```
appUpdater.initialize();
```

Metoda ta wykonuje następujące operacje:

- Inicjuje architekturę aktualizacji, przeprowadzając cichą instalację wszelkich oczekujących aktualizacji. Wymagane jest wywołanie tej metody podczas uruchamiania aplikacji, ponieważ może ona wymusić ponowne uruchomienie aplikacji.
- Sprawdza, czy istnieje oczekująca (odłożona w czasie) aktualizacja, a jeśli tak, instaluje ją.
- Jeśli w procesie aktualizacji wystąpi błąd, metoda usuwa informacje o pliku aktualizacji i o wersji z obszaru pamięci aplikacji.
- Jeśli minął okres opóźnienia, rozpoczyna proces aktualizacji. W przeciwnym razie ponownie uruchamia licznik czasu.

Wywołanie tej metody może spowodować, że obiekt Updater wywoła następujące zdarzenia:

- `UpdateEvent.INITIALIZED` — wywoływane po zakończeniu inicjowania.
- `ErrorEvent.ERROR` — wywoływane, jeśli w trakcie inicjowania wystąpi błąd.

Po wywołaniu zdarzenia `UpdateEvent.INITIALIZED` proces aktualizacji jest ukończony.

Wywołanie metody `initialize()` powoduje, że obiekt Updater rozpoczyna proces aktualizacji i wykonuje wszystkie kroki z uwzględnieniem ustawionego opóźnienia. Możliwe jest jednak także rozpoczęcie procesu aktualizacji w dowolnej chwili, poprzez wywołanie metody `checkNow()` obiektu Updater:

```
appUpdater.checkNow();
```

Jeśli proces aktualizacji już trwa, ta metoda nie wykonuje żadnych czynności. W przeciwnym razie rozpoczyna proces aktualizacji.

Obiekt Updater może wywoływać następujące zdarzenie w wyniku wywołania metody `checkNow()`:

- zdarzenie `UpdateEvent.CHECK_FOR_UPDATE` tuż przed próbą pobrania pliku deskryptora aktualizacji.

Jeśli zdarzenie `checkForUpdate` zostanie anulowane, można wywołać metodę `checkForUpdate()` obiektu Updater. (Patrz następna sekcja). Jeśli zdarzenie nie zostanie anulowane, proces aktualizacji przechodzi do etapu sprawdzania, czy istnieje plik deskryptora aktualizacji.

Zarządzanie procesem aktualizacji w wersji ApplicationUpdaterUI

W wersji ApplicationUpdaterUI użytkownik może anulować proces za pomocą przycisków Anuluj w oknach dialogowych interfejsu użytkownika. Możliwe jest także programowe anulowanie procesu aktualizacji poprzez wywołanie metody `cancelUpdate()` obiektu ApplicationUpdaterUI.

W celu określenia, które okna dialogowe potwierżeń mają być wyświetlane przez obiekt Updater, można ustawić właściwości obiektu ApplicationUpdaterUI lub zdefiniować elementy pliku konfiguracyjnego aplikacji. Szczegółowe informacje zawiera sekcja „[Konfigurowanie ustawień aktualizacji](#)” na stronie 286.

Zarządzanie procesem aktualizacji w wersji ApplicationUpdater

Możliwe jest wywoływanie metody `preventDefault()` obiektów zdarzeń wywoływanych przez obiekt ApplicationUpdater w celu anulowania odpowiednich kroków procesu aktualizacji (patrz „[Proces aktualizacji](#)” na stronie 288). Anulowanie domyślnego zachowania stwarza aplikacji możliwość wyświetlenia komunikatu z pytaniem, czy użytkownik chce kontynuować.

W dalszych sekcjach opisano sposób kontynuacji procesu aktualizacji w wypadku anulowania jednego z kroków procesu.

Pobieranie i interpretowanie pliku deskryptora aktualizacji

Obiekt ApplicationUpdater wywołuje zdarzenie `checkForUpdate` przed rozpoczęciem procesu aktualizacji, tuż przed próbą pobrania pliku deskryptora aktualizacji. Jeśli domyślne zachowanie zdarzenia `checkForUpdate` zostanie anulowane, obiekt Updater nie pobierze pliku deskryptora aktualizacji. Można wywołać metodę `checkForUpdate()` i tym samym wznowić proces aktualizacji:

```
appUpdater.checkForUpdate();
```

Wywołanie metody `checkForUpdate()` powoduje, że obiekt Updater asynchronicznie pobierze i zinterpretuje plik deskryptora. W wyniku wywołania metody `checkForUpdate()` obiekt Updater może wywołać następujące zdarzenia:

- `StatusUpdateEvent.UPDATE_STATUS` — obiekt Updater pomyślnie pobrał i zinterpretował plik deskryptora aktualizacji. To zdarzenie ma następujące właściwości:
 - `available` — wartość typu Boolean. Ustawiana na `true`, jeśli dostępna jest wersja aplikacji inna niż bieżąca; `false` w przeciwnym wypadku (wersje są takie same).
 - `version` — ciąg znaków (String). Wersja odczytana z pliku deskryptora aplikacji pliku aktualizacji.
 - `details` — tablica (Array). Jeśli nie istnieją zlokalizowane wersje opisu, jako pierwszy element tej tablicy jest zwracany pusty ciąg znaków (" "), a jako drugi element zwracany jest opis.

Jeśli istnieje wiele wersji opisu (w pliku deskryptora aplikacji), tablica zawiera wiele podtablic. Każda tablica ma dwa elementy: pierwszy jest kodem języka (na przykład "en"), a drugi jest odpowiednim opisem (typu String) w tym języku. Zobacz „[Definiowanie plików deskryptora aktualizacji i dodawanie pliku AIR do serwera internetowego](#)” na stronie 284.
- `StatusUpdateErrorEvent.UPDATE_ERROR` — wystąpił błąd i obiekt Updater nie mógł pobrać lub zinterpretować pliku deskryptora aktualizacji.

Pobieranie pliku AIR aktualizacji

Obiekt ApplicationUpdater wywołuje zdarzenie `updateStatus` po tym, jak pomyślnie pobierze i zinterpretuje plik deskryptora aktualizacji. Zachowanie domyślne polega na rozpoczęciu pobierania pliku aktualizacji, o ile jest on dostępny. Jeśli zachowanie domyślne zostanie anulowane, można wywołać metodę `downloadUpdate()` w celu wznowienia procesu aktualizacji.

```
appUpdater.downloadUpdate();
```

Wywołanie tej metody powoduje, że obiekt `Updater` rozpoczyna asynchroniczne pobieranie wersji pliku AIR będącej aktualizacją.

Metoda `downloadUpdate()` może wywoływać następujące zdarzenia:

- `UpdateEvent.DOWNLOAD_START` — zostało nawiązane połączenie z serwerem. W przypadku użycia biblioteki `ApplicationUpdaterUI` to zdarzenie powoduje wyświetlenie okna dialogowego z paskiem postępu pobierania.
- `ProgressEvent.PROGRESS` — wywoływane okresowo w trakcie pobierania pliku.
- `DownloadErrorEvent.DOWNLOAD_ERROR` — wywoływane, jeśli podczas nawiązywania połączenia lub pobierania pliku aktualizacji wystąpi błąd. To zdarzenie jest także wywoływane w wypadku odebrania niepoprawnego statusu HTTP (np. „404 - nie znaleziono pliku”). To zdarzenie ma właściwość `errorID` — liczbę całkowitą, która stanowi dodatkową informację o błędzie. Dodatkowa właściwość `subErrorID` może zawierać dalsze informacje o błędzie.
- `UpdateEvent.DOWNLOAD_COMPLETE` — obiekt `Updater` pomyślnie pobrał i zinterpretował plik deskryptora aktualizacji. Jeśli to zdarzenie nie zostanie anulowane, w wersji `ApplicationUpdater` nastąpi przejście do instalacji aktualizacji. W wersji `ApplicationUpdaterUI` zostanie wyświetlone okno dialogowe z opcją kontynuacji.

Aktualizowanie aplikacji

Obiekt `ApplicationUpdater` wywołuje zdarzenie `downloadComplete` po zakończeniu pobierania pliku aktualizacji. Jeśli zachowanie domyślne zostanie anulowane, można wywołać metodę `installUpdate()` w celu wznowienia procesu aktualizacji.

```
appUpdater.installUpdate(file);
```

Wywołanie tej metody powoduje, że obiekt `Updater` zainstaluje plik AIR aktualizacji. Metoda ma jeden parametr, `file`, który jest obiektem `File` wskazującym plik AIR aktualizacji.

Obiekt `ApplicationUpdater` może wywołać zdarzenie `beforeInstall` w wyniku wywołania metody `installUpdate()`:

- `UpdateEvent.BEFORE_INSTALL` — wywoływane tuż przed zainstalowaniem aktualizacji. Niekiedy celowe jest zablokowanie instalacji aktualizacji, aby użytkownik mógł dokończyć bieżące zadania przed jej rozpoczęciem. Metoda `preventDefault()` obiektu `Event` odracza instalację do następnego uruchomienia i uniemożliwia rozpoczęcie następnego procesu aktualizacji. (Dotyczy to także aktualizacji, które byłyby wynikiem wywołania metody `checkNow()` lub okresowego sprawdzania dostępności).

Instalowanie z arbitralnie wybranego pliku AIR

Istnieje możliwość wywołania metody `installFromAIRFile()` w celu zainstalowania zaktualizowanej wersji aplikacji z pliku AIR na komputerze użytkownika:

```
appUpdater.installFromAIRFile();
```

Metoda ta powoduje, że obiekt `Updater` instaluje aktualizację aplikacji z pliku AIR.

Metoda `installFromAIRFile()` może wywoływać następujące zdarzenia:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS` — wywoływane po tym, jak obiekt `ApplicationUpdater` pomyślnie sprawdzi poprawność pliku przy użyciu metody `installFromAIRFile()`. To zdarzenie ma następujące właściwości:
 - `available` — ustawiana na `true`, jeśli dostępna jest wersja aplikacji inna niż bieżąca; `false` w przeciwnym wypadku (wersje są takie same).
 - `version` — ciąg znaków reprezentujący nową dostępną wersję.

- `path` — rodzima ścieżka pliku aktualizacji.

Można anulować to zdarzenie, jeśli właściwość `available` obiektu `StatusFileUpdateEvent` jest ustawiona na `true`. Anulowanie zdarzenia powoduje anulowanie aktualizacji. Aby kontynuować anulowaną aktualizację, należy wywołać metodę `installUpdate()`.

- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR` — wystąpił błąd, a obiekt `Updater` nie mógł zainstalować aplikacji AIR.

Anulowanie procesu aktualizacji

Metoda `cancelUpdate()` umożliwia anulowanie procesu aktualizacji:

```
appUpdater.cancelUpdate();
```

Metoda ta anuluje wszystkie oczekujące operacje pobierania, usuwa wszystkie nie w pełni pobrane pliki i ponownie uruchamia zegar odliczający interwał między sprawdzeniami dostępności aktualizacji.

Metoda nie wykonuje żadnych czynności, jeśli trwa inicjowanie obiektu `Updater`.

Lokalizowanie interfejsu `ApplicationUpdaterUI`

Klasa `ApplicationUpdaterUI` udostępnia domyślny interfejs użytkownika procesu aktualizacji. W jego skład wchodzi okna dialogowe umożliwiające rozpoczęcie procesu, anulowanie procesu i wykonywanie innych pokrewnych czynności.

