

Разработка приложений ADOBE® AIR®

Юридическая информация

Юридическую информацию см. на веб-странице http://help.adobe.com/ru_RU/legalnotices/index.html.

Содержание

Глава 1. Об Adobe AIR

Глава 2. Установка Adobe AIR

Установка Adobe AIR	3
Удаление Adobe AIR	5
Установка и выполнение образцов приложений AIR	5
Обновления Adobe AIR	6

Глава 3. Работа с API-интерфейсами для AIR

Классы ActionScript 3.0, характерные для AIR	7
Классы проигрывателя Flash Player с функциями, характерными для Adobe AIR	12
Компоненты Flex, характерные для Adobe AIR	15

Глава 4. Инструменты Adobe Flash Platform для разработки приложений AIR

Установка AIR SDK	17
Настройка Flex SDK	19
Настройка внешних SDK	20

Глава 5. Создание первого приложения AIR

Создание первого настольного приложения Flex AIR с помощью Flash Builder	21
Создание первого настольного приложения AIR с помощью Flash Professional	24
Создание первого приложения AIR for Android с помощью Flash Professional	26
Создание первого приложения AIR для iOS	27
Создание первого HTML-приложения AIR с помощью Dreamweaver	31
Создание первого HTML-приложения AIR с помощью комплекта AIR SDK	34
Создание первого настольного приложения AIR с использованием пакета Flex SDK	39
Создание первого приложения AIR для Android с использованием пакета Flex SDK	43

Глава 6. Разработка приложений AIR для настольных систем

Процедура разработки приложений AIR для настольных систем	47
Настройка свойств приложения для настольных систем	48
Отладка приложения AIR для настольной системы	54
Упаковка файла установки AIR для настольных систем	55
Упаковка собственного установщика для настольной системы	59
Упаковка связанных пакетов среды выполнения для настольных компьютеров	63
Распространение пакетов AIR для настольных систем	65

Глава 7. Разработка приложений AIR для мобильных устройств

Настройка среды разработки	68
Принципы разработки мобильных приложений	69
Процедура создания приложений AIR для мобильных устройств	73
Настройка свойств мобильного приложения	74
Упаковка приложения AIR для мобильных устройств	99
Отладка приложений AIR для мобильных устройств	106

Установка AIR и приложений AIR на мобильные устройства	115
Обновление мобильных приложений AIR	118
Использование push-уведомлений	119
Глава 8. Разработка приложений AIR для телевизионных устройств	
Возможности AIR для телевизионных устройств	128
Принципы разработки приложений AIR for TV	130
Процедура разработки приложений AIR for TV	146
Свойства дескриптора приложений AIR for TV	148
Упаковка приложений AIR for TV	152
Отладка приложений AIR for TV	154
Глава 9. Использование собственных расширений для Adobe AIR	
Файлы собственных расширений AIR (ANE)	158
Сравнение собственных расширений с классами ActionScript NativeProcess	159
Сравнение собственных расширений с библиотеками классов ActionScript (SWC-файлы)	159
Поддерживаемые устройства	160
Поддерживаемые профили устройств	160
Список задач для использования собственного расширения	161
Объявление расширения в файле дескриптора приложения	161
Включение ANE-файла в путь к библиотеке приложения	161
Создание пакета приложения, в котором используются собственные расширения	162
Глава 10. Компиляторы ActionScript	
Сведения об инструментах командной строки AIR в пакете Flex SDK	165
Настройка компилятора	165
Компиляция исходных файлов MXML и ActionScript для AIR	166
Компиляция компонента AIR или библиотеки кодов (Flex)	168
Глава 11. AIR Debug Launcher (ADL)	
Использование ADL	170
Примеры ADL	173
Завершение работы ADL и коды ошибок	174
Глава 12. AIR Developer Tool (ADT)	
Команды ADT	176
Наборы параметров ADT	191
Сообщения об ошибках ADT	196
Переменные среды ADT	201
Глава 13. Подписание приложений AIR	
Цифровая подпись файлов AIR	203
Создание неподписанного промежуточного файла AIR с помощью ADT	213
Подписание промежуточного файла AIR с помощью ADT	213
Подписание обновленной версии приложения AIR	213
Создание самозаверяющего сертификата с помощью ADT	218

Глава 14. Файлы дескриптора приложения AIR

Изменения в дескрипторе приложения	221
Структура файла дескриптора приложения	223
Элементы дескриптора приложения AIR	224

Глава 15. Профили устройств

Ограничение целевых профилей в файле дескриптора приложения	264
Возможности различных профилей	265

Глава 16. AIR.SWF, встроенный в браузер интерфейс API

Настройка файла непрерывной установки badge.swf	268
Установка приложения AIR с помощью файла badge.swf	268
Загрузка файла air.swf	272
Проверка наличия установленной среды выполнения	272
Проверка наличия установленного AIR с веб-страницы	273
Установка приложений AIR из обозревателя	274
Запуск установленных приложений AIR из обозревателя	275

Глава 17. Обновление приложений AIR

Об обновлении приложений	278
Использование настраиваемого пользовательского интерфейса обновления приложения	280
Загрузка файла AIR на компьютер пользователя	280
Проверка факта первичного запуска приложения	282
Использование инфраструктуры обновления	284

Глава 18. Просмотр исходного кода

Загрузка, настройка и открытие объекта просмотра исходного кода	299
Интерфейс пользователя для объекта просмотра исходного кода	302

Глава 19. Отладка с помощью AIR HTML Introspector

О программе AIR Introspector	304
Загрузка кода AIR Introspector	304
Проверка объекта на вкладке Console (Консоль)	305
Настройка AIR Introspector	307
Интерфейс AIR Introspector	307
Использование AIR Introspector с содержимым, находящимся вне изолированной программной среды приложения	314

Глава 20. Локализация приложений AIR

Локализация названия и описания приложения в программе установки приложения AIR	316
Локализация HTML-содержимого с помощью инфраструктуры локализации AIR HTML	317

Глава 21. Переменные среды Path

Настройка переменной среды PATH в Linux и Mac OS с использованием оболочки Bash	327
Настройка переменной среды PATH в ОС Windows	328

Глава 1. Об Adobe AIR

Adobe® AIR® — это поддерживающая множество платформ и несколько экранов среда выполнения, позволяющая максимально эффективно использовать возможности разработки для сборки и развертывания мультимедийных интернет-приложений (RIA) на мобильных устройствах. Приложения AIR для настольных компьютеров, телевизионных и мобильных устройств можно создавать с использованием ActionScript 3.0 в Adobe® Flex и Adobe® Flash® (на основе SWF). Приложения AIR для настольных компьютеров также можно создавать с использованием HTML, JavaScript® и Ajax (на основе HTML).

Дополнительные сведения о том, как начать работать с Adobe AIR и как использовать этот продукт, см. на сайте Adobe AIR Developer Connection (<http://www.adobe.com/devnet/air/>).

AIR позволяет работать в знакомых средах разработки, используя наиболее удобные инструменты и методы. Благодаря поддержке Flash, Flex, HTML, JavaScript и Ajax можно создавать оптимальные условия работы, соответствующие конкретным потребностям.

В частности, при разработке приложений можно пользоваться одной или несколькими из приведенных ниже технологий:

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

С точки зрения пользователя, приложения AIR выглядят точно так же, как собственные приложения. Среда выполнения устанавливается на компьютер или устройство пользователя только один раз, после чего приложения AIR устанавливаются и используются, как любые другие программы (В iOS отдельная среда выполнения AIR не устанавливается. Приложения AIR for iOS являются автономными.)

Среда выполнения предлагает надежную платформу, совместимую с разными операционными системами, и инфраструктуру для разработки приложений. Таким образом, благодаря проверенной функциональности и взаимодействию с разными настольными компьютерами, она избавляет вас от необходимости тестирования приложений в множестве разных обозревателей. Вы разрабатываете приложение не под конкретную операционную систему, а для среды выполнения, что имеет ряд очевидных преимуществ:

- Приложения, разработанные для AIR, функционируют во многих операционных системах без дополнительных доработок с вашей стороны. Среда выполнения обеспечивает предсказуемое и надежное представление содержимого и взаимодействие программы с пользователем во всех операционных системах, которые поддерживает AIR.
- Можно ускорить создание приложений благодаря использованию существующих веб-технологий и шаблонов проектирования. Можно переносить веб-приложения на настольные компьютеры, не изучая традиционные технологии разработки для настольных систем или их сложные собственные коды.
- Такой способ разработки проще, чем разработка на низкоуровневых языках типа C и C++. Вам не придется сталкиваться со сложными низкоуровневыми API-интерфейсами каждой из операционных систем.

При разработке приложений для AIR вам доступен богатый выбор инфраструктур и API-интерфейсов:

- API-интерфейсы AIR, предоставляемые средой выполнения и инфраструктурой AIR;
- API-интерфейсы ActionScript, используемые в SWF-файлах и инфраструктурах Flex (и других библиотеках и инфраструктурах на основе ActionScript);
- языки HTML, CSS и JavaScript;
- большинство инфраструктур Ajax

- Собственные расширения для Adobe AIR предоставляют API-интерфейсы ActionScript, с помощью которых можно получать доступ к функциям платформы, запрограммированным с использованием собственного кода. Собственные расширения могут также предоставлять доступ к существующему собственному коду и собственному коду, обеспечивающему более высокую производительность.

AIR кардинальным образом меняет процесс создания, развертывания и использования приложений. Вы получаете дополнительные средства управления для расширения приложений на основе Flash, Flex, HTML и Ajax для настольных компьютеров, мобильных и телевизионных устройств.

Сведения о том, что включено в обновление AIR см. в примечаниях к выпуску Adobe AIR по адресу (http://www.adobe.com/go/learn_air_relnotes_ru).