Element `description` pliku deskryptora aktualizacji umożliwia zdefiniowanie opisu aplikacji w wielu językach. Należy w tym celu użyć wielu elementów `text` definiujących atrybuty `lang`, co zilustrowano poniżej:

```
<?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>
```

Architektura aktualizacji wybiera opis najlepiej dopasowany do łańcucha lokalizacji użytkownika końcowego. Więcej informacji zawiera sekcja Definiowanie pliku deskryptora aktualizacji i dodawanie pliku AIR na serwer WWW.

Programiści korzystający ze środowiska Flex mogą bezpośrednio dodawać nowe języki do pakunku "ApplicationUpdaterDialogs".

W języku JavaScript można wywoływać metodę `addResources()` obiektu `Updater`. Ta metoda dynamicznie dodaje nowy pakunek zasobów dla języka. Pakunek zasobów definiuje zlokalizowane ciągi znaków w danym języku. Ciągi te są używane w polach tekstowych okien dialogowych.

W języku JavaScript można skorzystać z właściwości `localeChain` klasy `ApplicationUpdaterUI` w celu zdefiniowania łańcucha ustawień narodowych używanego w interfejsie użytkownika. Zazwyczaj z właściwości tej korzystają tylko programiści posługujący się językiem JavaScript (HTML). W środowisku Flex do zarządzania łańcuchem ustawień narodowych można używać menedżera zasobów.

Poniższy przykładowy kod w języku JavaScript definiuje pakunki zasobów dla języka rumuńskiego i węgierskiego:

```
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;
```

Szczegółowe informacje zawiera opis metody `addResources()` klasy `ApplicationUpdaterUI` w skorowidzu języka.

Rozdział 18: Wyświetlanie kodu źródłowego

Kod źródłowy aplikacji AIR z kodem HTML można wyświetlać, podobnie jak kod źródłowy strony HTML w przeglądarce internetowej. Program Adobe® AIR® SDK zawiera plik JavaScript AIRSourceViewer.js, który może być używany w aplikacji, aby ułatwić wyświetlanie kodu źródłowego użytkownikom końcowym.

Ładowanie, konfigurowanie i otwieranie przeglądarki źródła

Kod przeglądarki źródła jest zawarty w pliku JavaScript AIRSourceViewer.js, który znajduje się w katalogu infrastruktury pakietu AIR SDK. W celu korzystania z przeglądarki źródła w aplikacji należy skopiować plik AIRSourceViewer.js do katalogu projektu aplikacji i załadować go za pośrednictwem znacznika script w głównym pliku HTML aplikacji:

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

Plik AIRSourceViewer.js definiuje klasę SourceViewer, do której dostęp można uzyskać z kodu JavaScript poprzez wywołanie `air.SourceViewer`.

Klasa SourceViewer definiuje trzy metody: `getDefault()`, `setup()` i `viewSource()`.

Metoda	Opis
<code>getDefault()</code>	Metoda statyczna. Zwraca instancję klasy SourceViewer, której można użyć w celu wywołania innych metod.
<code>setup()</code>	Powoduje stosowanie ustawień konfiguracji do przeglądarki źródła. Szczegółowe informacje zawiera sekcja „Konfigurowanie przeglądarki źródła” na stronie 294
<code>viewSource()</code>	Otwiera nowe okno, w którym można przeglądać i otwierać pliki źródłowe aplikacji głównej.

Uwaga: Kod korzystający z przeglądarki źródła musi znajdować się w bezpiecznym obszarze izolowanym aplikacji (w pliku w katalogu aplikacji).

Przykład: poniższy kod JavaScript tworzy instancję klasy SourceViewer i otwiera okno przeglądarki źródła zawierające wszystkie pliki źródłowe:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Konfigurowanie przeglądarki źródła

Metoda `config()` powoduje stosowanie określonych ustawień do przeglądarki źródła. Ta metoda przyjmuje jeden parametr: `configObject`. Obiekt `configObject` zawiera właściwości, które definiują ustawienia konfiguracji dla przeglądarki źródła. Właściwości są następujące: `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` i `typesToAdd`.

default

Ciąg znaków określający ścieżkę względną do pliku początkowego, który ma być wyświetlany w przeglądarce źródła.

Na przykład: poniższy kod JavaScript powoduje otwarcie okna przeglądarki źródła z plikiem index.html wyświetlonym jako plik początkowy:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

Tablica ciągów znaków określająca pliki lub katalogi, które mają zostać wykluczone z listingu w przeglądarce źródła. Ścieżki są określone względem katalogu aplikacji. Znaki wieloznaczne nie są obsługiwane.

Na przykład: poniższy kod JavaScript powoduje otwarcie okna przeglądarki źródła z wszystkimi plikami źródłowymi oprócz pliku AIRSourceViewer.js oraz plików w podkatalogach Images i Sounds:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

Tablica, która zawiera dwie liczby określające początkowe współrzędne x i y okna przeglądarki źródła.

Na przykład: poniższy kod JavaScript powoduje otwarcie okna przeglądarki źródła na współrzędnych [40, 60] (X = 40, Y = 60):

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

Wartość logiczna, która określa, czy przeglądarka źródła powinna być oknem modalnym (true), czy niemodalnym (false). Domyślnie okno przeglądarki źródła jest modalne.

Na przykład: poniższy kod JavaScript powoduje otwarcie okna przeglądarki źródła w taki sposób, który umożliwia użytkownikowi pracę z oknem przeglądarki źródła i dowolnymi innymi oknami aplikacji:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

Tablica ciągów znaków określających typy plików, jakie będą uwzględniane w listingu przeglądarki źródła (oprócz typów domyślnych).

Domyślnie w przeglądarce źródła wyświetlane są pliki następujących typów:

- Pliki tekstowe — TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Pliki graficzne — JPG, JPEG, PNG, GIF

Jeśli nie określono żadnej wartości, wyświetlane będą wszystkie typy domyślne (z wyjątkiem określonych we właściwości typesToExclude).

Na przykład: poniższy kod JavaScript powoduje otwarcie okna przeglądarki źródła z plikami VCF i VCARD:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToAdd = ["text.vcf", "text.vcard"];
viewer.viewSource(configObj);
```

Dla każdego typu pliku należy określić "text" (w przypadku plików tekstowych) lub "image" (dla plików graficznych).

typesToExclude

Tablica ciągów znaków, która określa typy plików, które nie będą wyświetlane w przeglądarce źródła.

Domyślnie w przeglądarce źródła wyświetlane są pliki następujących typów:

- Pliki tekstowe — TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Pliki graficzne — JPG, JPEG, PNG, GIF

Na przykład poniższy kod JavaScript powoduje otwarcie okna przeglądarki źródła bez plików GIF i XML:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

Dla każdego typu pliku należy określić "text" (w przypadku plików tekstowych) lub "image" (dla plików graficznych).

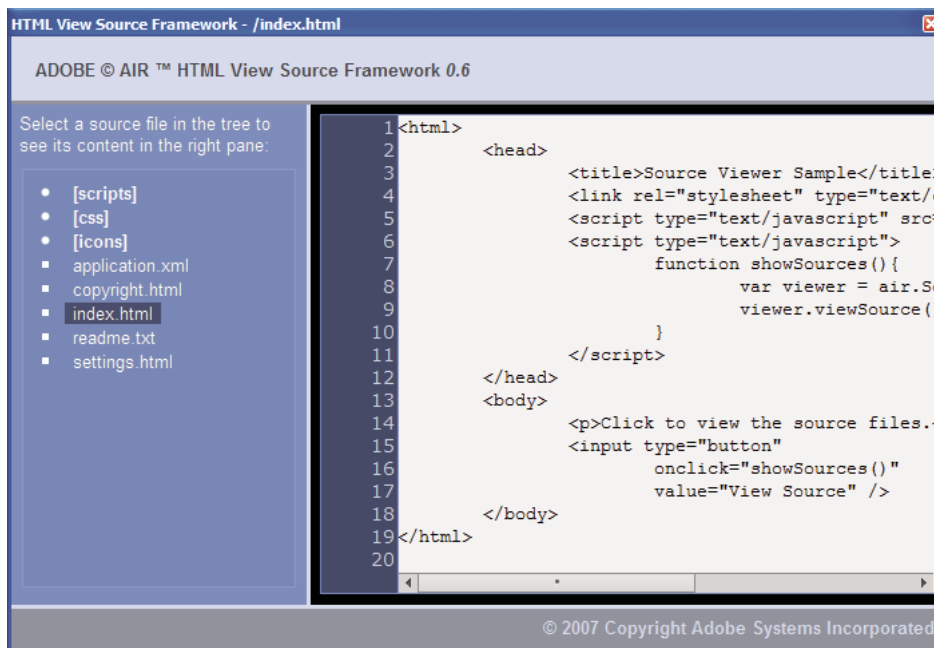
Otwieranie przeglądarki źródła

W aplikacji należy udostępnić element interfejsu użytkownika, taki jak odsyłacz, przycisk lub polecenie menu, którego kliknięcie będzie powodowało wywołanie kodu przeglądarki źródła. Przykład: poniższa prosta aplikacja otwiera przeglądarkę źródła, gdy użytkownik kliknie odsyłacz:

```
<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>
```

Interfejs użytkownika przeglądarki źródła

Gdy aplikacja wywoła metodę `viewSource()` obiektu `SourceViewer`, aplikacja AIR otwiera okno przeglądarki źródła. Okno zawiera listę plików źródłowych i katalogów (po lewej stronie) oraz obszar, w którym wyświetlany jest kod źródłowy wybranego pliku (po prawej stronie):



Nazwy katalogów są podawane w nawiasach kwadratowych. W celu rozwinięcia lub zwinięcia listingu katalogu wystarczy kliknąć nawias.

W przeglądarce można wyświetlać źródła plików tekstowych o znanych rozszerzeniach (takich jak HTML, JS, TXT i XML) oraz plików graficznych o znanych rozszerzeniach (JPG, JPEG, PNG i GIF). Jeśli użytkownik wybierze plik, którego rozszerzenie nie jest znane, pojawi się komunikat o błędzie („Nie można uzyskać treści tekstu z pliku tego typu”).

Pliki wykluczone za pomocą metody `setup()` nie są wyświetlane (informacje zawiera sekcja „Ładowanie, konfigurowanie i otwieranie przeglądarki źródła” na stronie 294).

Rozdział 19: Debugowanie za pomocą introspektora HTML w środowisku AIR

Pakiet SDK Adobe® AIR® zawiera plik JavaScript o nazwie AIRIntrospector.js, który można dołączyć do aplikacji, aby ułatwić debugowanie aplikacji HTML.

Informacje o introspektorze AIR

Introspektor aplikacji HTML/JavaScript w środowisku Adobe AIR (nazywany introspektorem HTML AIR) udostępnia użyteczne funkcje przeznaczone do wdrażania i debugowania aplikacji opartych na kodzie HTML:

- Do tych funkcji należy introspektor, który umożliwia wskazywanie na element interfejsu użytkownika w aplikacji w celu sprawdzenia oznaczenia i właściwości DOM tego elementu.
- Zawiera także konsolę przeznaczoną do wysyłania odwołań do obiektu w celu sprawdzenia, a także umożliwia dostosowywanie wartości właściwości i wykonywanie kodu JavaScript. Możliwa jest także serializacja obiektów do konsoli, co ogranicza możliwości edytowania danych. Można także kopiować i zapisywać tekst z konsoli.
- Introspektor zawiera widok drzewa dla właściwości i funkcji DOM.
- Umożliwia edytowanie atrybutów i węzłów tekstu dla elementów DOM.
- Zawiera listy odsyłaczy, stylów CSS, obrazów oraz plików JavaScript załadowanych do aplikacji.
- Umożliwia sprawdzanie pierwotnego źródła HTML oraz źródła bieżących oznaczeń dla interfejsu użytkownika.
- Umożliwia uzyskiwanie dostępu do plików w katalogu aplikacji. (Jest to możliwe tylko w przypadku konsoli introspektora HTML AIR otwartej dla obszaru izolowanego aplikacji. Nie jest możliwe w przypadku konsoli otwartych dla nieaplikacyjnych obszarów izolowanych).
- Zawiera przeglądarkę obiektów XMLHttpRequest oraz ich właściwości, łącznie z właściwościami `responseText` i `responseXML` (jeśli są dostępne).
- Zgodny tekst można wyszukiwać w kodzie źródłowym i plikach źródłowych.

Ładowanie kodu introspektora AIR

Kod introspektora AIR jest zawarty w pliku JavaScript AIRIntrospector.js, który znajduje się w katalogu infrastruktury pakietu AIR SDK. W celu korzystania z introspektora AIR w aplikacji należy skopiować plik AIRIntrospector.js do katalogu projektu aplikacji i załadować go za pośrednictwem znacznika `script` w głównym pliku HTML aplikacji:

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

Plik należy również dołączyć do każdego pliku HTML, który odpowiada różnym rodzimym oknom aplikacji.

Ważne: Plik `AIRIntrospector.js` należy dołączyć tylko wówczas, gdy wymagane jest wdrażanie i debugowanie aplikacji. Należy go usunąć z pakietu aplikacji AIR przeznaczonej do dystrybucji.

Plik AIRIntrospector.js definiuje klasę `Console`, do której dostęp można uzyskać z kodu JavaScript poprzez wywołanie `air.Introspector.Console`.

Uwaga: Kod korzystający z introspektora AIR musi znajdować się w bezpiecznym obszarze izolowanym aplikacji (w pliku w katalogu aplikacji).

Kontrola obiektu na karcie Konsola

Klasa `Console` definiuje pięć metod: `log()`, `warn()`, `info()`, `error()` i `dump()`.

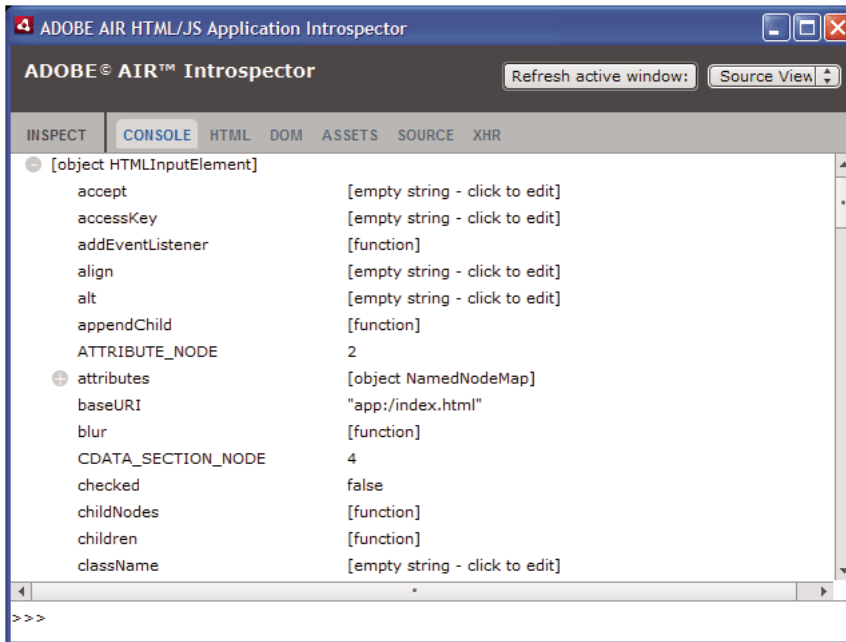
Metody `log()`, `warn()`, `info()` i `error()` umożliwiają wysłanie obiektu na kartę Konsola. Najprostszą z tych metod jest `log()`. Poniższy kod wysyła prosty obiekt, reprezentowany przez zmienną `test`, na kartę Konsola:

```
var test = "hello";  
air.Introspector.Console.log(test);
```

Jednak bardziej użyteczne jest wysłanie do karty Konsola obiektu złożonego. Na przykład: poniższa strona HTML zawiera przycisk (`btn1`), który wywołuje funkcję, która wysyła obiekt przycisku na kartę Konsola:

```
<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>
```


Po kliknięciu przycisku na karcie Konsola pojawi się obiekt btn1 i możliwe będzie rozwinięcie widoku drzewa obiektu w celu sprawdzenia jego właściwości:



W celu edytowania właściwości obiektu należy kliknąć listing po prawej stronie nazwy właściwości, a następnie zmodyfikować listing.

Metody `info()`, `error()` i `warn()` działają podobnie, jak metoda `log()`. Jednak wywołanie tych metod powoduje, że w konsoli wyświetlana jest ikona na początku wiersza:

Metoda	Ikona
<code>info()</code>	
<code>error()</code>	
<code>warn()</code>	

Metody `log()`, `warn()`, `info()` i `error()` wysyłają odwołanie tylko do rzeczywistego obiektu, dlatego dostępne są tylko właściwości w momencie wyświetlania. Jeśli wymagana jest serializacja obiektu rzeczywistego, należy użyć metody `dump()`. Metoda zawiera dwa parametry:

Parametr	Opis
<code>dumpObject</code>	Obiekt przeznaczony do serializacji.
<code>levels</code>	Maksymalna liczba poziomów do sprawdzenia w drzewie obiektu (oprócz poziomu głównego). Wartością domyślną jest 1 (co oznacza, że wyświetlany jest jeden poziom drzewa (oprócz poziomu głównego)). Ten parametr jest opcjonalny.

Wywołanie metody `dump()` powoduje serializację obiektu przed wysłaniem go do karty Konsola, co sprawia, że nie można edytować jego właściwości. Weźmy na przykład pod uwagę następujący kod:

```
var testObject = new Object();  
testObject.foo = "foo";  
testObject.bar = 234;  
air.Introspector.Console.dump(testObject);
```

Po wykonaniu tego kodu na karcie Konsola pojawia się obiekt `testObject` oraz jego właściwości, ale wartości właściwości nie można edytować w Konsoli.

Konfigurowanie introspektora AIR

Konsolę można skonfigurować poprzez ustawienie właściwości globalnej zmiennej `AIRIntrospectorConfig`. Na przykład: poniższy kod JavaScript konfiguruje Introspektora AIR w taki sposób, aby zawijał zawartość kolumn przy 100 znakach:

```
var AIRIntrospectorConfig = new Object();  
AIRIntrospectorConfig.wrapColumns = 100;
```

Właściwości zmiennej `AIRIntrospectorConfig` należy ustawić przed załadowaniem pliku `AIRIntrospector.js` (za pośrednictwem znacznika `script`).

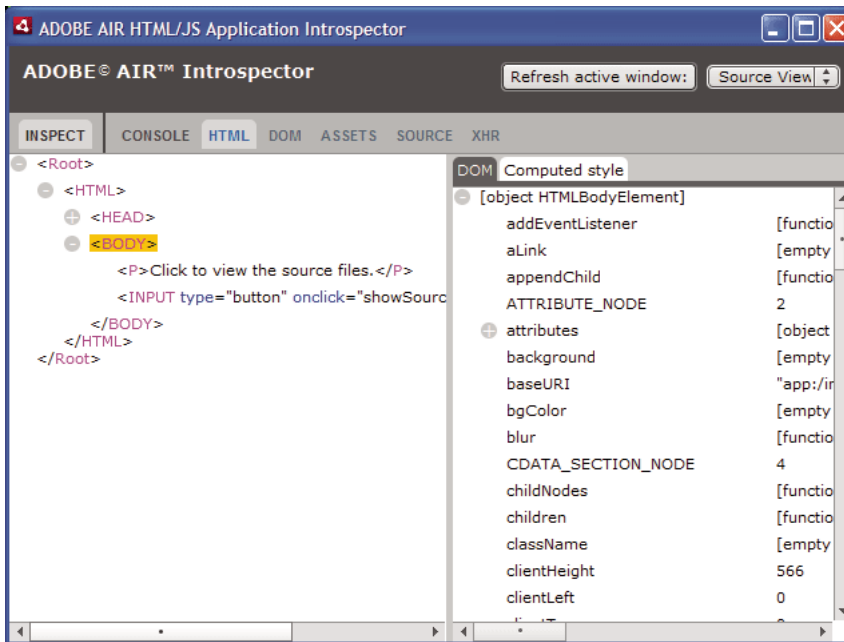
Zmienna `AIRIntrospectorConfig` ma osiem właściwości:

Właściwość	Wartość domyślna	Opis
<code>closeIntrospectorOnExit</code>	<code>true</code>	Ustawia okno Introspektor w taki sposób, że zostaje ono zamknięte, gdy wszystkie pozostałe okna aplikacji są zamknięte.
<code>debuggerKey</code>	123 (klawisz F12)	Kod skrótu klawiaturowego, który umożliwia wyświetlanie i ukrywanie okna Introspektor AIR.
<code>debugRuntimeObjects</code>	<code>true</code>	Ustawia introspektor w taki sposób, aby rozszerzał obiekty środowiska wykonawczego, a także obiekty zdefiniowane w JavaScript.
<code>flashTabLabels</code>	<code>true</code>	Powoduje, że karty Konsola i XMLHttpRequest migają przy każdej zmianie (na przykład, gdy dojdzie do zarejestrowania tekstu na kartach).
<code>introspectorKey</code>	122 (klawisz F11)	Kod skrótu klawiszowego przeznaczonego do otwierania panelu Sprawdź.
<code>showTimestamp</code>	<code>true</code>	Ustawia kartę Konsola w taki sposób, aby wyświetlała znaczniki czasu na początku każdego wiersza.
<code>showSender</code>	<code>true</code>	Ustawia kartę Konsola w taki sposób, aby na początku każdego wiersza wyświetlane były informacje o obiekcie wysyłającym komunikat.
<code>wrapColumns</code>	2000	Liczba kolumn, przy której pliki źródłowe są zawijane.

Interfejs introspektora AIR

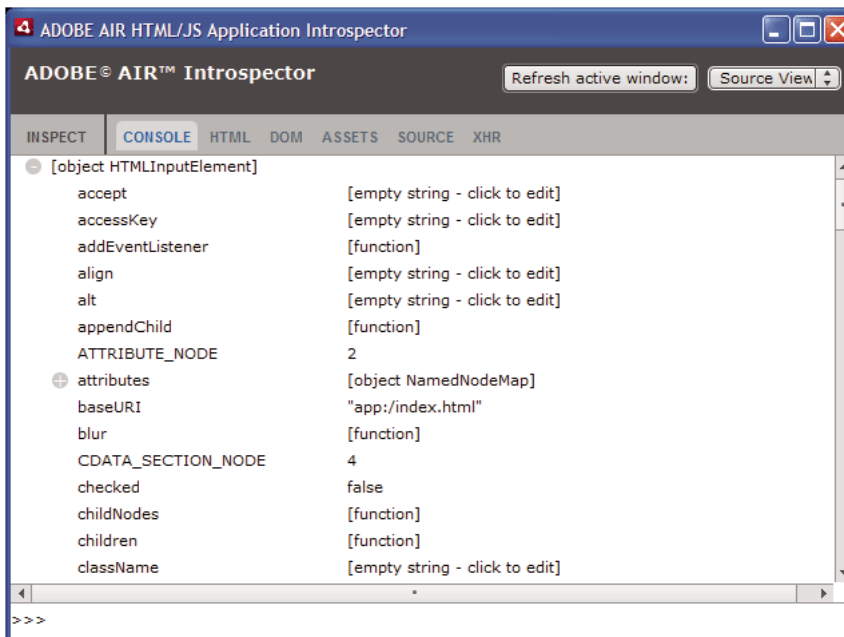
W celu otwarcia okna introspektora AIR podczas debugowania aplikacji należy nacisnąć klawisz F12 lub wywołać jedną z metod klasy `Console` (patrz „[Kontrola obiektu na karcie Konsola](#)” na stronie 299). Istnieje możliwość skonfigurowania innego klawisza aktywnego niż F12; informacje zawiera sekcja „[Konfigurowanie introspektora AIR](#)” na stronie 301.

Okno Introspektor AIR zawiera sześć kart — Konsola, HTML, DOM, Zasoby, Źródło i XHR — co przedstawiono na poniższej ilustracji:



Karta Konsola

Na karcie Konsola widoczne są wartości właściwości przekazanych jako parametry do jednej z metod klasy `air.Introspector.Console`. Szczegółowe informacje zawiera sekcja „[Kontrola obiektu na karcie Konsola](#)” na stronie 299.

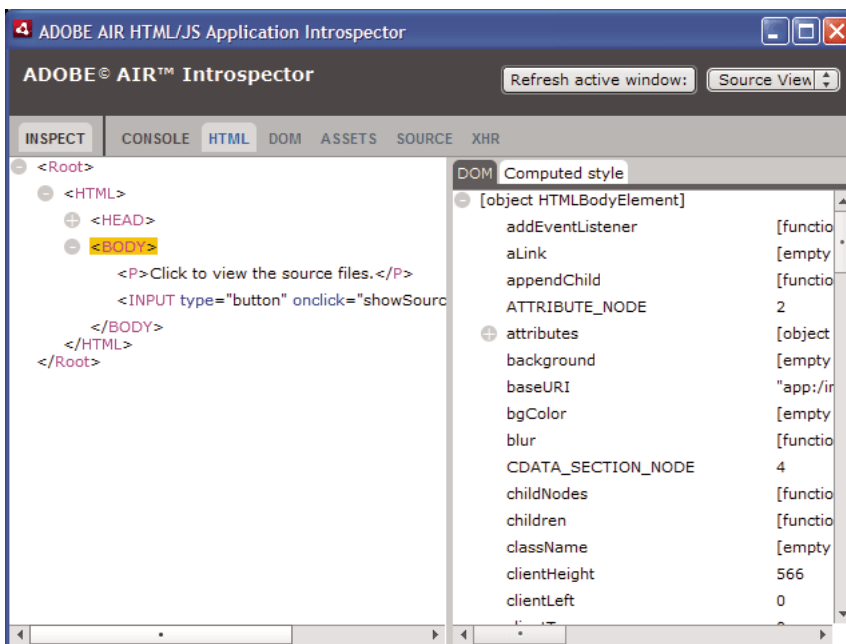


- Aby wyczyścić zawartość konsoli, należy kliknąć prawym przyciskiem myszy tekst i wybrać opcję Wyczyść konsolę.

- Aby zapisać tekst z karty Konsola w pliku, należy kliknąć prawym przyciskiem myszy kartę Konsola i wybrać opcję Zapisz konsolę w pliku.
- Aby zapisać tekst z karty Konsola w schowku, należy kliknąć prawym przyciskiem myszy kartę Konsola i wybrać opcję Zapisz konsolę w schowku. Aby skopiować tylko zaznaczony tekst do konsoli, należy kliknąć tekst prawym przyciskiem myszy i wybrać opcję Kopiuj.
- Aby zapisać tekst z klasy Console w pliku, należy kliknąć prawym przyciskiem myszy kartę Konsola i wybrać opcję Zapisz konsolę w pliku.
- W celu wyszukiwania zgodnego tekstu wyświetlanego na karcie należy nacisnąć klawisze CTRL+F w systemie Windows i Command+F w systemie Mac OS. (Niewidoczne węzły drzewa nie będą przeszukiwane).

Karta HTML

Na karcie HTML można wyświetlać całą strukturę DOM HTML w postaci drzewa. Kliknięcie elementu powoduje wyświetlenie jego właściwości po prawej stronie karty. Klikanie ikon + i - obok węzłów umożliwia rozwijanie i zwijanie węzłów w drzewie.



Na karcie HTML można edytować dowolne atrybuty i elementy tekstu, a zmiana wartości zostanie uwzględniona w aplikacji.

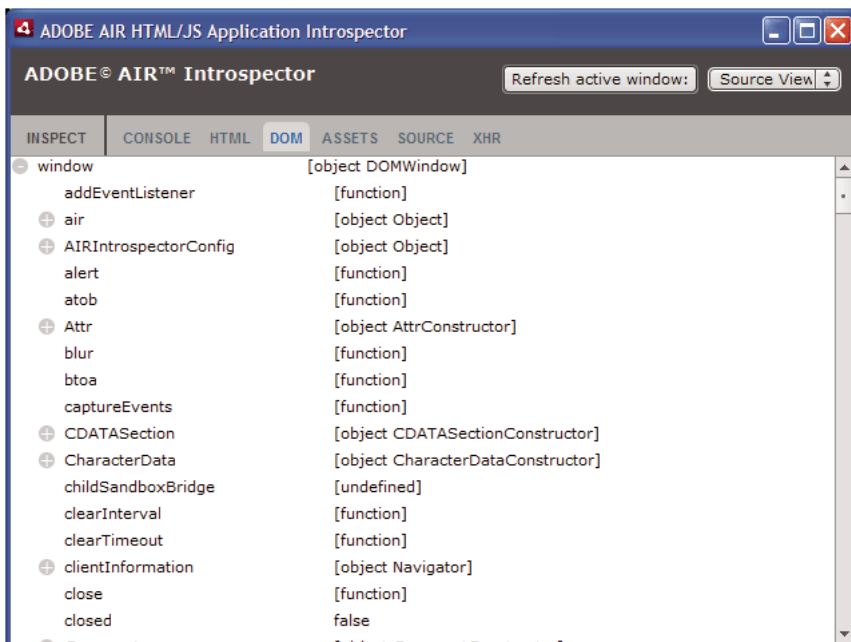
Kliknij przycisk Sprawdź (po lewej stronie listy kart w oknie Introspektor AIR). Można kliknąć dowolny element na stronie HTML głównego okna, a powiązany obiekt DOM zostanie wyświetlony na karcie HTML. Jeśli aktywne jest główne okno, można również nacisnąć skrót klawiaturowy, który powoduje aktywację i dezaktywację przycisku Sprawdź. Domyślnie jest to klawisz F11. Istnieje możliwość skonfigurowania innego klawisza aktywnego niż F11; informacje zawiera sekcja „Konfigurowanie introspektora AIR” na stronie 301.

Kliknięcie przycisku Odśwież aktywne okno (u góry okna Introspektor AIR) powoduje odświeżenie danych widocznych na karcie HTML.

W celu wyszukiwania tekstu na karcie należy nacisnąć klawisze CTRL+F (w systemie Windows) lub Command+F (w systemie Mac OS). (Niewidoczne węzły drzewa nie będą przeszukiwane).

Karta DOM

Karta DOM przedstawia obiekt okna w strukturze drzewa. Na tej karcie można edytować dowolne właściwości ciągów znaków i liczb, a zmiana wartości zostanie uwzględniona w aplikacji.

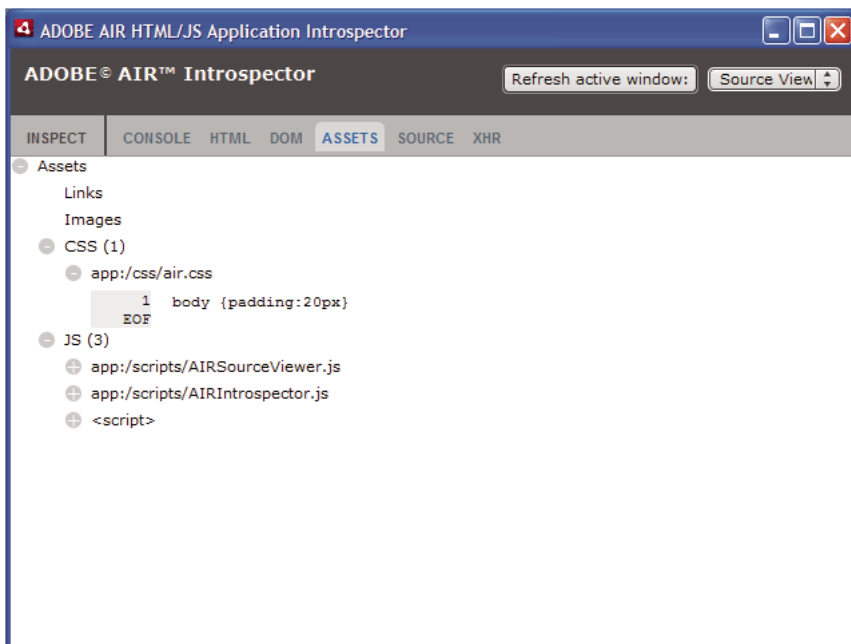


Kliknięcie przycisku Odśwież aktywne okno (u góry okna Introspektor AIR) powoduje odświeżenie danych widocznych na karcie DOM.

W celu wyszukiwania tekstu na karcie należy nacisnąć klawisze CTRL+F (w systemie Windows) lub Command+F (w systemie Mac OS). (Niewidoczne węzły drzewa nie będą przeszukiwane).

Karta Zasoby

Na karcie Zasoby można sprawdzać odsyłacze, obrazy, style CSS oraz pliki JavaScript załadowane w rodzimym oknie. Rozwinięcie jednego z tych węzłów powoduje wyświetlenie zawartości pliku lub wyświetlenie rzeczywistego używanego obrazu.



Kliknięcie przycisku Odśwież aktywne okno (u góry okna Introspektor AIR) powoduje odświeżenie danych widocznych na karcie Zasoby.

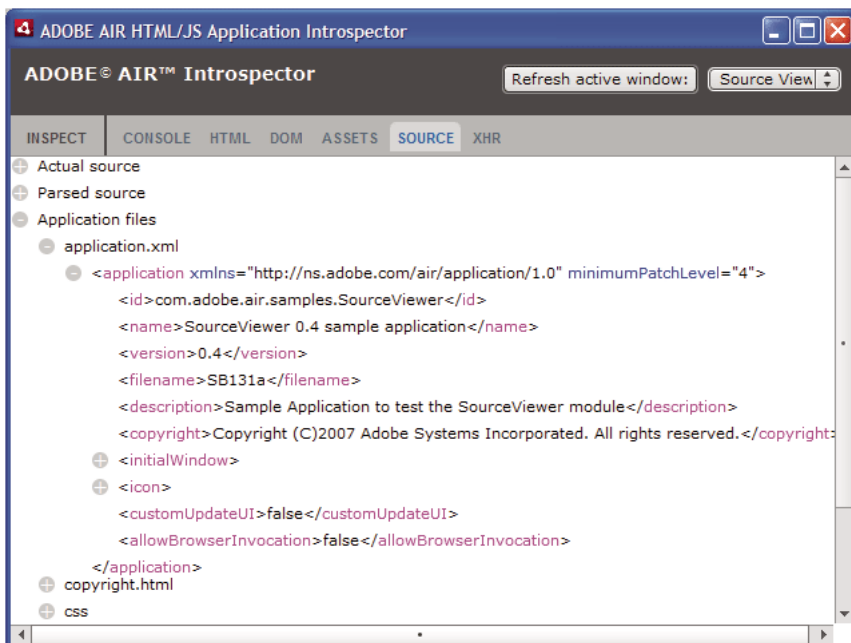
W celu wyszukiwania tekstu na karcie należy nacisnąć klawisze CTRL+F (w systemie Windows) lub Command+F (w systemie Mac OS). (Niewidoczne węzły drzewa nie będą przeszukiwane).

Karta Źródło

Karta Źródło zawiera trzy sekcje:

- Rzeczywiste źródło — przedstawia źródło HTML strony ładowanej jako główna treść podczas uruchamiania aplikacji.
- Źródło analizowane — przedstawia bieżące oznaczenie, które tworzy interfejs użytkownika aplikacji, co może się różnić od źródła rzeczywistego, ponieważ aplikacja generuje kod oznaczeń na bieżąco, korzystając z technologii Ajax.

- Pliki aplikacji — wyświetla pliki w katalogu aplikacji. Ten listing jest dostępny dla introspektora AIR, pod warunkiem że został uruchomiony z treści znajdującej w bezpiecznym obszarze izolowanym aplikacji. W tej sekcji można wyświetlać treść plików tekstowych, a także wyświetlać obrazy.

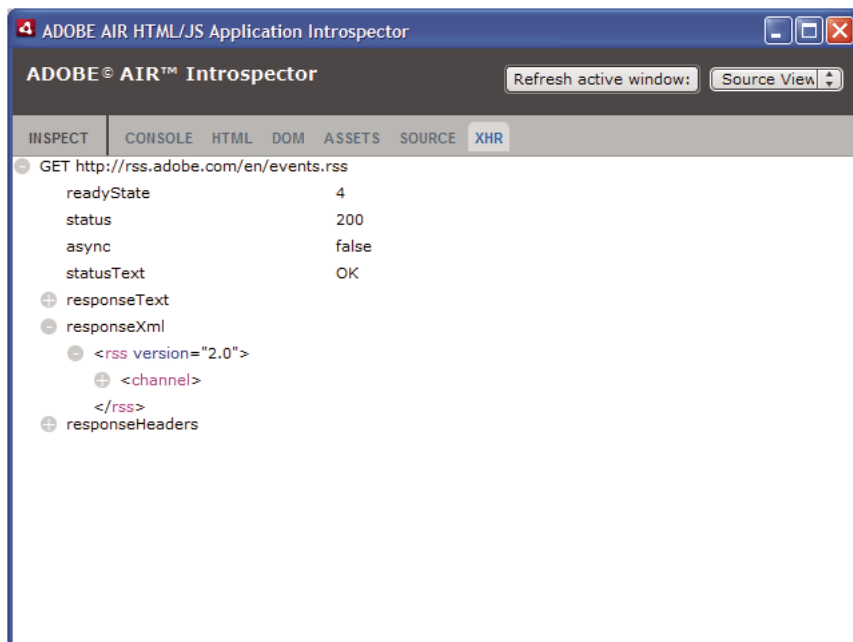


Kliknięcie przycisku Odśwież aktywne okno (u góry okna Introspektor AIR) powoduje odświeżenie danych widocznych na karcie Źródło.

W celu wyszukiwania tekstu na karcie należy nacisnąć klawisze CTRL+F (w systemie Windows) lub Command+F (w systemie Mac OS). (Niewidoczne węzły drzewa nie będą przeszukiwane).

Karta XHR

Karta XHR przechwytuje całą komunikację XMLHttpRequest w aplikacji oraz rejestruje informacje. Na tej karcie można wyświetlać właściwości XMLHttpRequest łącznie z `responseText` i `responseXML` (jeśli są dostępne) w strukturze drzewa.



W celu wyszukiwania tekstu na karcie należy nacisnąć klawisze CTRL+F (w systemie Windows) lub Command+F (w systemie Mac OS). (Niewidoczne węzły drzewa nie będą przeszukiwane).

Korzystanie z introspektora AIR z treścią w nieaplikacyjnym obszarze izolowanym

Treść z katalogu aplikacji można załadować do ramki i-frame lub ramki odwzorowanej na nieaplikacyjny obszar izolowany. Informacje zawiera sekcja [Zabezpieczenia HTML w środowisku Adobe AIR](#) (dla programistów ActionScript) oraz sekcja [Zabezpieczenia HTML w środowisku Adobe AIR](#) (dla programistów HTML). Introspektor AIR może być używany z taką treścią, ale należy przestrzegać następujących zasad:

- Plik AIRIntrospector.js należy umieścić w treści należącej do aplikacyjnego i niaplikacyjnego obszaru izolowanego (i-frame)
- Nie należy nadpisywać właściwości `parentSandboxBridge`; kod introspektora AIR korzysta z tej właściwości. W razie potrzeby należy dodać właściwości. Dlatego zamiast kodu:

```
parentSandboxBridge = mytrace: function(str) {runtime.trace(str)} ;
```

Należy użyć następującej składni:

```
parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
```


- Z treści zawartej w nieaplikacyjnym obszarze izolowanym nie można otworzyć introspektora AIR poprzez naciśnięcie klawisza F12 ani poprzez wywołanie jednej z metod w klasie `air.Introspector.Console`. Okno Introspektor można otworzyć tylko poprzez kliknięcie przycisku Otwórz introspektora. Domyślnie przycisk ten znajduje się w prawym górnym rogu ramki lub ramki i-frame. (Z powodu zabezpieczeń dotyczących treści nieaplikacyjnego obszaru izolowanego nowe okno może zostać otwarte tylko w wyniku gestu użytkownika, takiego jak kliknięcie przycisku).
- Dla aplikacyjnego i nieaplikacyjnego obszaru izolowanego można otworzyć osobne okna Introspektor AIR. Okna można rozróżnić na podstawie tytułów widocznych w oknach Introspektor AIR.
- Jeśli introspektor AIR został uruchomiony z nieaplikacyjnego obszaru izolowanego, wówczas na karcie Źródło nie będą widoczne pliki aplikacji.
- Introspektor AIR może sprawdzać tylko kod zawarty w obszarze izolowanym, z którego został otwarty,

Rozdział 20: Lokalizowanie aplikacji AIR

Adobe AIR 1.1 i wersje późniejsze

Środowisko Adobe® AIR® udostępnia funkcje obsługi dla wielu języków.

Informacje o lokalizowaniu treści w języku ActionScript 3.0 i środowisku Flex zawiera sekcja „Lokalizowanie aplikacji” w podręczniku dla programistów ActionScript 3.0.

Języki obsługiwane w AIR

Funkcje obsługi lokalizacji dla aplikacji AIR w następujących językach zostały wprowadzone do wersji AIR 1.1:

- chiński uproszczony
- chiński tradycyjny
- francuski
- niemiecki
- włoski
- japoński
- koreański
- portugalski (Brazylia)
- rosyjski
- hiszpański

W wersji AIR 1.5 dodano następujące języki:

- czeski
- holenderski
- polski
- szwedzki
- turecki

Więcej tematów Pomocy

[Budowanie wielojęzycznych aplikacji Flex w Adobe AIR](#)

[Budowanie wielojęzycznej aplikacji opartej na kodzie HTML](#)

Lokalizowanie nazwy i opisu aplikacji w instalatorze aplikacji AIR

Adobe AIR 1.1 i wersje późniejsze

Elementy `name` i `description` w pliku deskryptora aplikacji można określić w wielu językach. Przykład: poniżej przedstawiono nazwę aplikacji w trzech językach (angielskim, francuskim i niemieckim):

```
<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>
```

Atrybut `xml:lang` dla każdego elementu tekstowego określa kod języka zdefiniowany w dokumencie RFC4646 (<http://www.ietf.org/rfc/rfc4646.txt>).

Element `name` definiuje nazwę aplikacji, którą wyświetla instalator aplikacji AIR. Instalator aplikacji AIR korzysta ze zlokalizowanej wartości, która najlepiej pasuje do języków interfejsu użytkownika zdefiniowanych przez ustawienia systemu operacyjnego.

W pliku deskryptora aplikacji można również określić wiele wersji językowych elementu `description`. Ten element zawiera tekst opisu, który wyświetla instalator aplikacji AIR.

Te ustawienia mają zastosowanie tylko do języków dostępnych w instalatorze aplikacji AIR. Nie definiują ustawień narodowych dostępnych dla działających, zainstalowanych aplikacji. Aplikacje AIR mogą udostępniać interfejsy obsługujące wiele języków, łącznie z językami dostępnymi w instalatorze aplikacji AIR.

Więcej informacji zawiera sekcja „[Elementy deskryptora aplikacji AIR](#)” na stronie 221.

Więcej tematów Pomocy

[Budowanie wielojęzycznych aplikacji Flex w Adobe AIR](#)

[Budowanie wielojęzycznej aplikacji opartej na kodzie HTML](#)

Lokalizowanie treści HTML w strukturze lokalizowania AIR HTML

Adobe AIR 1.1 i wersje późniejsze

Pakiet SDK AIR 1.1 zawiera strukturę lokalizowania HTML. Strukturę definiuje plik JavaScript `AIRLocalizer.js`. Katalog `frameworks` directory pakietu SDK AIR zawiera plik `AIRLocalizer.js`. Ten plik zawiera klasę `air.Localizer`, która udostępnia funkcje ułatwiające tworzenie aplikacji, które obsługują wiele zlokalizowanych wersji.

Ładowanie kodu struktury lokalizowania HTML

W celu użycia struktury lokalizowania należy skopiować do projektu plik `AIRLocalizer.js`. Następnie należy dołączyć go do głównego pliku HTML aplikacji, stosując znacznik `script`:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

Następnie kod JavaScript może wywołać obiekt `air.Localizer.localizer`:

```
<script>
  var localizer = air.Localizer.localizer;
</script>
```

Obiekt `air.Localizer.localizer` jest obiektem singletonowym, który definiuje metody i właściwości przeznaczone do korzystania ze zlokalizowanych zasobów i zarządzania nimi. Klasa `Localizer` zawiera następujące metody:

Metoda	Opis
<code>getFile()</code>	Pobiera tekst określonego pakunku zasobów dla wybranych ustawień narodowych. Zobacz „ Pobieranie zasobów dla określonych ustawień narodowych ” na stronie 317.
<code>getLocaleChain()</code>	Zwraca języki do łańcucha ustawień narodowych. Zobacz „ Definiowanie łańcucha ustawień narodowych ” na stronie 316.
<code>getResourceBundle()</code>	Zwraca klucze pakunku oraz odpowiednie wartości jako obiekt. Zobacz „ Pobieranie zasobów dla określonych ustawień narodowych ” na stronie 317.
<code>getString()</code>	Pobiera ciąg znaków zdefiniowany dla zasobu. Patrz „ Pobieranie zasobów dla określonych ustawień narodowych ” na stronie 317.
<code>setBundlesDirectory()</code>	Ustawia lokalizację katalogu bundles. Zobacz „ Dostosowywanie ustawień obiektu Localizer HTML AIR ” na stronie 315.
<code>setLocalAttributePrefix()</code>	Ustawia przedrostek używany przez atrybuty obiektu Localizer używane w elementach HTML DOM. Zobacz „ Dostosowywanie ustawień obiektu Localizer HTML AIR ” na stronie 315.
<code>setLocaleChain()</code>	Ustawia kolejność języków w łańcuchu ustawień narodowych. Zobacz „ Definiowanie łańcucha ustawień narodowych ” na stronie 316.
<code>sortLanguagesByPreference()</code>	Sortuje ustawienia narodowe w łańcuchu tych ustawień na podstawie ich kolejności zdefiniowanej w systemie operacyjnym. Zobacz „ Definiowanie łańcucha ustawień narodowych ” na stronie 316.
<code>update()</code>	Aktualizuje HTML DOM (lub element DOM) z uwzględnieniem zlokalizowanych ciągów znaków dla bieżącego łańcucha ustawień narodowych. Opis łańcuchów ustawień regionalnych zawiera rozdział „ Zarządzanie łańcuchami ustawień narodowych ” na stronie 313. Więcej informacji o metodzie <code>update()</code> zawiera sekcja „ Aktualizowanie elementów DOM w celu użycia bieżących ustawień narodowych ” na stronie 314.

Klasa Localizer zawiera następujące właściwości statyczne:

Właściwość	Opis
<code>localizer</code>	Zwraca odwołanie do obiektu singletonowego Localizer dla aplikacji.
<code>ultimateFallbackLocale</code>	Ustawienia narodowe używane wówczas, gdy aplikacja nie obsługuje preferencji użytkownika. Zobacz „ Definiowanie łańcucha ustawień narodowych ” na stronie 316.

Określanie obsługiwanych języków

Element `<supportedLanguages>` w pliku deskryptora aplikacji pozwala określić języki obsługiwane przez aplikację. Ten element jest używany tylko w systemie iOS, w dołączanym środowisku wykonawczym dla komputerów Mac i aplikacjach dla systemu Android. Jest ignorowany w aplikacjach wszystkich innych typów.

Jeśli element `<supportedLanguages>` nie zostanie określony, pakowacz domyślnie wykona następujące operacje zależnie od typu aplikacji:

- System iOS: Na liście języków obsługiwanych przez aplikację w sklepie App Store dla systemu iOS zostaną umieszczone wszystkie języki obsługiwane przez środowisko wykonawcze AIR.
- Środowisko dołączane dla komputerów Mac: Aplikacja spakowana razem ze środowiskiem nie będzie zawierać informacji o tłumaczeniach.
- Android: Pakiet aplikacji zawiera zasoby dla wszystkich języków obsługiwanych przez środowisko wykonawcze AIR.

Więcej informacji zawiera „[supportedLanguages](#)” na stronie 252.

Definiowanie pakunków zasobów

Struktura lokalizowania HTML odczytuje zlokalizowane wersje ciągów znaków z plików *lokalizacji*. Plik lokalizacji jest kolekcją wartości opartych na kluczu, serializowanych w pliku tekstowym. Plik lokalizacji jest czasami określanymi jako *pakunek*.

Należy utworzyć podkatalog katalogu projektu aplikacji, o nazwie locale. (Można użyć innej nazwy. Zobacz „[Dostosowywanie ustawień obiektu Localizer HTML AIR](#)” na stronie 315). Ten katalog będzie zawierał pliki lokalizacji. Ten katalog jest określanymi jako *katalog bundles*.

Dla każdego ustawienia narodowego obsługiwane przez aplikację należy utworzyć podkatalog katalogu bundles. Każdy podkatalog należy nazwać zgodnie z kodem ustawień narodowych. Na przykład: katalog języka francuskiego powinien mieć nazwę „fr”, a katalog języka angielskiego nazwę „en”. W celu zdefiniowania ustawień narodowych z kodem języka i kraju można użyć znaku podkreślenia (_). Np. katalog języka angielskiego dla USA powinien mieć nazwę „en_us”. (Zamiast znaku podkreślenia można użyć kreski, np. „en-us”. Struktura lokalizowania HTML obsługuje obydwie te znaki).

Do podkatalogu ustawień narodowych można dodać dowolną liczbę plików zasobów. Zwykle plik lokalizacji jest tworzony dla każdego języka (plik należy umieścić w katalogu dla tego języka). Struktura lokalizowania kodu HTML zawiera metodę `getFile()`, która umożliwia odczyt treści pliku (zobacz „[Pobieranie zasobów dla określonych ustawień narodowych](#).” na stronie 317).

Pliki z rozszerzeniem `.properties` są plikami właściwości lokalizacji. W tych plikach można zdefiniować wartości klucz-wartość dla ustawień narodowych. Plik właściwości definiuje jedną wartość ciągu znaków w każdej linii. Przykład: poniższy kod definiuje ciąg znaków "Hello in English." dla klucza o nazwie `greeting`:

```
greeting=Hello in English.
```

Plik właściwości definiujący poniższy tekst definiuje sześć par klucz-wartość:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Ten przykład prezentuje angielską wersję pliku właściwości, który może być zapisany w katalogu `en`.

Francuska wersja pliku właściwości zostanie umieszczona w katalogu `fr`:

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Dla różnych typów informacji można zdefiniować wiele plików zasobów. Na przykład: plik `legal.properties` może zawierać szablon tekstu prawnego (np. informacje o prawach autorskich). Zasób taki można wykorzystywać wielokrotnie, w wielu aplikacjach. I podobnie — możliwe jest zdefiniowanie osobnych plików, które będą lokalizowały treść dla różnych części interfejsu użytkownika.

W celu zapewnienia obsługi wielu języków należy dla tych plików stosować kodowanie UTF-8.

Zarządzanie łańcuchami ustawień narodowych

Gdy aplikacja ładuje plik AIRLocalizer.js, sprawdza ustawienia narodowe zdefiniowane w aplikacji. Te ustawienia narodowe odpowiadają podkatalogom katalogu bundles (patrz „[Definiowanie pakunków zasobów](#)” na stronie 312). Ta lista dostępnych ustawień narodowych jest znana jako *łańcuch ustawień narodowych*. Plik AIRLocalizer.js automatycznie sortuje łańcuch ustawień narodowych na podstawie preferowanej kolejności zdefiniowanej w systemie operacyjnym. (Właściwość `Capabilities.languages` zawiera listę języków interfejsu użytkownika systemu operacyjnego, w preferowanej kolejności).

Dlatego jeśli aplikacja zdefiniuje zasoby dla ustawień narodowych „en”, „en_US” i „en_UK”, wówczas struktura obiektu Localizer AIR HTML sortuje odpowiednio łańcuch ustawień narodowych. Gdy aplikacja uruchomi system, który zgłasza „en” jako podstawowe ustawienie narodowe, wówczas łańcuch ustawień narodowych jest sterowany w taki sposób: [“en”, “en_US”, “en_UK”]. W takim przypadku aplikacja najpierw wyszukuje zasoby w pakunku „en”, a następnie w pakunku „en_US”.

Jeśli jednak system zgłasza „en-US” jako podstawowe ustawienie narodowe, wówczas sortowanie jest następujące: [“en_US”, “en”, “en_UK”]. W takim przypadku aplikacja najpierw wyszukuje zasoby w pakunku „en_US”, a następnie w pakunku „en”.

Domyślnie aplikacja definiuje pierwsze ustawienie narodowe w łańcuchu lokalizacji jako domyślne ustawienie narodowe do użytku. Użytkownik może wybrać ustawienia narodowe po pierwszym uruchomieniu aplikacji. Następnie możliwe jest zapisanie wybranych ustawień w pliku preferencji i wykorzystanie tych ustawień narodowych przy kolejnym uruchamianiu aplikacji.

Aplikacja może korzystać z ciągów zasobów w dowolnych ustawieniach narodowych z łańcucha ustawień narodowych. Jeśli określone ustawienia narodowe nie zdefiniują ciągu znaków zasobu, aplikacja korzysta z kolejnego zgodnego ciągu zasobu dla innych ustawień narodowych zdefiniowanych w łańcuchu tych ustawień.

Łańcuch ustawień narodowych można dostosować poprzez wywołanie metody `setLocaleChain()` obiektu Localizer. Patrz „[Definiowanie łańcucha ustawień narodowych](#)” na stronie 316.

Aktualizowanie elementów DOM za pomocą treści zlokalizowanej

Element aplikacji może odwoływać się do wartości klucza w pliku właściwości lokalizacji. Na przykład: element `title` w poniższym przykładzie określa atrybut `local_innerHTML`. Struktura lokalizacji korzysta z tego atrybutu w celu wyszukania zlokalizowanej wartości. Domyślnie struktura wyszukuje nazwy atrybutów, które rozpoczynają się od znaków „local_”. Struktura aktualizuje atrybuty z nazwami zgodnymi z tekstem za znakami „local_”. W takim przypadku struktura ustawia atrybut `innerHTML` elementu `title`. Atrybut `innerHTML` korzysta z wartości zdefiniowanej dla klucza `mainWindowTitle` w domyślnym pliku właściwości (`default.properties`):

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Jeśli bieżące ustawienia narodowe definiują wartość niezgodną, wówczas struktura obiektu Localizer wyszukuje pozostałą część łańcucha ustawień narodowych. Korzysta z następných ustawień narodowych z łańcucha ustawień, dla którego zdefiniowano wartość.

W poniższym przykładzie tekst (atrybut `innerHTML`) elementu `p` korzysta z wartości klucza `greeting` zdefiniowanego w domyślnym pliku właściwości:

```
<p local_innerHTML="default.greeting" />
```

W poniższym przykładzie atrybut `value` (i wyświetlony tekst) elementu `input` korzysta z wartości klucza `btnBlue` z domyślnego pliku właściwości:

```
<input type="button" local_value="default.btnBlue" />
```

W celu zaktualizowania modelu DOM HTML w taki sposób, aby korzystał z ciągów znaków zdefiniowanych w bieżącym łańcuchu ustawień narodowych, należy wywołać metodę `update()` obiektu `Localizer`. Wywołanie metody `update()` powoduje, że obiekt `Localizer` analizuje model DOM i stosuje manipulowanie w miejscach, w których znajduje atrybuty lokalizacji ("`local_...`");

```
air.Localizer.localizer.update();
```

Możliwe jest zdefiniowanie wartości dla atrybutu (takiego jak „`innerHTML`”) oraz odpowiadającego mu atrybutu lokalizacji (np. „`local_innerHTML`”). W takim przypadku struktura lokalizowania zastępuje tylko wartość atrybutu, jeśli znajdzie zgodną wartość w łańcuchu lokalizacji. Przykład: poniższy element definiuje atrybuty `value` i `local_value`:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

Możliwe jest również zaktualizowanie wybranego elementu modelu DOM. Informacje zawiera następująca sekcja „[Aktualizowanie elementów DOM w celu użycia bieżących ustawień narodowych](#)” na stronie 314.

Domyślnie obiekt `Localizer HTML AIR` korzysta z przedrostka `local_` dla atrybutów definiujących ustawienia lokalizacji dla elementu. Na przykład: domyślnie atrybut `local_innerHTML` definiuje nazwę pakunku i zasobu używaną dla wartości `innerHTML` elementu. Ponadto domyślnie atrybut `local_value` definiuje pakunek i nazwę zasobu dla atrybutu `value` elementu. Możliwe jest takie skonfigurowanie obiektu `Localizer`, aby korzystał on z przedrostka atrybutu innego niż "`local_`". Zobacz „[Dostosowywanie ustawień obiektu Localizer HTML AIR](#)” na stronie 315.

Aktualizowanie elementów DOM w celu użycia bieżących ustawień narodowych

Gdy obiekt `Localizer` zaktualizuje model DOM HTML, powoduje, że oznaczone elementy korzystają z wartości atrybutów na podstawie ciągów znaków zdefiniowanych w bieżącym łańcuchu ustawień narodowych. Aby obiekt `localizer HTML` zaktualizował model DOM HTML, należy wywołać metodę `update()` obiektu `Localizer`:

```
air.Localizer.localizer.update();
```

W celu zaktualizowania tylko określonego elementu DOM należy przekazać ten element jako parametr metody `update()`. Metoda `update()` zawiera tylko jeden parametr, `parentNode`, który jest opcjonalny. Po określeniu parametru `parentNode` definiuje on element DOM przeznaczony do zlokalizowania. Wywołanie metody `update()` i określenie parametru `parentNode` ustawia zlokalizowane wartości jako elementy podrzędne, które określają atrybuty lokalizacji.

Przykładem może być poniższy element `div`:

```
<div id="colorsDiv">
  <h1 local_innerHTML="default.lblColors" ></h1>
  <p><input type="button" local_value="default.btnBlue" /></p>
  <p><input type="button" local_value="default.btnRed" /></p>
  <p><input type="button" local_value="default.btnGreen" /></p>
</div>
```

W celu zaktualizowania elementu do użycia zlokalizowanych ciągów znaków zdefiniowanych w bieżącym łańcuchu ustawień narodowych należy użyć poniższego kodu JavaScript:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Jeśli wartość klucza nie zostanie znaleziona w łańcuchu ustawień narodowych, wówczas struktura lokalizacji ustawia wartość atrybutu na wartość atrybutu "local_". Przykład: założymy, że w poprzednim przykładzie struktura lokalizowania nie może znaleźć wartości dla klucza lblColors (w żadnym z plików default.properties w łańcuchu ustawień narodowych). W takim przypadku użyje "default.lblColors" jako wartości innerHTML. Użycie tej wartości oznacza (dla programisty) brak zasobów.

Metoda update() wywołuje zdarzenie resourceNotFound, nawet jeśli nie może znaleźć zasobu w łańcuchu ustawień narodowych. Stała air.Localizer.RESOURCE_NOT_FOUND definiuje ciąg znaków "resourceNotFound". Zdarzenie ma trzy właściwości: bundleName, resourceName i locale. Właściwość bundleName jest nazwą pakietu, w którym nie można znaleźć zasobu. Właściwość resourceName jest nazwą pakunku, w którym nie można znaleźć zasobu. Właściwość locale jest nazwą ustawień regionalnych, w których nie można znaleźć zasobu.

Metoda update() wywołuje zdarzenie bundleNotFound, gdy nie może znaleźć określonego pakunku. Stała air.Localizer.BUNDLE_NOT_FOUND definiuje ciąg znaków "bundleNotFound". Zdarzenie ma dwie właściwości: bundleName i locale. Właściwość bundleName jest nazwą pakunku, w którym nie można znaleźć zasobu. Właściwość locale jest nazwą ustawień narodowych, w których nie można znaleźć zasobu.

Metoda update() działa w sposób asynchroniczny (i wywołuje asynchronicznie zdarzenia resourceNotFound i bundleNotFound). Poniższy kod ustawia detektory zdarzeń dla zdarzeń resourceNotFound i 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);
}
```

Dostosowywanie ustawień obiektu Localizer HTML AIR

Metoda setBundlesDirectory() obiektu Localizer umożliwia dostosowanie ścieżki katalogu bundles. Metoda setLocalAttributePrefix() obiektu Localizer umożliwia dostosowanie ścieżki katalogu bundles oraz dostosowanie wartości atrybutu używanego przez obiekt Localizer.

Domyślnie katalog bundles jest zdefiniowany jako podkatalog locale katalogu aplikacji. Za pomocą metody setBundlesDirectory() obiektu Localizer można określić inny katalog. Ta metoda przyjmuje jeden parametr, path, który jest ścieżką żądanego katalogu bundles, w postaci ciągu znaków. Parametr path może mieć jedną z następujących wartości:

- Ciąg znaków definiujący ścieżkę względną do katalogu aplikacji, np. "locales"
- Ciąg znaków definiujący poprawny adres URL, który korzysta ze schematów app, app-storage lub file URL, takich jak "app://languages" (nie należy używać schematu http URL)
- Obiekt File

Informacje na temat adresów URL i ścieżek katalogów zawiera sekcja:

- [Ścieżki obiektów File](#) (dla programistów ActionScript)
- [Ścieżki obiektów File](#) (dla programistów HTML)

Przykład: poniższy kod ustawia katalog bundles jako podkatalog languages katalogu zapisu aplikacji (nie jako katalog aplikacji):


```
air.Localizer.localizer.setBundlesDirectory("languages");
```

Należy przekazać poprawną ścieżkę jako parametr `path`. W przeciwnym wypadku metoda zwróci wyjątek `BundlePathNotFoundError`. Właściwość `name` tego błędu to `"BundlePathNotFoundError"`, a właściwość `message` określa niepoprawną ścieżkę.

Domyślnie obiekt `Localizer HTML AIR` korzysta z `"local_"` jako z przedrostka dla atrybutów definiujących ustawienia lokalizacji dla elementu. Na przykład: atrybut `local_innerHTML` definiuje nazwę pakunku i nazwę zasobu używaną dla wartości `innerHTML` poniższego elementu `input`:

```
<p local_innerHTML="default.greeting" />
```

Metoda `setLocalAttributePrefix()` obiektu `Localizer` umożliwia korzystanie z przedrostka atrybutu innego niż `"local_"`. Ta statyczna metoda przyjmuje jeden parametr, który jest ciągiem znaków używanym jako przedrostek atrybutu. Przykład: poniższy kod ustawia strukturę lokalizowania w taki sposób, aby użyć „`loc_`” jako przedrostka aplikacji:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Istnieje możliwość dostosowania przedrostka atrybutu, z którego korzysta struktura lokalizowania. Możliwe jest dostosowanie przedrostka, jeśli wartość domyślna (`"local_"`) powoduje konflikt z nazwą innego atrybutu używanego w kodzie. Podczas wywoływania tej metody należy stosować poprawne znaki dla atrybutów HTML. (Na przykład: wartość nie może zawierać znaku spacji).

Więcej informacji na temat korzystania z atrybutów lokalizacji w elementach HTML zawiera sekcja „[Aktualizowanie elementów DOM za pomocą treści zlokalizowanej](#)” na stronie 313.

Ustawienia katalogu `bundles` i przedrostka aplikacji nie są zachowywane między poszczególnymi sesjami aplikacji. Jeśli używany jest niestandardowy katalog `bundles` lub niestandardowe ustawienie przedrostka atrybutu, należy określić to ustawienie każdorazowo przy inicjowaniu aplikacji.

Definiowanie łańcucha ustawień narodowych

Domyślnie załadowanie kodu `AIRLocalizer.js` powoduje ustawienie domyślnego łańcucha ustawień narodowych. Ustawienia narodowe dostępne w katalogu `bundles` oraz ustawienia języka systemu operacyjnego definiują ten łańcuch ustawień narodowych. (Szczegółowe informacje zawiera sekcja „[Zarządzanie łańcuchami ustawień narodowych](#)” na stronie 313).

Łańcuch ustawień narodowych można zmodyfikować poprzez wywołanie statycznej metody `setLocaleChain()` obiektu `Localizer`. Na przykład: tę metodę można wywołać, jeśli użytkownik preferuje konkretny język. Metoda `setLocaleChain()` przyjmuje jeden parametr `chain`, który jest tablicą ustawień narodowych, np. `["fr_FR", "fr", "fr_CA"]`. Kolejność ustawień narodowych w tablicy określa kolejność, w jakiej struktura wyszukuje zasoby (w kolejnych operacjach). Jeśli zasób nie zostanie znaleziony dla pierwszego ustawienia narodowego w łańcuchu, wówczas będzie przeszukiwać zasoby innego ustawienia narodowego. Jeśli argument `chain` nie jest dostępny, nie jest tablicą lub jest pustą tablicą, wówczas działanie funkcji kończy się niepowodzeniem i następuje wywołanie wyjątku `IllegalArgumentsError`.

Statyczna metoda `getLocaleChain()` obiektu `Localizer` zwraca tablicę zawierającą ustawienia narodowe z bieżącego łańcucha ustawień narodowych.

Poniższy kod odczytuje bieżący łańcuch ustawień narodowych i dodaje dwa francuskie ustawienia do początku łańcucha:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

Metoda `setLocaleChain()` wywołuje zdarzenie "change", gdy zakończy aktualizowanie łańcucha ustawień narodowych. Stała `air.Localizer.LOCALE_CHANGE` definiuje ciąg znaków "change". Zdarzenie ma jedną właściwość `localeChain` — jest to tablica kodów ustawień narodowych w nowym łańcuchu tych ustawień. Poniższy kod ustawia detektor dla tego zdarzenia:

```
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);
}
```

Statyczna właściwość `air.Localizer.ultimateFallbackLocale` reprezentuje ustawienia narodowe używane wówczas, gdy aplikacja nie obsługuje żadnych preferencji użytkownika. Wartością domyślną jest "en". Można również ustawić inne ustawienia narodowe, co prezentuje poniższy kod:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Pobieranie zasobów dla określonych ustawień narodowych.

Metoda `getString()` obiektu `Localizer` zwraca ciąg znaków zdefiniowany dla zasobów w określonych ustawieniach narodowych. Przy wywołaniu tej metody nie ma potrzeby określania wartości `locale`. W tym przypadku metoda sprawdza cały łańcuch ustawień narodowych i zwraca ciąg znaków pierwszego ustawienia narodowego, które udostępnią określoną nazwę zasobu. Ta metoda ma następujące parametry:

Parametr	Opis
<code>bundleName</code>	Pakunek, który zawiera zasób. Jest to nazwa pliku właściwości bez rozszerzenia <code>.properties</code> . (Na przykład: jeśli ten parametr ma wartość "alerts", kod obiektu <code>Localizer</code> wyszukuje pliki lokalizacji o nazwie <code>alerts.properties</code> .)
<code>resourceName</code>	Nazwa zasobu.
<code>templateArgs</code>	Opcjonalnie. Tablica ciągów znaków, która zastępuje numerowane znaczniki w ciągu znaków. Przykład: rozważmy wywołanie funkcji, w której parametr <code>templateArgs</code> ma wartość ["Raúl", "4"], a zgodny ciąg znaków zasobu to "Hello, {0}. You have {1} new messages.". W tym przypadku funkcja zwraca "Hello, Raúl. You have 4 new messages.". W celu zignorowania tego ustawienia należy przekazać wartość <code>null</code> .
<code>locale</code>	Opcjonalnie. Kod ustawień narodowych (np. "en", "en_us" lub "fr") do użycia. Jeśli ustawienia narodowe są określone, ale nie znaleziono zgodnej wartości, metoda nie kontynuuje wyszukiwania wartości w innych ustawieniach narodowych w łańcuchu. Jeśli nie określono kodu ustawień narodowych, funkcja zwraca ciąg znaków z pierwszego ustawienia narodowego z łańcucha, w którym dostępna jest określona nazwa zasobu.

Struktura lokalizowania może zaktualizować oznaczone atrybuty modelu DOM HTML. Jednak zlokalizowane ciągi znaków mogą być również używane na inne sposoby. Na przykład: ciąg znaków może być używany w niektórych dynamicznie wygenerowanych kodach HTML lub jako wartość parametru w wywołaniu funkcji. Przykład: poniższy kod wywołuje funkcję `alert()` z ciągiem znaków zdefiniowanym w zasobie `error114` w domyślnym pliku właściwości ustawień narodowych `fr_FR`:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

Metoda `getString()` wywołuje zdarzenie `resourceNotFound`, jeśli nie może znaleźć zasobu w określonym pakunku. Stała `air.Localizer.RESOURCE_NOT_FOUND` definiuje ciąg znaków `"resourceNotFound"`. Zdarzenie ma trzy właściwości: `bundleName`, `resourceName` i `locale`. Właściwość `bundleName` jest nazwą pakietu, w którym nie można znaleźć zasobu. Właściwość `resourceName` jest nazwą pakunku, w którym nie można znaleźć zasobu. Właściwość `locale` jest nazwą ustawień regionalnych, w których nie można znaleźć zasobu.