Глава 2. Установка Adobe AIR

Среда выполнения Adobe® AIR® позволяет выполнять приложения AIR. Его можно установить следующими способами:

- установить только среду выполнения (не устанавливать приложение AIR)
- в первый раз установить приложение AIR с использованием «значка» установки с веб-страницы (также появляется запрос на установку среды выполнения);
- создать пользовательский установщик, устанавливающий как приложение, так и среду выполнения. Необходимо получить разрешение Adobe на распространение среды выполнения AIR таким способом. Разрешение можно запросить на странице [лицензирования среды выполнения Adobe](#). Обратите внимание, что компания Adobe не предоставляет инструменты для создания подобного установщика. Однако доступно множество сторонних наборов инструментов для создания установщиков.
- Установить приложение AIR со связанной средой выполнения AIR. Связанная среда выполнения используется только путем связывания приложения. Она не используется для выполнения других приложений AIR. Возможность связывания среды выполнения доступна на платформах Mac и Windows. В ОС iOS все приложения включают связанную среду выполнения. В AIR 3.7 все программы для Android начали по умолчанию поддерживать связанную среду выполнения (но предусмотрена возможность использования отдельной среды выполнения).
- установить в качестве среды разработки AIR комплект AIR SDK, Adobe® Flex® Builder™ или комплект Adobe Flex® SDK (включающий инструменты разработки командной строки AIR). Среда выполнения, включенная в SDK, используется только при отладке приложений: она не применяется для выполнения установленных приложений AIR.

Системные требования для установки Adobe AIR и выполнения приложений AIR подробно описаны здесь: [Adobe AIR: системные требования \(http://www.adobe.com/ru/products/air/systemreqs/\)](http://www.adobe.com/ru/products/air/systemreqs/).

Файлы журналов создаются установщиками среды выполнения и приложения AIR при установке, обновлении или удалении приложений AIR или среды выполнения AIR. Журналы установок позволяют определить причины проблем при установке или обновлении. См. статью [Журналы установки](#).

Установка Adobe AIR

Для установки или обновления среды выполнения на компьютере необходимы права администратора.

Установка среды выполнения на компьютер Windows

- 1 Загрузите файл установки среды выполнения с веб-страницы <http://get.adobe.com/air>.
- 2 Дважды щелкните его.
- 3 Следуйте инструкциям в окне установки для ее выполнения.

Установка среды выполнения на компьютер Mac

- 1 Загрузите файл установки среды выполнения с веб-страницы <http://get.adobe.com/air>.
- 2 Дважды щелкните его.
- 3 Следуйте инструкциям в окне установки для ее выполнения.

- 4 Если установщик отображает окно идентификации, введите свое имя пользователя в системе Mac OS и пароль.

Установка среды выполнения на компьютере с Linux

Примечание. В настоящее время среда выполнения AIR 2.7 и более поздних версий не поддерживается на платформе Linux. Приложения AIR, разворачиваемые в ОС Linux, должны по-прежнему использовать AIR 2.6 SDK.

Использование двоичного установщика

- 1 Найдите и загрузите двоичный установщик на странице http://kb2.adobe.com/cps/853/cpsid_85304.html.
- 2 Установите права на доступ к файлам таким образом, чтобы мог выполняться установщик приложения. С помощью командной строки можно установить разрешения для доступа к файлам:

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

Некоторые версии Linux позволяют установить права доступа к файлам с помощью диалогового окна «Свойства» (Properties), открываемого из контекстного меню.

- 3 Запустите программу установки из командной строки или двойным щелчком выполняемого установочного файла.
- 4 Следуйте инструкциям в окне установки для ее выполнения.

Adobe AIR устанавливается как собственный пакет. Другими словами, как rpm при распределении на базе rpm и как deb при распределении Debian. В настоящее время AIR не поддерживает другие форматы пакета.

Использование пакетных установщиков

- 1 Найдите и загрузите пакетный файл AIR на странице http://kb2.adobe.com/cps/853/cpsid_85304.html. Загрузите пакет rpm или Debian в зависимости от того, какой формат пакета поддерживает система.
- 2 При необходимости дважды щелкните файл пакета AIR, чтобы установить его.

Установку можно запустить с помощью командной строки:

- a В системе Debian:

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

- b В системе на базе rpm:

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

А если выполняется обновление существующей версии (AIR 1.5.3 или более поздней):

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

Для установки приложений AIR 2 и AIR на компьютере необходимы права администратора.

Adobe AIR устанавливается в папку /opt/Adobe AIR/Versions/1.0.

AIR регистрирует MIME-тип application/vnd.adobe.air-application-installer-package+zip, то есть файлы .air относятся к этому MIME-типу и поэтому регистрируются в среде выполнения AIR.

Установка среды выполнения на устройства Android

Последнюю рабочую версию среды выполнения AIR можно установить с Android Маркета.

Версии среды выполнения AIR для разработчиков можно установить по ссылке на веб-странице или с помощью команды ADT -installRuntime. Одновременно может быть установлена только одна версия среды выполнения AIR. Установка рабочей версии вместе с версией для разработчиков невозможна.

Дополнительные сведения см. в разделе «[Команда ADT installRuntime](#)» на странице 189».

Установка среды выполнения на устройства iOS

Необходимый код среды выполнения AIR включается в пакет приложения, создаваемый для устройств iPhone, iPod touch и iPad. Устанавливать среду выполнения в виде отдельного компонента не требуется.

Дополнительные разделы справки

«AIR for iOS» на странице 74

Удаление Adobe AIR

Установленную на компьютер среду выполнения можно удалить. Ниже описано, как это сделать.

Удаление среды выполнения с компьютера под управлением Windows

- 1 В меню «Пуск» выберите «Настройки» > «Панель управления».
- 2 Откройте меню «Программы», «Программы и компоненты» или «Установка и удаление программ» (в зависимости от установленной версии Windows).
- 3 Выберите «Adobe AIR», чтобы удалить среду выполнения.
- 4 Нажмите кнопку «Изменить/Удалить».

Удаление среды выполнения с компьютера под управлением Mac OS

- Дважды щелкните «Adobe AIR Uninstaller» в папке /Applications/Utilities.

Удаление среды выполнения с компьютера под управлением Linux

Выполните одно из следующих действий:

- Выберите команду «Adobe AIR Uninstaller» из меню «Приложения» (Applications).
- Запустите двоичный файл программы установки AIR с параметром `-uninstall`
- Удалите пакеты AIR (`adobeair` и `adobecerts`) с помощью диспетчера пакетов.

Удаление среды выполнения с устройства Android

- 1 Откройте приложение «Настройки» на устройстве.
- 2 Нажмите на элемент Adobe AIR в разделе «Приложения» > «Управление приложениями».
- 3 Нажмите кнопку «Удалить».

Также можно использовать команду `ADT -uninstallRuntime`. Дополнительные сведения см. в разделе ««Команда `ADT uninstallRuntime`» на странице 190».

Удаление связанной среды выполнения

Для удаления связанной среды выполнения необходимо удалить приложение, с которым установлена среда. Обратите внимание, что связанные среды выполнения используются только для выполнения установки приложения.

Установка и выполнение образцов приложений AIR

Для установки или обновления приложения AIR на компьютере необходимы права администратора.

Вы можете ознакомиться с образцами некоторых приложений AIR и их характеристиками. Ниже описано, как их загрузить и установить:

- 1 Загрузите и запустите [образцы приложений AIR](#). Доступны как скомпилированные файлы, так и исходный код.
- 2 Для загрузки и запуска образца приложения щелкните по кнопке «Установить» рядом с ним. Вам будет предложено установить и запустить приложение.
- 3 Если вы решите загрузить образцы приложений, но запустить их позднее, выберите только ссылки для загрузки. Приложения AIR можно запустить в любое время.
 - В Windows дважды щелкните значок приложения на рабочем столе или выберите его в меню «Пуск».
 - В Mac OS дважды щелкните по значку приложения, по умолчанию установленного в папку Applications в каталоге пользователя (например, Macintosh HD/Пользователи/Иван/Applications/).

Примечание. Проверьте, не обновлялись ли эти инструкции, в примечаниях к выпуску:
http://www.adobe.com/go/learn_air_relnotes_ru.

Обновления Adobe AIR

Время от времени компания Adobe выпускает обновления Adobe AIR, содержащие новые функции или исправления незначительных проблем. С помощью функции автоматического уведомления и обновления компания Adobe автоматически уведомляет пользователей о доступности обновленной версии Adobe AIR.

Обновления Adobe AIR обеспечивают правильную работу Adobe AIR и часто содержат важные изменения системы защиты. Компания Adobe рекомендует выполнять обновление до последней версии Adobe AIR при появлении новой версии, особенно если упоминается обновление системы защиты.

По умолчанию при запуске приложения AIR среда выполнения проверяет доступность обновления. Она выполняет эту проверку, если прошло более двух недель с момента последней проверки наличия обновлений. Если обновление доступно, среда AIR загружает обновление в фоновом режиме.

Пользователи могут отключить возможность автоматического обновления с помощью приложения AIR SettingsManager. Приложение AIR SettingsManager можно загрузить с веб-страницы <http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air>.

Стандартный процесс установки Adobe AIR включает соединение с веб-сайтом <http://airinstall.adobe.com> для отправки основной информации о среде установки, такой как версия и язык операционной системы. Эта информация передается только один раз при каждой установке и позволяет компании Adobe подтвердить успешное выполнение установки. Персональные данные не собираются и не передаются.

Обновление связанных сред выполнения

Если приложение распространяется вместе со связанной средой выполнения, связанная среда выполнения не обновляется автоматически. Для обеспечения безопасности пользователей необходимо следить за обновлениями, публикуемыми компанией Adobe, и обновлять приложение новой версией среды выполнения при публикации соответствующего изменения системы защиты.

Глава 3. Работа с API-интерфейсами для AIR

Adobe® AIR® включает функции, недоступные в SWF-содержимом, выполняемом в проигрывателе Adobe® Flash® Player.

Разработчикам ActionScript 3.0

API-интерфейсы Adobe AIR подробно описаны в следующих двух книгах:

- [Руководство разработчика по ActionScript 3.0](#)
- [Справочник ActionScript® 3.0 для платформы Adobe® Flash® Platform](#)

Разработчикам HTML

Для разработчиков HTML-приложений AIR API-интерфейсы, предоставляемые в JavaScript посредством файла AIRAliases.js (см. документ [Доступ к классам API-интерфейсов AIR через JavaScript](#)), описаны в следующих двух книгах:

- [Руководство разработчика HTML для Adobe AIR](#)
- [Справочник по API-интерфейсу Adobe AIR для разработчиков HTML](#)

Классы ActionScript 3.0, характерные для AIR

В следующей таблице перечислены классы времени выполнения, которые характерны для Adobe AIR. Они недоступны для SWF-содержимого, выполняемого в Adobe® Flash® Player в браузере.

Разработчикам HTML

Классы, доступные в JavaScript посредством файла AIRAliases.js, перечислены в документе [Справочник по API-интерфейсам Adobe AIR для разработчиков HTML](#).

Класс	Пакет ActionScript 3.0	Добавлено в версии AIR
ARecord	flash.net.dns	2.0
AAAARecord	flash.net.dns	2.0
ApplicationUpdater	air.update	1.5
ApplicationUpdaterUI	air.update	1.5
AudioPlaybackMode	flash.media	3.0
AutoCapitalize	flash.text	3.0
BrowserInvokeEvent	flash.events	1.0
CameraPosition	flash.media	3.0
CameraRoll	flash.media	2.0

Класс	Пакет ActionScript 3.0	Добавлено в версии AIR
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 и более ранних версий; отсутствует, начиная с версии 3.7
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 и более ранних версий; отсутствует, начиная с версии 3.7
GameInputHand	flash.ui	3.6 и более ранних версий; отсутствует, начиная с версии 3.7
Geolocation	flash.sensors	2.0
GeolocationEvent	flash.events	2.0

Класс	Пакет ActionScript 3.0	Добавлено в версии AIR
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
Interactivelcon	flash.desktop	1.0
InterfaceAddress	flash.net	2.0
InvokeEvent	flash.events	1.0
InvokeEventReason	flash.desktop	1.5.1
IPVersion	flash.net	2.0
IURIDereferencer	flash.security	1.0
LocationChangeEvent	flash.events	2.5
MediaEvent	flash.events	2.5
MediaPromise	flash.media	2.5
MediaType	flash.media	2.5
MXRecord	flash.net.dns	2.0
NativeApplication	flash.desktop	1.0
NativeDragActions	flash.desktop	1.0
NativeDragEvent	flash.events	1.0
NativeDragManager	flash.desktop	1.0
NativeDragOptions	flash.desktop	1.0
NativeMenu	flash.display	1.0
NativeMenuItem	flash.display	1.0
NativeProcess	flash.desktop	2.0
NativeProcessExitEvent	flash.events	2.0
NativeProcessStartupInfo	flash.desktop	2.0
NativeWindow	flash.display	1.0
NativeWindowBoundsEvent	flash.events	1.0
NativeWindowDisplayState	flash.display	1.0

Класс	Пакет ActionScript 3.0	Добавлено в версии AIR
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
SQLExceptionEvent	flash.events	1.0

Класс	Пакет ActionScript 3.0	Добавлено в версии AIR
SQLErrorOperation	flash.errors	1.0
SQLEvent	flash.events	1.0
SQLIndexSchema	flash.data	1.0
SQLMode	flash.data	1.0
SQLResult	flash.data	1.0
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

Класс	Пакет ActionScript 3.0	Добавлено в версии AIR
URLMonitor	air.net	1.0
URLRequestDefaults	flash.net	1.0
XMLSignatureValidator	flash.security	1.0

Классы проигрывателя Flash Player с функциями, характерными для Adobe AIR

Ниже перечислены классы, которые могут использоваться при воспроизведении в обозревателе SWF-содержимого и для которых AIR предлагает дополнительные методы и свойства:

Пакет	Класс	Свойство, метод или событие	Добавлено в версии 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	
		событие orientationChange	2.0	
		событие orientationChanging	2.0	
		setAspectRatio	2.0	
		setOrientation	2.0	
		softKeyboardRect	2.6	
		supportedOrientations	2.6	
	supportsOrientationChange	2.0		
	NativeWindow	владелец	2.6	
		listOwnedWindows	2.6	
	NativeWindowInitOptions	владелец	2.6	

Пакет	Класс	Свойство, метод или событие	Добавлено в версии 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

Пакет	Класс	Свойство, метод или событие	Добавлено в версии AIR
flash.net	FileReference	extension	1.0
		событие httpResponseStatus	1.0
		uploadUnencoded()	1.0
	NetStream	событие drmAuthenticate	1.0
		событие 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

Пакет	Класс	Свойство, метод или событие	Добавлено в версии 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
		printMethod	2.0
flash.system	Capabilities	languages	1.1
	LoaderContext	allowLoadBytesCodeExecution	1.0
	Security	APPLICATION	1.0
flash.ui	KeyLocation	D_PAD	2.5

Большинство перечисленных новых свойств и методов доступно только для содержимого в изолированной программной среде AIR. Тем не менее новые члены классов URLRequest также доступны для воспроизведения содержимого в других изолированных программных средах.

Методы `ByteArray.compress()` и `ByteArray.uncompress()` имеют новый параметр `algorithm`, позволяющий выбрать метод сжатия `deflate` или `zlib`. Этот параметр доступен только для содержимого, воспроизводимого в Adobe AIR.

Компоненты Flex, характерные для Adobe AIR

Следующие компоненты Adobe® Flex™ MX доступны при разработке содержимого для Adobe AIR:

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

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

Помимо этого, Flex 4 включает следующие компоненты Spark среды AIR:

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

Дополнительные сведения о компонентах AIR Flex см. в руководстве [Использование компонентов Flex AIR](#).

Глава 4. Инструменты Adobe Flash Platform для разработки приложений AIR

Приложения AIR можно создавать с помощью следующих инструментов платформы Adobe Flash Platform.

Для разработчиков ActionScript 3.0 (Flash и Flex):

- Adobe Flash Professional (см. веб-страницу [«Публикация для AIR»](#))
- Комплекты SDK Adobe Flex 3.x и 4.x (см. разделы [«Настройка Flex SDK»](#) на странице 19 и [«AIR Developer Tool \(ADT\)»](#) на странице 176»)
- Adobe Flash Builder (см. веб-страницу [«Разработка приложений AIR в программе Flash Builder»](#))

Для разработчиков HTML и Ajax:

- SDK Adobe AIR (см. разделы [«Установка AIR SDK»](#) на странице 17 и [«AIR Developer Tool \(ADT\)»](#) на странице 176»)
- Adobe Dreamweaver CS3, CS4, CS5 (см. раздел [«Расширение AIR для Dreamweaver»](#))

Установка AIR SDK

В состав SDK Adobe AIR входят следующие инструменты командной строки для запуска и упаковки приложений.

AIR Debug Launcher (ADL) Позволяет запускать приложения AIR, не устанавливая их. См. раздел [«AIR Debug Launcher \(ADL\)»](#) на странице 170».

AIR Development Tool (ADT) Предназначен для упаковки приложений AIR в развертываемые установочные пакеты. См. раздел [«AIR Developer Tool \(ADT\)»](#) на странице 176».

Для работы инструментов командной строки AIR требуется Java. Можно использовать виртуальную машину Java из комплекта JRE или JDK (версии 1.5 или более поздней версии). Java JRE и Java JDK можно загрузить с веб-сайта <http://java.sun.com/>.

Для запуска инструмента ADT требуется не менее 2 ГБ памяти на компьютере.

Примечание. Для запуска приложений AIR конечным пользователям устанавливать Java не требуется.

Краткий обзор создания приложения AIR с помощью AIR SDK см. в разделе [«Создание первого HTML-приложения AIR с помощью комплекта AIR SDK»](#) на странице 34».

Загрузка и установка AIR SDK

Ниже описано, как загрузить и установить AIR SDK:

Установка AIR SDK в ОС Windows

- Загрузите установочный файл AIR SDK.

- AIR SDK распространяется в виде стандартного файла архива. Чтобы установить AIR, извлеките содержимое SDK в папку на компьютере (например, в C:\Program Files\Adobe\AIRSDK или C:\AIRSDK).
- Инструменты ADL и ADT содержатся в папке bin комплекта AIR SDK; добавьте этот путь в переменную среды PATH.

Установка AIR SDK в ОС Mac OS X

- Загрузите установочный файл AIR SDK.
- AIR SDK распространяется в виде стандартного файла архива. Чтобы установить AIR, извлеките содержимое SDK в папку на компьютере (например, в /Users/<имя_пользователя>/Applications/AIRSDK).
- Инструменты ADL и ADT содержатся в папке bin комплекта AIR SDK; добавьте этот путь в переменную среды PATH.

Установка AIR SDK в ОС Linux

- Пакет SDK доступен в формате tbz2.
- Чтобы установить SDK, создайте папку и распакуйте в нее содержимое SDK, используя команду `tar -jxvf <путь к файлу AIR-SDK.tbz2>`

Сведения о начале работы с инструментами AIR SDK см. в разделе «Создание приложения AIR с помощью инструментов командной строки».

Состав пакета AIR SDK

В таблице ниже приводится описание файлов пакета AIR SDK:

Папка SDK	Описание файлов/инструментов
bin	AIR Debug Launcher (ADL) позволяет запустить приложение без предварительной упаковки и установки. Дополнительные сведения см. в разделе « AIR Debug Launcher (ADL) » на странице 170. AIR Developer Tool (ADT) упаковывает приложение в AIR-файл для распространения. Дополнительные сведения об использовании инструмента см. в разделе « AIR Developer Tool (ADT) » на странице 176.
frameworks	Каталог libs содержит библиотеки кодов, используемые в приложениях AIR. Каталог projects содержит код для скомпилированных библиотек SWF и SWC.
include	Включенный каталог содержит файл заголовка на языке C для написания собственных расширений.
install	Каталог install содержит USB-драйверы Windows для устройств Android (эти драйверы Google предоставляет в составе пакета Android SDK).
lib	Содержит код поддержки для инструментов AIR SDK.