Metoda `getString()` wywołuje zdarzenie `bundleNotFound`, gdy nie może znaleźć określonego pakunku. Stała `air.Localizer.BUNDLE_NOT_FOUND` definiuje ciąg znaków `"bundleNotFound"`. Zdarzenie ma dwie właściwości: `bundleName` i `locale`. Właściwość `bundleName` jest nazwą pakunku, w którym nie można znaleźć zasobu. Właściwość `locale` jest nazwą ustawień narodowych, w których nie można znaleźć zasobu.

Metoda `getString()` działa asynchronicznie (i wywołuje zdarzenia `resourceNotFound` oraz `bundleNotFound` w sposób asynchroniczny). Poniższy kod ustawia detektory zdarzeń dla zdarzeń `resourceNotFound` i `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);
}
```

Metoda `getResourceBundle()` obiektu `Localizer` zwraca określony pakunek dla danego ustawienia narodowego. Wartość zwracana przez metodę jest obiektem z właściwościami zgodnymi z kluczami w pakunku. (Jeśli aplikacja nie może odnaleźć określonego pakunku, metoda zwraca wartość `null`).

Metoda przyjmuje dwa parametry — `locale` i `bundleName`.

Parametr	Opis
<code>locale</code>	Ustawienie narodowe (np. <code>"fr"</code>).
<code>bundleName</code>	Nazwa pakunku.

Na przykład poniższy kod wywołuje metodę `document.write()`, aby załadować domyślny pakunek dla ustawienia narodowego „fr”. Następnie wywoływana jest metoda `document.write()` w celu zapisania wartości kluczy `str1` i `str2` w tym pakunku:

```
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/>");
```

Metoda `getResourceBundle()` wywołuje zdarzenie `bundleNotFound`, gdy nie może znaleźć określonego pakunku. Stała `air.Localizer.BUNDLE_NOT_FOUND` definiuje ciąg znaków `"bundleNotFound"`. Zdarzenie ma dwie właściwości: `bundleName` i `locale`. Właściwość `bundleName` jest nazwą pakunku, w którym nie można znaleźć zasobu. Właściwość `locale` jest nazwą ustawień narodowych, w których nie można znaleźć zasobu.

Metoda `getFile()` obiektu `Localizer` zwraca zawartość pakunku w postaci ciągu znaków dla określonych ustawień narodowych. Plik pakunku jest odczytywany jako plik UTF-8. Metoda zawiera następujące parametry:

Parametr	Opis
resourceFileName	Nazwa pliku zasobu (np. "about.html").
templateArgs	Opcjonalnie. Tablica ciągów znaków, która zastępuje numerowane znaczniki w ciągu znaków. Przykład: rozważmy wywołanie funkcji, w której parametr <code>templateArgs</code> ma wartość ["Raúl", "4"], a zgodny plik zasobu zawiera dwie linie: <pre><html> <body>Hello, {0}. You have {1} new messages.</body> </html></pre> W tym przypadku funkcja zwraca ciąg znaków z dwoma liniami: <pre><html> <body>Hello, Raúl. You have 4 new messages. </body> </html></pre>
locale	Kod ustawień narodowych (np. "en_GB") do użycia. Jeśli ustawienia narodowe są określone, ale nie znaleziono zgodnego pliku, metoda nie kontynuuje wyszukiwania wartości w innych ustawieniach narodowych w łańcuchu. Jeśli nie określono kodu ustawień narodowych, funkcja zwraca tekst z pierwszego ustawienia narodowego w łańcuchu, który zawiera plik zgodny z wartością <code>resourceFileName</code> .

Na przykład: poniższy kod wywołuje metodę `document.write()` za pomocą treści pliku `about.html` ustawień narodowych `fr`:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

Metoda `getFile()` wywołuje zdarzenie `fileNotFound`, nawet jeśli nie może znaleźć zasobu w łańcuchu ustawień narodowych. Stała `air.Localizer.FILE_NOT_FOUND` definiuje ciąg znaków `"resourceNotFound"`. Metoda `getFile()` działa asynchronicznie (i wywołuje zdarzenie `fileNotFound` w sposób asynchroniczny). Zdarzenie ma dwie właściwości: `fileName` i `locale`. Właściwość `fileName` jest nazwą nieznanego pliku. Właściwość `locale` jest nazwą ustawień narodowych, w których nie można znaleźć zasobu. Poniższy kod ustawia detektor dla tego zdarzenia.

```
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);
}
```

Więcej tematów Pomocy

[Budowanie wielojęzycznej aplikacji opartej na kodzie HTML](#)

Rozdział 21: Zmienne środowiskowe ścieżek

Zestaw SDK środowiska AIR zawiera kilka programów, które można uruchamiać z poziomu wiersza polecenia lub terminala. Zazwyczaj uruchamianie tych programów jest łatwiejsze, gdy ścieżka do katalogu bin w zestawie SDK jest uwzględniona w zmiennej środowiskowej ścieżek.

Przedstawione tutaj informacje powinny być przydatne podczas ustawiania zmiennej ścieżek w systemach Windows, Mac i Linux. Konfiguracje komputerów mogą znacznie się różnić, dlatego ta procedura nie ma zastosowania w niektórych systemach. W takich przypadkach potrzebne informacje powinny być dostępne w dokumentacji systemu operacyjnego lub w Internecie.

Ustawianie zmiennej PATH w systemach Linux i Mac OS za pomocą powłoki Bash

Po wpisaniu polecenia w oknie terminala (powłoki) program odczytuje wpisane przez użytkownika informacje i próbuje odpowiednio zareagować. W tym celu musi najpierw zlokalizować w systemie plików program obsługujący polecenie. Powłoka szuka poleceń na liście katalogów przechowywanych w zmiennej środowiskowej ścieżek (zmiennej \$PATH). Aby wyświetlić bieżącą zawartość tej zmiennej, należy wpisać następujące polecenie.

```
echo $PATH
```

Spowoduje to zwrócenie rozdzielanej dwukropkami listy katalogów, która powinna być podobna do poniższej.

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

Należy dodać ścieżkę do katalogu bin w zestawie SDK środowiska AIR, tak aby powłoka mogła znaleźć narzędzia ADT i ADL. Jeśli zestaw SDK środowiska AIR został umieszczony w katalogu `/Users/fred/SDKs/AIR`, następujące polecenie spowoduje dodanie wymaganych katalogów do zmiennej PATH.

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Uwaga: Jeśli ścieżka zawiera spacje, należy poprzedzić je znakami ukośnika odwrotnego, tak jak pokazano poniżej.

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

Aby upewnić się, że operacja została wykonana pomyślnie, można użyć polecenia `echo`.

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Jak dotąd wszystko jest w porządku. Teraz można wpisać poniższe polecenie, co powinno spowodować wyświetlenie odpowiedniego komunikatu.

```
adt -version
```

Jeśli zmienna \$PATH została poprawnie zmodyfikowana, polecenie powinno wyświetlić wersję programu ADT.

Istnieje jednak jeszcze pewien problem. Po następnym otwarciu nowego okna terminala zmienna ścieżek nie będzie już zawierać nowych pozycji. Polecenie ustawiające zmienną ścieżek musi być uruchamiane za każdym razem po uruchomieniu nowego terminala.

Typowym rozwiązaniem tego problemu jest dodanie tego polecenia do jednego ze skryptów uruchamiania używanych przez powłokę. W systemie Mac OS można utworzyć plik o nazwie `.bash_profile` w katalogu `~/nazwa_użytkownika`. Będzie on uruchamiany za każdym razem po otwarciu nowego okna terminala. W systemie Ubuntu skrypt uruchamiany podczas otwierania nowego okna terminala ma nazwę `.bashrc`. W innych dystrybucjach systemu Linux i programach powłoki obowiązują podobne konwencje.

Aby dodać odpowiednie polecenie do skryptu uruchamiania powłoki:

- 1 Przejdź do swojego katalogu osobistego.

```
cd
```

- 2 Utwórz profil konfiguracji powłoki (w razie potrzeby) i przekieruj wpisywany tekst na koniec pliku, używając polecenia `cat >>`. Użyj pliku odpowiedniego dla systemu operacyjnego i powłoki, z których korzystasz. Można na przykład użyć pliku `.bash_profile` w systemie Mac OS lub pliku `.bashrc` w systemie Ubuntu.

```
cat >> .bash_profile
```

- 3 Wpisz tekst, który ma zostać dodany do pliku.

```
export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
```

- 4 Zakończ przekierowywanie tekstu, naciskając klawisze `CTRL+SHIFT+D` na klawiaturze.

- 5 Wyświetl plik w celu sprawdzenia, czy wszystko jest w porządku.

```
cat .bash_profile
```

- 6 Otwórz nowe okno terminala, aby sprawdzić zmienną ścieżek.

```
echo $PATH
```

Powinny zostać wyświetlone informacje dodane do tej zmiennej.

Jeśli w późniejszym czasie zostanie utworzona nowa wersja jednego z pakietów SDK w innym katalogu, należy zaktualizować polecenie dotyczące zmiennej ścieżek w pliku konfiguracyjnym. W przeciwnym razie powłoka będzie nadal używać starej wersji.

Ustawianie zmiennej PATH w systemie Windows

W systemie Windows otwierane okna wiersza polecenia dziedziczą globalne zmienne środowiskowe zdefiniowane we właściwościach systemu. Jedną z istotnych zmiennych jest zmienna `PATH` (zmienna ścieżek) określająca listę katalogów, które przeszukuje program wykonujący polecenia, gdy użytkownik wpisuje nazwę programu do uruchomienia. Aby wyświetlić bieżącą zawartość zmiennej ścieżek podczas korzystania z okna wiersza polecenia, można wpisać poniższe polecenie.

```
set path
```

Spowoduje to wyświetlenie rozdzielanej średnikami listy katalogów podobnej do poniższej.

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

Należy dodać ścieżkę do katalogu `bin` w zestawie SDK środowiska AIR, tak aby program wykonujący polecenia mógł znaleźć narzędzia ADT i ADL. Jeśli zestaw SDK środowiska AIR został umieszczony w katalogu `C:\SDKs\AIR`, można dodać odpowiednią pozycję do zmiennej ścieżek, wykonując następującą procedurę:

- 1 Otwórz okno dialogowe Właściwości systemu w Panelu sterowania lub kliknij prawym przyciskiem myszy ikonę Mój komputer i wybierz z menu polecenie Właściwości.
- 2 Na karcie Zaawansowane kliknij przycisk Zmienne środowiskowe.

3 W oknie dialogowym Zmienne środowiskowe zaznacz pozycję PATH w sekcji Zmienne systemowe.

4 Kliknij przycisk Edytuj.

5 Przewiń tekst w polu Wartość zmiennej, tak aby wyświetlić jego końcową część.

6 Wprowadź następujący tekst na końcu bieżącej wartości:

```
;C:\SDKs\AIR\bin
```

7 Kliknij przycisk OK we wszystkich oknach dialogowych, aby zapisać zmienną ścieżek.

Jeśli są otwarte jakiegokolwiek okna wiersza polecenia, ich środowiska nie są aktualizowane. Otwórz nowe okno wiersza polecenia i wpisz poniższe polecenie, aby upewnić się, że ścieżki są poprawnie skonfigurowane.

```
adt -version
```

Jeśli w późniejszym czasie zostanie zmieniona lokalizacja zestawu SDK środowiska AIR lub dodana nowa wersja, należy zaktualizować zmienną ścieżek.