Папка SDK	Описание файлов/инструментов
runtimes	Среда выполнения AIR для настольных компьютеров и мобильных устройств. ADL использует среду выполнения на настольном компьютере для запуска приложений AIR до их распаковки или установки. Среду выполнения AIR для Android (пакеты APK) можно устанавливать на устройства Android и в эмуляторы для разработки и тестирования. Отдельные пакеты APK используются для устройств и эмуляторов (общедоступную версию среды выполнения AIR for Android можно загрузить с Android Маркета).
samples	В этой папке содержится пример файла дескриптора приложения, пример функции незаметной установки (badge.swf), а также значки приложения AIR по умолчанию.
templates	descriptor-template.xml — шаблон файла дескриптора приложения, необходимого для каждого приложения AIR. Подробное описание файла дескриптора приложения см. в разделе « Файлы дескриптора приложения AIR » на странице 220. В этом каталоге также расположены файлы схемы для XML-структуры дескриптора приложения для каждой рабочей версии AIR.

Настройка Flex SDK

Создать приложение Adobe® AIR® с помощью Adobe® Flex™ можно одним из следующих способов.

- Можно загрузить и установить пакет Adobe® Flash® Builder™, в котором содержатся инструменты для создания проектов Adobe AIR, их проверки, отладки и упаковки приложений AIR. См. раздел «[Создание первого настольного приложения Flex AIR с помощью Flash Builder](#)» на странице 21.
- Можно загрузить Adobe® Flex™ SDK и разрабатывать приложения Flex AIR с помощью привычного текстового редактора и инструментов командной строки.

Краткий обзор создания приложения AIR с помощью Flex SDK см. в разделе «[Создание первого настольного приложения AIR с использованием пакета Flex SDK](#)» на странице 39».

Установка Flex SDK

Для создания приложений AIR с помощью инструментов командной строки необходимо, чтобы на компьютере была установлена Java. Можно использовать виртуальную машину Java из комплекта JRE или JDK (версии 1.5 или более поздней версии). Java JRE и Java JDK можно загрузить с веб-сайта <http://java.sun.com/>.

Примечание. Для запуска приложений AIR конечным пользователям устанавливать Java не требуется.

В состав Flex SDK входят инструменты командной строки для упаковки, компиляции и отладки приложений AIR.

- 1 Если это еще не сделано, загрузите пакет Flex SDK по адресу <http://opensource.adobe.com/wiki/display/flexsdk/Downloads>.
- 2 Распакуйте содержимое SDK в папку (например, во Flex SDK).
- 3 Скопируйте содержимое AIR SDK, заменив файлы в папке Flex SDK.

Примечание. На компьютерах Mac следует скопировать или заменить отдельные файлы в папках SDK, а не все каталоги. По умолчанию на Mac при копировании каталога в папку с таким же именем существующие файлы в целевом каталоге удаляются, то есть слияние двух каталогов не выполняется. Можно использовать команду `ditto` в окне терминала для объединения пакета AIR SDK с пакетом Flex SDK:
`ditto air_sdk_folder flex_sdk_folder`

4 Инструменты командной строки AIR находятся в папке bin.

Настройка внешних SDK

Чтобы приступить к разработке приложений для Android и iOS, необходимо загрузить файлы обеспечения, пакеты SDK и другие инструменты разработки, предоставляемые производителями платформы.

Дополнительные сведения по загрузке и установке Android SDK см. в разделе [«Разработчики Android: установка SDK»](#). Начиная с версии AIR 2.6 загружать Android SDK больше не требуется. AIR SDK теперь содержит базовые компоненты, необходимые для установки и запуска пакетов APK. Тем не менее, Android SDK можно использовать для выполнения различных задач разработки, включая создание и запуск программных эмуляторов и получение моментальных снимков экрана устройства.

Внешний SDK для разработки приложений для iOS не требуется. Однако необходимы специальные сертификаты и профили поставки. Дополнительные сведения см. в разделе [Получение файлов разработчика у компании Apple](#).

Глава 5. Создание первого приложения AIR

Создание первого настольного приложения Flex AIR с помощью Flash Builder

Чтобы быстро на практике продемонстрировать, как работает Adobe® AIR®, используйте эти инструкции по созданию и упаковке простейшего SWF-приложения AIR «Hello World» с помощью Adobe® Flash® Builder.

Загрузите и установите Flash Builder, если это еще не сделано. Кроме того, следует загрузить и установить последнюю версию среды Adobe AIR, которая доступна на веб-сайте www.adobe.com/go/air_ru.

Создание проекта AIR

Flash Builder включает инструменты, необходимые для разработки и упаковки приложений AIR.

В начале для создания приложений AIR во Flash Builder или Flex Builder выполняется тот же шаг, что и при создании других проектов приложений Flex: определяется новый проект.

- 1 Запустите Flash Builder.
- 2 Выберите «Файл» > «Создать» > «Проект Flex».
- 3 Введите AIRHelloWorld в качестве имени проекта.
- 4 Приложения AIR считаются типом приложений во Flex. Существует два типа.
 - Веб-приложения, которые запускаются в Adobe® Flash® Player.
 - Настольные приложения, которые запускаются в Adobe AIR.

Выберите тип настольных приложений.

- 5 Для создания проекта нажмите «Готово».

Проекты AIR изначально состоят из двух файлов: основного файла MXML и файла приложения XML (также называется файлом дескриптора приложения). В последнем файле определены свойства приложения.

Дополнительные сведения см. в разделе [Разработка приложений AIR в программе Flash Builder](#).

Написание кода приложения AIR

Для написания кода приложения «Hello World» выполняется правка файла приложения MXML (AIRHelloWorld.mxml), который открывается в редакторе. (Если файл не открыт, используйте навигатор проекта, чтобы открыть его.)

Приложения Flex AIR для настольного компьютера заключены в тег WindowedApplication в MXML. Тег WindowedApplication в MXML создает простое окно, включающее простейшие элементы управления окна, такие как строка заголовка и кнопка закрытия окна.

- 1 Добавьте атрибут `title` к компоненту `WindowedApplication` и назначьте ему значение "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 Добавьте в приложение компонент Label (поместите его внутрь тега WindowedApplication). Задайте для свойства text компонента Label значение "Hello AIR" и установите ограничения макета, чтобы обеспечить центрирование элемента, как показано на следующем рисунке:

```
<?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 Добавьте следующий блок стиля сразу после открытия тега WindowedApplication и перед тегом только что введенного компонента Label.

```
<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>
```

Эти настройки стиля применяются ко всему приложению и визуализируют фон окна в немного прозрачный серый цвет.

Код приложения теперь выглядит следующим образом:

```
<?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>
```

Далее будут изменены некоторые настройки в дескрипторе приложения, позволяющие обеспечить прозрачность в приложении:

- 1 На панели навигатора Flex найдите файл дескриптора приложения в исходном каталоге проекта. Если проект назван AIRHelloWorld, этот файл будет называться AIRHelloWorld-app.xml.
- 2 Дважды щелкните файл дескриптора приложения, чтобы отредактировать его в среде Flash Builder.
- 3 В коде XML найдите закомментированные строки для свойств `systemChrome` и `transparent` (в свойстве `initialWindow`). Удалите комментарии. (Удалите разграничители комментариев "`<!--`" и "`-->`".)

- 4 Задайте для свойства `systemChrome` текстовое значение `none`, как в следующем примере:

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


- 5 Задайте для свойства `transparent` текстовое значение `true`, как в следующем примере:

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

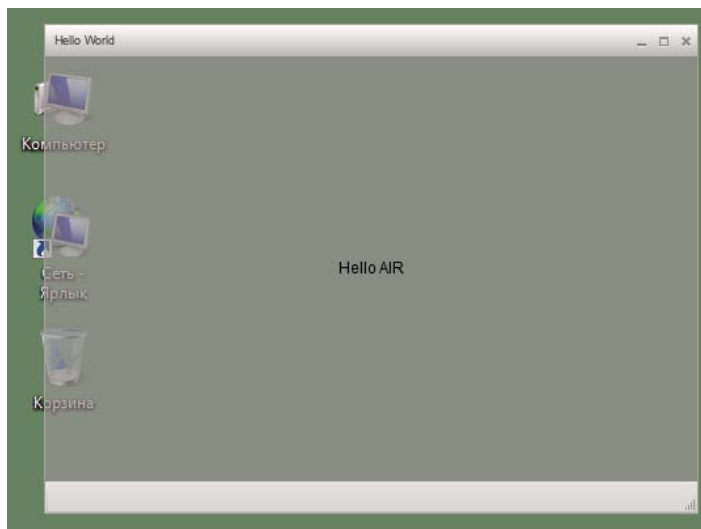
- 6 Сохраните файл.

Проверка приложения AIR

Чтобы проверить написанный код приложения, выполните его в режиме отладки.

- 1 Нажмите кнопку «Отладка»  на главной панели инструментов
Также можно выбрать в меню «Запуск» > «Отладка» > AIRHelloWorld.

Полученное приложение AIR должно выглядеть, как показано в следующем примере:



- 2 С использованием свойств `horizontalCenter` и `verticalCenter` элемента управления `Label` текст размещается в центре окна. Перемещайте или изменяйте размер окна также, как в любом другом приложении для настольного компьютера.

Примечание. Если приложение не компилируется, найдите и исправьте ошибки в синтаксисе или написании введенного кода. Ошибки и предупреждения отображаются в среде Flash Builder с помощью представления «Проблемы».

Упаковка, подписывание и выполнение приложения AIR

Теперь можно упаковать приложение «Hello World» в файл AIR для распространения. Файл AIR — это архивный файл, содержащий файлы приложения, к которым относятся все файлы в папке bin данного проекта. В этом простом примере такими файлами являются SWF- и XML-файлы приложения. Пакет AIR распространяется среди пользователей, которые затем используют его для установки приложения. Обязательным шагом в этом процессе является цифровая подпись приложения.

- 1 Убедитесь, что приложение правильно скомпилировано и работает как предполагалось.
- 2 Выберите «Проект» > «Экспорт сборки выпуска».
- 3 Убедитесь, что указан проект AIRHelloWorld и приложение AIRHelloWorld.mxml.
- 4 Выберите «Экспорт в виде подписанного пакета AIR». Нажмите кнопку «Далее».
- 5 Если имеется существующий цифровой сертификат, нажмите «Обзор», чтобы перейти к нему и выбрать его.
- 6 Если необходимо создать новый самозаверяющий цифровой сертификат, выберите «Создать».
- 7 Введите необходимую информацию и нажмите «ОК».
- 8 Нажмите кнопку «Готово», чтобы создать пакет AIR с названием AIRHelloWorld.air.

Теперь можно установить и запустить приложение из навигатора проектов в среде Flash Builder или из файловой системы, дважды щелкнув соответствующий файл AIR.

Создание первого настольного приложения AIR с помощью Flash Professional

Для быстрого практического знакомства с работой Adobe® AIR® следуйте инструкциям в этом разделе, создав и упаковав простейшее приложение AIR «Hello World» с помощью приложения Adobe® Flash® Professional.

Если это еще не сделано, загрузите и установите пакет Adobe AIR, расположенный по следующему адресу: www.adobe.com/go/air_ru.

Создание приложения «Hello World» в Adobe Flash

Создание приложения Adobe AIR в Adobe Flash во многом напоминает процесс создания любого другого FLA-файла. В следующей процедуре описаны шаги, позволяющие создать простейшее приложение Hello World с помощью Flash Professional.

Создание приложения «Hello World»

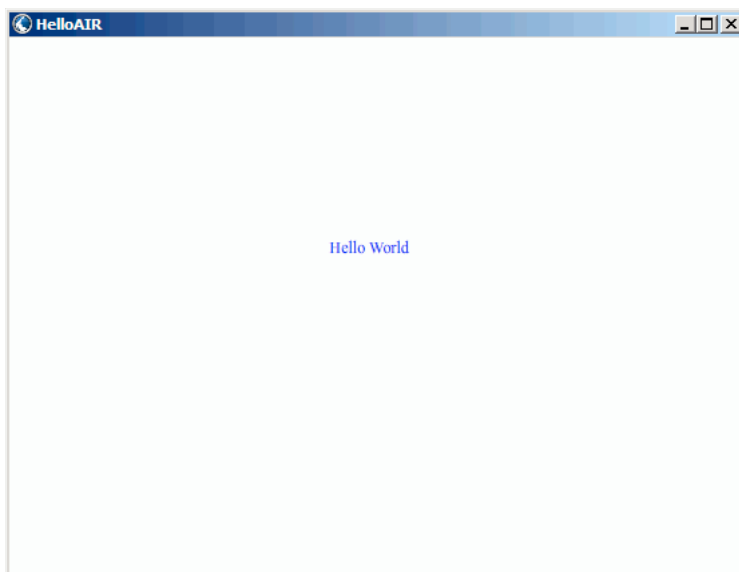
- 1 Запустите приложение Flash.
- 2 В окне приветствия нажмите AIR, чтобы создать пустой FLA-файл с параметрами публикации Adobe AIR.
- 3 Выберите инструмент «Текст» на панели «Инструменты» и создайте статическое текстовое поле (по умолчанию) в центре рабочей области. Сделайте поле достаточно широким, чтобы в нем помещалось 15-20 символов.
- 4 Введите текст «Hello World» в это текстовое поле.
- 5 Сохраните файл, задав для него имя (например, helloAIR).

Проверка приложения

- 1 Нажмите клавиши Ctrl + Enter или выберите «Управление» -> «Тестировать ролик» -> «Проверка», чтобы проверить приложение в Adobe AIR.
- 2 Для использования функции «Отладка ролика» вначале добавьте к приложению программный код ActionScript. Можно попробовать сделать это быстро, добавив инструкцию трассировки, как показано в следующем примере.

```
trace("Running AIR application using Debug Movie");
```
- 3 Нажмите клавиши Ctrl + Shift + Enter или выберите «Отладка» -> «Отладка ролика» -> «Отладка», чтобы запустить приложение в режиме отладки ролика.

Приложение «Hello World» выглядит, как показано на следующем рисунке.



Упаковка приложения

- 1 Выберите «Файл» > «Опубликовать».
- 2 Подпишите пакет Adobe AIR с использованием существующего цифрового сертификата или создайте самозаверенный сертификат, выполнив следующие действия:
 - a Возле поля «Сертификат» нажмите кнопку «Создать».
 - b Введите сведения в поля «Имя владельца», «Подразделение», «Название организации», «Электронная почта», «Страна», «Пароль» и «Подтверждение пароля».
 - c Укажите тип сертификата. Тип сертификата определяет уровень безопасности: 1024-RSA использует 1024-разрядный ключ (менее защищенный), а 2048-RSA использует 2048-разрядный ключ (более защищенный).
 - d Сохраните информацию в файле сертификата, выбрав строчку «Сохранить как», или нажмите кнопку «Обзор...», чтобы найти требуемую папку. (Например, *C:/Temp/mycert.pfx*.) Когда закончите, нажмите кнопку «ОК».

- e Adobe Flash открывает диалоговое окно «Цифровая подпись». Имя файла и путь к созданному цифровому сертификату для автоматического подписывания отображается в текстовом поле «Сертификат». Если этой информации нет, введите имя файла и путь к нему или нажмите кнопку «Обзор», чтобы найти файл и выбрать его.
- f В текстовом поле «Пароль» диалогового окна «Цифровая подпись» введите пароль, совпадающий с паролем, назначенным на шаге b. Дополнительные сведения о создании подписей приложений Adobe AIR см. в разделе «[«Цифровая подпись файлов AIR»](#) на странице 203».
- 3 Чтобы создать приложение и файл установщика, нажмите кнопку «Опубликовать». (В среде Flash CS4 и CS5 нажмите кнопку «ОК».) Необходимо выполнить команды «Тестировать ролик» или «Отладка ролика», чтобы создать SWF-файл и xml-файлы приложения перед созданием файла Adobe AIR.
- 4 Чтобы установить приложение, дважды щелкните AIR-файл (*application.air*) в той папке, где сохранено приложение.
- 5 Нажмите кнопку «Установить» в диалоговом окне «Установка приложения».
- 6 Просмотрите настройки установки и параметры расположения и убедитесь, что установлен флажок «Запуск приложения после установки». Нажмите кнопку «Продолжить».
- 7 Нажмите кнопку «Готово», когда появится сообщение о завершении установки приложения.

Создание первого приложения AIR for Android с помощью Flash Professional

Чтобы создавать приложения AIR for Android, требуется загрузить расширение Flash Professional CS5 для Android с веб-сайта [Adobe Labs](#).

Также необходимо загрузить и установить Android SDK с веб-сайта Android, как описано на странице [Разработчики Android: установка SDK](#).

Создание проекта

- 1 Откройте Flash Professional CS5.
- 2 Создайте новый проект AIR for Android.
На главной странице Flash Professional имеется ссылка для создания приложения AIR for Android. Также можно выбрать меню «Файл» > «Создать» и затем выбрать шаблон AIR for Android.
- 3 Сохраните документ как файл HelloWorld fla.

Написание кода

В данном руководстве процесс написания кода не рассматривается, поэтому с помощью инструмента «Текст» напишите «Hello, World!» в рабочей области.

Настройка свойств приложения

- 1 Выберите «Файл» > «Настройки AIR Android».
- 2 На вкладке «Общие» задайте следующие настройки.
 - Файл вывода: HelloWorld.apk
 - Имя приложения: HelloWorld
 - Идентификатор приложения: HelloWorld

- Номер версии: 0.0.1
 - Соотношение сторон: книжное
- 3 На вкладке «Развертывание» задайте следующие настройки.
- Сертификат: укажите действительный сертификат для подписи кода AIR. Нажмите кнопку «Создать», если требуется создать новый сертификат (Приложения Android, развертывание которых выполняется через Android Маркет, должны иметь сертификат, действующий по крайней мере до 2033 года.) Введите пароль сертификата в поле «Пароль».
 - Тип развертывания Android: отладка
 - После публикации: выберите оба параметра
 - Введите путь к инструменту ADB в подкаталоге tools каталога Android SDK.
- 4 Закройте диалоговое окно «Настройки Android», нажав кнопку «ОК».
- На этом этапе разработки значки и разрешения для приложения не требуются. Большинство приложений AIR for Android требуют определенные полномочия для получения доступа к защищенным функциям. Настраивать эти полномочия следует, только если они действительно требуются для приложения. Так как когда приложение запрашивает слишком много разрешений, пользователь может отказаться от его использования.
- 5 Сохраните файл.

Упаковка приложения и установка его на устройство Android

- 1 Убедитесь, что на устройстве включена отладка по USB. Отладку по USB можно включить в настройках приложения в разделе «Приложения» > «Разработка».
 - 2 Подключите устройство к компьютеру с помощью кабеля USB.
 - 3 Установите среду выполнения AIR, если она еще не установлена, загрузив Adobe AIR с Android Маркета. (AIR также можно установить локально с помощью «Команда ADT installRuntime» на странице 189. Пакеты Android, предназначенные для использования на устройствах Android и в эмуляторах, включены в AIR SDK.)
 - 4 Выберите «Файл» > «Опубликовать».
- Flash Professional создает файл APK, устанавливает приложение на подключенное устройство Android и запускает его.

Создание первого приложения AIR для iOS

AIR 2.6 и более поздние версии, iOS 4.2 и более поздние версии

Написание кода, создание сборки и тестирование базовых функций приложений для iOS можно выполнять с помощью инструментов Adobe. Однако для установки и тестирования приложения для iOS на устройстве и распространения этого приложения требуется присоединиться к программе Apple iOS Developer (участие в программе предусматривает оплату). После присоединения к программе iOS Developer вам будет открыт доступ к portalу iOS Provisioning Portal, на котором можно получить файлы и ресурсы от Apple, необходимые для установки приложения на устройстве, проведения тестирования и последующего распространения. Предоставляется доступ к следующим файлам и ресурсам:

- Сертификаты разработки и распространения
- Идентификаторы приложений

- Файлы для разработки и распространения

Создание содержимого приложения

Создайте SWF-файл, который выводит текст «Hello world!». Для этого можно использовать Flash Professional, Flash Builder или другую интегрированную среду разработки. В данном примере воспользуйтесь текстовым редактором и компилятором SWF с интерфейсом командной строки, который включен в пакет Flex SDK.

- 1 В удобном месте создайте каталог для хранения файлов приложения. Создайте файл с именем *HelloWorld.as* и отредактируйте файл в привычном редакторе кода.
- 2 Добавьте следующий код:

```
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 Скомпилируйте класс, используя компилятор amxmlc:

```
amxmlc HelloWorld.as
```

В той же папке создайте SWF-файл *HelloWorld.swf*.

Примечание. В данном примере предполагается, что путь к каталогу с компилятором *amxmlc* настроен в переменной среды *Path*. Сведения о настройке пути см. в разделе «[Переменные среды Path](#)» на странице 327». Как альтернативный вариант можно ввести полный путь к *amxmlc* и другим инструментам командной строки, используемым в данном примере.

Создание изображений для значка и начального экрана приложения

Все приложения для iOS имеют значки, которые отображаются в пользовательском интерфейсе приложения iTunes и на экране устройства.

- 1 Создайте подкаталог в каталоге проекта и присвойте ему имя *icons* (значки).
- 2 Создайте три PNG-файла в каталоге *icons*. Сохраните их под именами *Icon_29.png*, *Icon_57.png* и *Icon_512.png*.

- 3 Отредактируйте PNG-файлы, чтобы создать подходящие значки для приложения. Изображения должны иметь размеры 29 x 29, 57 x 57 и 512 x 512 пикселей. В целях тестирования просто используйте в качестве изображений одноцветные квадраты.

Примечание. При отправке приложения в магазин Apple App Store используется версия JPG (а не PNG) изображения со стороной 512 пикселей. Версия PNG используется при тестировании рабочих версий приложения.

Все приложения iPhone отображают начальное изображение в процессе загрузки на iPhone. Определите начальное изображение в PNG-файле.

- 1 В основном каталоге разработки создайте PNG-файл с именем Default.png. (Не помещайте этот файл в подкаталог icons. Файлу нужно обязательно присвоить имя Default.png, с прописной буквой D.)
- 2 Внесите в файл изменения, чтобы его ширина равнялась 320 пикселям, а высота 480 пикселям. Пока в качестве начального изображения можно использовать простой белый прямоугольник. (В дальнейшем его можно будет изменить.)

Дополнительные сведения об этих изображениях см. в разделе «[Значки приложения](#)» на странице 92».

Создание файла дескриптора приложения

Создайте файл дескриптора приложения, в котором определяются базовые свойства приложения. Это можно сделать с помощью ИСР, например Flash Builder, или текстового редактора.

- 1 В папке проекта, содержащей файл HelloWorld.as, создайте XML-файл с именем *HelloWorld-app.xml*. Отредактируйте этот файл в привычном редакторе XML.
- 2 Добавьте следующий XML-код:

```
<?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>
```

Для простоты в данном примере настраивается только несколько свойств.

Примечание. Если используется AIR 2 или более ранней версии, необходимо использовать элемент `<version>` вместо элемента `<versionNumber>`.

- 3 Измените идентификатор приложения, установив идентификатор приложения, полученные на портале iOS Provisioning Portal (не включайте случайный идентификатор начального числа пакета).
- 4 Выполните тестирование приложения с помощью ADL:
adl HelloWorld-app.xml -screenize iPhone

ADL должен открывать окно на компьютере с текстом «Hello World!». Если этого не происходит, проверьте исходный код и дескриптор приложения на наличие ошибок.

Компиляция файла IPA

Теперь можно скомпилировать файл установщика IPA с помощью ADT. Вам требуется сертификат разработчика Apple и закрытый ключ в формате P12, а также профиль обеспечения разработки Apple.

Запустите утилиту ADT со следующими параметрами, подставив собственные значения для аргументов keystore, storepass и provisioning-profile:

```
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
```

(Введите команду в одной строке. Разрывы строки в этом примере используются только в целях удобства восприятия.)

ADT сгенерирует файл установщика приложения для iOS *HelloWorld.ipa*, который будет помещен в каталог проекта. Компиляция файла IPA может занять несколько минут.

Установка приложения на устройстве

Процедура установки приложения для iOS для тестирования:

- 1 Откройте приложение iTunes.
- 2 Если это еще не сделано, добавьте профиль обеспечения данного приложения в iTunes. В iTunes выберите «Файл» > «Добавить в медиатеку». Затем выберите файл профиля обеспечения (с типом файла mobileprovision).

На данном этапе для тестирования приложения на рабочем устройстве можно использовать профиль обеспечения разработки.

Позднее, при отправке приложения в iTunes Store, потребуются использовать профиль распространения. Для специального распространения приложения (на несколько устройств без отправки в iTunes Store) нужно использовать специальный профиль обеспечения.

Дополнительные сведения о профилях поставки см. в разделе «[Настройка ОС iOS](#)» на странице 69».

- 3 В некоторых версиях iTunes не выполняется замена, если уже установлена та же версия приложения. В таком случае удалите приложение с устройства и из списка приложений в iTunes.
- 4 Дважды щелкните IPA-файл для приложения. Он должен быть указан в списке приложений в iTunes.
- 5 Подключите устройство к порту USB компьютера.
- 6 В iTunes перейдите на вкладку «Программа» для устройства и проверьте, что программа выбрана в списке устанавливаемых.
- 7 Выберите устройство в списке на левой панели программы iTunes. Затем нажмите кнопку «Синхронизировать». По окончании синхронизации приложение Hello World появится на устройстве iPhone.

Если новая версия не установилась, удалите приложение с устройства и из списка приложений в iTunes, а затем повторите процедуру. Такое происходит, если текущая установленная версия использует тот же идентификатор приложения и номер версии.

Редактирование изображения начального экрана

Перед компиляцией приложения был создан файл Default.png (см. раздел «[Создание изображений для значка и начального экрана приложения](#)» на странице 28»). Этот PNG-файл служит начальным экраном, который отображается во время загрузки приложения. При тестировании приложения на iPhone вы, возможно, заметили этот пустой экран при загрузке.

Это изображение необходимо изменить, чтобы оно соответствовало начальному экрану приложения («Здравствуй, мир!»).

- 1 Откройте приложение на устройстве. Когда появится первый текст «Здравствуй, мир!», нажмите и удерживайте кнопку «Домой» (под экраном). Удерживая кнопку «Домой», нажмите кнопку «Питание/Режим сна» (вверху на iPhone). В результате будет сделан снимок экрана и отправлен в приложение «Фотопленка».
- 2 Перенесите изображение на рабочий компьютер с помощью функции переноса фотографий из iPhoto или другого приложения для перемещения фотографий. (В ОС Mac OS также можно использовать программу «Захват изображений».)

Также фотографию можно отправить на рабочий компьютер по электронной почте.

- Откройте приложение «Фотографии».
- Откройте инструмент «Фотопленка».
- Откройте сделанный снимок экрана.
- Нажмите изображение, а затем нажмите кнопку со стрелкой («переслать») в нижнем левом углу. Нажмите кнопку «Отправить по e-mail» и отправьте изображение на свой адрес.

- 3 Замените файл Default.png (в каталоге проекта) на PNG-версию снимка экрана.
- 4 Выполните повторную компиляцию приложения (см. раздел «[Компиляция файла IPA](#)» на странице 30») и переустановите его на устройство.

Теперь при загрузке приложения отображается новый начальный экран.

***Примечание.** В файле Default.png может использоваться любая графика при условии сохранения правильных размеров (320 x 480 пикселей). Однако в большинстве случаев рекомендуется, чтобы изображение Default.png соответствовало начальному экрану приложения.*

Создание первого HTML-приложения AIR с помощью Dreamweaver

Чтобы быстро на практике продемонстрировать как работает Adobe® AIR®, используйте эти инструкции по созданию и упаковке простейшего HTML-приложения AIR «Hello World», используя расширение Adobe® AIR® Extension для Dreamweaver®.

Если это еще не сделано, загрузите и установите пакет Adobe AIR, расположенный по следующему адресу: www.adobe.com/go/air_ru.

Инструкции по установке расширения Adobe AIR для Dreamweaver см. в разделе [«Установка расширения AIR для Dreamweaver»](#).

Обзор расширения, включая системные требования, см. в разделе [«Расширение AIR для Dreamweaver»](#).

***Примечание.** Приложения AIR на базе HTML могут быть разработаны только для профилей desktop и extendedDesktop. Мобильный профиль не поддерживается.*

Подготовка файлов приложений

Начальная страница приложения Adobe AIR и все относящиеся к нему страницы должны быть определены на сайте Dreamweaver:

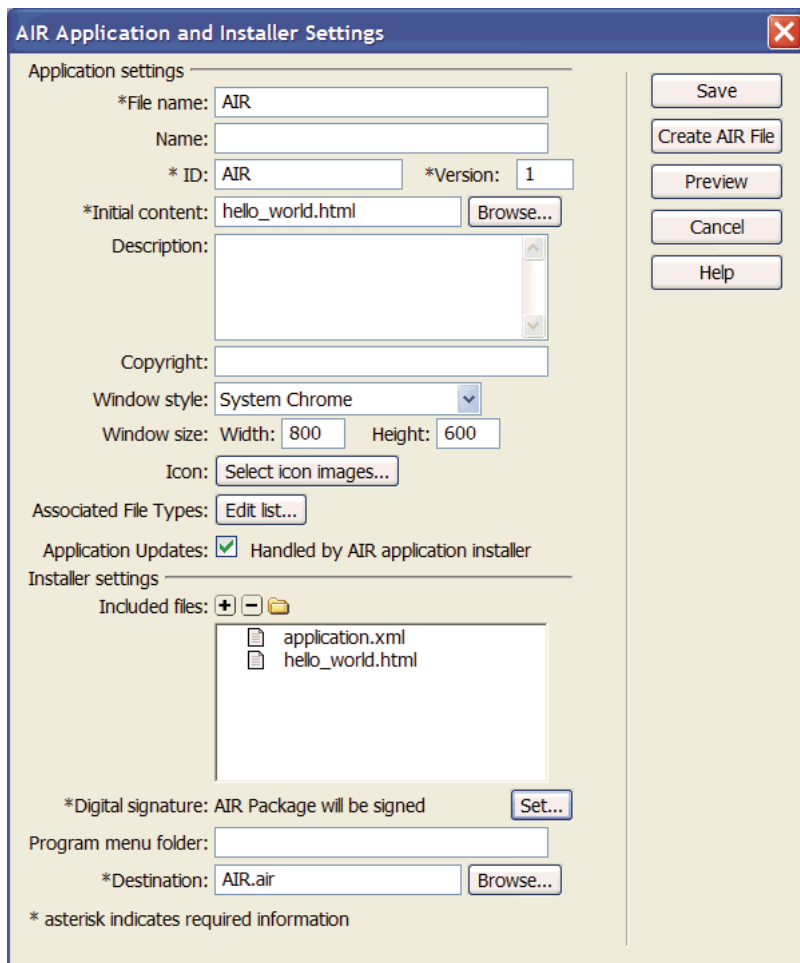
- 1 Запустите Dreamweaver и убедитесь, что был определен сайт.
- 2 Откройте новую страницу HTML, выбрав «Файл» > «Создать», затем выберите HTML в столбце «Тип страницы», выберите «Нет» в столбце «Макет», щелкните «Создать».
- 3 На новой странице введите **Hello World!**
Этот пример исключительно прост, но, если потребуется, можно добавить к тексту стиль по своему выбору, добавить дополнительное содержимое к странице, связать другие страницы с начальной и т.п.
- 4 Сохраните страницы («Файл» > «Сохранить») как файл hello_world.html. Убедитесь, что файл сохранен на сайте Dreamweaver.

Дополнительные сведения о сайте Dreamweaver см. в справке Dreamweaver.

Создание приложения Adobe AIR

- 1 Убедитесь, что страница hello_world.html открыта в окне «Документ» в Dreamweaver. (См. инструкции по ее созданию в предыдущем разделе.)
- 2 Выберите «Сайт» > «Настройки приложения Air».
Большинство требуемых настроек в диалоговом окне «AIR — приложение и настройки» заполняются автоматически. Но требуется выбрать исходное содержимое (или начальную страницу) для приложения.
- 3 Нажмите кнопку «Обзор», рядом с пунктом «Исходное содержимое», чтобы перейти к странице hello_world.html и выбрать ее.
- 4 Рядом с пунктом «Цифровая подпись» нажмите кнопку «Задать».
Цифровая подпись позволяет гарантировать, что код приложения не был изменен или поврежден с момента его создания автором программного обеспечения. Такая подпись необходима для всех приложений Adobe AIR.
- 5 В диалоговом окне «Цифровая подпись» выберите «Подписать пакет AIR цифровым сертификатом», а затем нажмите кнопку «Создать». (Если доступ к цифровому сертификату уже есть, можно нажать кнопку «Обзор» чтобы выбрать его.)
- 6 Заполните обязательные поля в диалоговом окне «Самозаверяющий цифровой сертификат». Необходимо указать свое имя, ввести пароль и подтвердить его, а затем ввести имя для файла цифрового сертификата. Dreamweaver сохраняет цифровой сертификат в корневом каталоге сайта.
- 7 Нажмите «ОК», чтобы вернуться в диалоговое окно «Цифровая подпись».
- 8 В диалоговом окне «Цифровая подпись» введите пароль, указанный для цифрового сертификата, и нажмите «ОК».

Заполненное диалоговое окно «AIR — настройки приложения и установщика» может выглядеть следующим образом.



Подробное описание всех параметров этого диалогового окна и их настройки см. в разделе [«Создание приложения AIR в Dreamweaver»](#).

9 Нажмите кнопку «Создать файл AIR».

Dreamweaver создает файл приложения Adobe AIR и сохраняет его в корневую папку сайта. Dreamweaver также создает файл application.xml и сохраняет его в той же папке. Этот файл выполняет роль манифеста, определяя различные свойства приложения.

Установка приложения на настольном компьютере

Теперь, когда создан файл приложения, можно установить его на любом настольном компьютере.

1 Переместите файл приложения Adobe AIR с сайта Dreamweaver на персональный компьютер.

Этот шаг не обязателен. Если потребуется, приложение можно установить на свой компьютер непосредственно из каталога на сайте Dreamweaver.

2 Дважды щелкните исполняемый файл приложения (файл .air), чтобы установить это приложение.

Предварительный просмотр приложения Adobe AIR

В любое время можно выполнить предварительный просмотр страниц, которые затем станут частью приложений AIR. Поэтому нет необходимости упаковывать приложение для того, чтобы посмотреть, как оно будет выглядеть после установки.

- 1 Убедитесь, что страница `hello_world.html` открыта в окне «Документ» в Dreamweaver.
- 2 На панели инструментов «Документ» нажмите кнопку «Просмотр и отладка в браузере», а затем выберите «Просмотр в AIR».

Также можно нажать комбинацию клавиш `Ctrl+Shift+F12` (Windows) или `Cmd+Shift+F12` (Macintosh).

При выполнении предварительного просмотра данной страницы отображается то, что пользователи увидят на начальной странице приложения после его установки на настольный компьютер.

Создание первого HTML-приложения AIR с помощью комплекта AIR SDK

Чтобы быстро на практике продемонстрировать как работает Adobe® AIR®, используйте эти инструкции по созданию и упаковке простейшего HTML-приложения AIR «Hello World».

Для начала необходимо установить среду выполнения и настроить AIR SDK. Используются инструменты *AIR Debug Launcher* (ADL) и *AIR Developer Tool* (ADT), описанные в данном руководстве. ADL и ADT — это служебные программы, запускаемые из командной строки, которые можно найти в каталоге `bin` в пакете AIR SDK (см. раздел «[Установка AIR SDK](#)» на странице 17»). Предполагается, что читатель данного руководства уже знаком с тем, как запускать программы из командной строки и знает как настроить необходимые переменные пути в операционной системе.

Примечание. Если используется продукт Adobe® Dreamweaver®, ознакомьтесь с разделом «[Создание первого HTML-приложения AIR с помощью Dreamweaver](#)» на странице 31».

Примечание. Приложения AIR на базе HTML могут быть разработаны только для профилей `desktop` и `extendedDesktop`. Мобильный профиль не поддерживается.

Создание файлов проекта

В каждом HTML-приложении AIR содержатся следующие два файла: файл дескриптора приложения, определяющий метаданные приложения, и страница HTML верхнего уровня. В дополнение к этим обязательным файлам данный проект содержит файл с кодом JavaScript, `AIRAliases.js`, который определяет подходящие псевдонимы-переменные для классов прикладного интерфейса программирования AIR.

- 1 Создайте каталог с именем `helloWorld`, в котором будут содержаться файлы проекта.
- 2 Создайте файл XML с именем `helloWorld-app.xml`.
- 3 Создайте файл HTML с именем `helloWorld.html`.
- 4 Скопируйте файл `AIRAliases.js` из папки `frameworks` в AIR SDK в каталог проекта.

Создание файла дескриптора приложения AIR

Чтобы приступить к разработке приложения AIR, создайте XML-файл дескриптора приложения со следующей структурой:

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

- 1 Откройте файл `HelloWorld-app.xml` для редактирования.
- 2 Добавьте корневой элемент `<application>`, включая атрибут пространства имен AIR:
`<application xmlns="http://ns.adobe.com/air/application/2.7">` Последний сегмент пространства имен (2.7) указывает версию среды выполнения, которая требуется для приложения.
- 3 Добавьте элемент `<id>`:
`<id>examples.html.HelloWorld</id>` Идентификатор приложения однозначно идентифицирует приложение в сочетании с идентификатором издателя (который AIR извлекает из сертификата, используемого для подписи пакета приложения). Идентификатор приложения используется для установки, обращения к частному каталогу хранилища файловой системы приложения, доступа к частному зашифрованному хранилищу и взаимодействия между приложениями.
- 4 Добавьте элемент `<versionNumber>`:
`<versionNumber>0.1</versionNumber>` Помогает пользователям определить, какую версию приложения они устанавливают.
Примечание. Если используется AIR 2 или более ранней версии, необходимо использовать элемент `<version>` вместо элемента `<versionNumber>`.
- 5 Добавьте элемент `<filename>`:
`<filename>HelloWorld</filename>` Имя, используемое для выполняемого файла приложения, каталога установки и других ссылок на приложение в операционной системе.
- 6 Добавьте элемент `<initialWindow>`, содержащий следующие дочерние элементы, чтобы указать свойства исходного окна приложения:
`<content>HelloWorld.html</content>` Указывает корневой HTML-файл приложения AIR для загрузки.
`<visible>true</visible>` Сразу делает окно видимым.
`<width>400</width>` Задаёт ширину окна (в пикселах).
`<height>200</height>` Задаёт высоту окна.
- 7 Сохраните файл. Завершённый файл дескриптора приложения должен выглядеть следующим образом.


```
<?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>
```

В этом примере устанавливается всего несколько возможных свойств приложения. Полный набор свойств приложения, которые позволяют указывать такие аспекты как хром окна, размер окна, прозрачность, каталог установки по умолчанию, связанные типы файлов и значки приложения, см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».

Создание HTML-страницы приложения

Теперь необходимо создать простую HTML-страницу, которая будет выполнять роль основного файла приложения AIR.

- 1 Откройте файл `helloWorld.html` для редактирования. Добавьте следующий код HTML:

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

- 2 В разделе `<head>` HTML, импортируйте файл `AIRAliases.js`:

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

AIR определяет свойство с именем `runtime` для объекта окна HTML. Свойство `runtime` обеспечивает доступ к встроенным классам AIR, используя полностью уточненное имя пакета для класса. Например, чтобы создать объект файла AIR, можно добавить следующую инструкцию в JavaScript:

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

Файл `AIRAliases.js` определяет подходящие псевдонимы для наиболее полезных прикладных программных интерфейсов AIR. Используя `AIRAliases.js`, можно укоротить ссылку на класс `File` следующим образом:

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

- 3 Ниже приведен тег скрипта `AIRAliases`, добавьте другой тег скрипта, содержащий функцию JavaScript для обработки события `onLoad`:

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

Функция `appLoad()` просто вызывает функцию `air.trace()`. При запуске приложения с помощью ADL сообщение трассировки вводится на консоль управления. Сообщения трассировки могут быть очень полезны для отладки.

4 Сохраните файл.

Файл `HelloWorld.html` должен выглядеть следующим образом:

```
<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>
```

Проверка приложения

Чтобы запустить и проверить приложение из командной строки, используйте утилиту AIR Debug Launcher (ADL). Исполняемый файл ADL можно найти в каталоге `bin` пакета AIR SDK. Если настройка AIR SDK еще не выполнена, см. раздел «[Установка AIR SDK](#)» на странице 17».

- 1 Откройте консоль управления или оболочку ОС. Перейдите в каталог, который создан для этого проекта.
- 2 Выполните следующую команду:

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

Открывается окно AIR, в котором отображается приложение. Также в окне консоли отображается сообщение, полученное в результате вызова `air.trace()`.

Дополнительные сведения см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».

Создание файла установки AIR

Если приложение успешно запущено, можно использовать утилиту ADT, чтобы упаковать приложение в файл установки AIR. Файл установки AIR — это файл архива, в котором содержатся все файлы приложения, которые можно передавать другим пользователям. Перед установкой такого упакованного файла AIR необходимо установить Adobe AIR.

Чтобы обеспечить защиту приложения, все файлы установки AIR должны содержать цифровую подпись. С помощью ADT или другого инструмента генерации сертификатов можно создать простой, самозаверяющий сертификат, используемый в целях разработки. Также можно приобрести коммерческий сертификат для подписывания кодов, выпущенный специальными органами сертификации, например компаниями VeriSign или Thawte. Если пользователь устанавливает файл AIR с самозаверяющим сертификатом, то в ходе процесса установки показывается «неизвестный» издатель. Это объясняется тем, что самозаверяющие сертификаты гарантируют лишь то, что файл AIR не изменялся с момента своего создания. Ничто не сможет помешать

злоумышленнику сделать файл AIR с самозаверяющим сертификатом и представить его как ваше приложение. Поэтому в выпускаемых для широкого использования файлах AIR рекомендуется использовать обеспечивающие возможность проверки коммерческие сертификаты. Краткие сведения о проблемах системы защиты AIR см. в разделе [Система защиты AIR](#) (для разработчиков ActionScript) или [Система защиты AIR](#) (для разработчиков HTML).

Создание самозаверяющих подписывающих сертификатов и пары ключей

- ❖ Из командной строки выполните следующую команду (выполняемый файл ADT находится в каталоге bin пакета AIR SDK):

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

ADT создает файл хранения ключа с именем *sampleCert.pfx*, содержащий сертификат и связанный личный ключ.

В этом примере используется минимальный набор атрибутов, которые могут быть установлены для сертификата. Ключ должен иметь тип *1024-RSA* или *2048-RSA* (см. раздел «[Подписание приложений AIR](#)» на странице 203»).

Создание файла установки AIR

- ❖ Из командной строки выполните следующую команду (вводится одной строкой):

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

Будет выдано предложение ввести пароль для файла ключей.

Аргумент HelloWorld.air — это файл AIR, создаваемый ADT. HelloWorld-app.xml — это файл дескриптора приложения. Последующими аргументами являются файлы, используемые приложением. В этом примере используется только два файла, но можно включить в проект любое число файлов и каталогов. ADT проверяет, что главный файл с содержимым HelloWorld.html включен в пакет. Однако если вы забудете включить файл AIRAliases.js, приложение не будет работать.

После создания пакета AIR можно установить и выполнить приложение, дважды щелкнув этот упакованный файл. Также можно ввести имя файла AIR, используя команду в оболочке ОС или консоль управления.

Следующие шаги

В приложении AIR коды HTML и JavaScript обычно выполняются так же, как и в стандартном веб-обозревателе. (Действительно, AIR использует тот же механизм визуализации WebKit, который применяется веб-обозревателем Safari.) Однако существует несколько важных отличий, которые необходимо учитывать при разработке HTML-приложения в AIR. Дополнительные сведения об этих различиях, а также другие полезные сведения см. в разделе [Программирование HTML и JavaScript](#).

Создание первого настольного приложения AIR с использованием пакета Flex SDK

Для быстрого и наглядного представления принципов работы Adobe® AIR® используйте эти инструкции по созданию простого SWF-приложения AIR «Hello World» с помощью пакета Flex SDK. В данном учебном пособии описаны процедуры компиляции, тестирования и упаковки приложений AIR с помощью инструментов командной строки, включенных в пакет Flex SDK (Flex SDK содержит AIR SDK).

Для начала необходимо установить среду выполнения и настроить Adobe® Flex™. В этом пособии показано использование компилятора *AMXMLC*, средства запуска *AIR Debug Launcher* (ADL) и инструмента *AIR Developer Tool* (ADT). Эти программы находятся в каталоге `bin` среды Flex SDK (см. раздел «[Настройка Flex SDK](#)» на странице 19).

Создание файла дескриптора приложения AIR

В этом разделе описаны принципы создания дескриптора приложения, представляющего собой XML-файл со следующей структурой:

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

1 Создайте XML-файл `helloWorld-app.xml` и сохраните его в каталоге проекта.

2 Добавьте элемент `<application>`, включая атрибут пространства имен AIR:

`<application xmlns="http://ns.adobe.com/air/application/2.7">` Последний сегмент пространства имен (2.7) задает версию среды выполнения, которая требуется для приложения.

3 Добавьте элемент `<id>`:

`<id>samples.flex.HelloWorld</id>` Идентификатор приложения однозначно идентифицирует приложение в сочетании с идентификатором издателя (который AIR извлекает из сертификата, используемого для подписи пакета приложения). Рекомендуемым форматом является обратная запись строки DNS с использованием точки в качестве разделителя, например `"com.company.AppName"`. Идентификатор приложения используется для установки, обращения к частному каталогу хранилища файловой системы приложения, доступа к частному зашифрованному хранилищу и взаимодействия между приложениями.

4 Добавьте элемент `<versionNumber>`:

`<versionNumber>1.0</versionNumber>` Помогает пользователям определить, какую версию приложения они устанавливают.

Примечание. Если используется AIR 2 или более ранней версии, необходимо использовать элемент `<version>` вместо элемента `<versionNumber>`.

5 Добавьте элемент `<filename>`:

`<filename>HelloWorld</filename>` Имя, используемое для выполняемого файла приложения, каталога установки и аналогичных ссылок на приложение в операционной системе.

- 6 Добавьте элемент `<initialWindow>`, содержащий следующие дочерние элементы, чтобы указать свойства исходного окна приложения:

`<content>HelloWorld.swf</content>` Определяет корневой SWF-файл приложения AIR для загрузки.

`<visible>true</visible>` Сразу делает окно видимым.

`<width>400</width>` Задаёт ширину окна (в пикселах).

`<height>200</height>` Задаёт высоту окна.

- 7 Сохраните файл. По завершении файл дескриптора приложения должен иметь следующий вид:

```
<?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>
```

В этом примере устанавливается всего несколько возможных свойств приложения. Полный набор свойств приложения, которые позволяют указывать такие аспекты как хром окна, размер окна, прозрачность, каталог установки по умолчанию, связанные типы файлов и значки приложения, см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».

Написание кода приложения

***Примечание.** В SWF-приложениях AIR может использоваться основной класс, заданный с использованием MXML или Adobe® ActionScript® 3.0. В этом примере MXML-файл используется для определения основного класса. Процесс создания приложения AIR с использованием основного класса ActionScript аналогичен. Вместо компиляции файла MXML в SWF-файл выполняется компиляция файла класса ActionScript. При использовании ActionScript основной класс должен расширять `flash.display.Sprite`.*

Как и во всех приложениях Flex, в приложениях AIR, созданных в среде Flex, содержится основной файл MXML. В приложениях AIR для настольных компьютеров в качестве корневого элемента используется компонент `WindowedApplication`, а не компонент `Application`. Компонент `WindowedApplication` предоставляет свойства, методы и события для управления приложением и его начальным окном. На платформах и в профилях, для которых AIR не поддерживает несколько окон, следует использовать компонент `Application`. В мобильных приложениях Flex можно также использовать компоненты `View` и `TabbedViewNavigatorApplication`.

Следующая процедура позволяет создать приложение «Hello World»:

- 1 В текстовом редакторе создайте файл `HelloWorld.mxml` и добавьте следующий код 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 Далее добавьте компонент Label в приложение (поместите его внутрь тега WindowedApplication).
- 3 Задайте для свойства text компонента Label значение "Hello AIR".
- 4 Установите ограничения макета, чтобы он всегда располагался по центру.

В следующем примере показан код на данном этапе:

```
<?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>
```

Компиляция приложения

Перед выполнением и отладкой приложения скомпилируйте код MXML в SWF-файл с использованием компилятора amxmlc. Компилятор amxmlc находится в каталоге bin пакета Flex SDK. При необходимости можно включить каталог bin пакета Flex SDK в переменную среды для пути на компьютере. Настройка пути делает выполнение утилит в командной строке более простым.

- 1 Откройте командную среду или терминал команд и перейдите к папке проекта приложения AIR.
- 2 Введите следующую команду:

```
amxmlc HelloWorld.mxml
```

При выполнении компилятора amxmlc создается файл HelloWorld.swf, содержащий скомпилированный код приложения.

Примечание. Если не удастся скомпилировать приложение, исправьте синтаксические или орфографические ошибки. Ошибки и предупреждения отображаются в окне консоли, используемом для выполнения компилятора amxmlc.

Дополнительные сведения см. в разделе «[Компиляция исходных файлов MXML и ActionScript для AIR](#)» на странице 166».

Проверка приложения

Чтобы выполнить и протестировать приложение из командной строки, используйте инструмент AIR Debug Launcher (ADL) для запуска приложения с помощью соответствующего файла дескриптора приложения. (Инструмент ADL находится в каталоге bin пакета Flex SDK.)

- ❖ В командной строке введите следующую команду:

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

Получившаяся программа AIR должна выглядеть примерно так:



С использованием свойств `horizontalCenter` и `verticalCenter` элемента управления `Label` текст размещается в центре окна. Перемещайте или изменяйте размер окна также, как в любом другом приложении для настольного компьютера.

Дополнительные сведения см. в разделе «[AIR Debug Launcher \(ADL\)](#)» на странице 170».

Создание файла установки AIR

Если приложение успешно запущено, можно использовать утилиту ADT, чтобы упаковать приложение в файл установки AIR. Файл установки AIR — это файл архива, в котором содержатся все файлы приложения, которые можно передавать другим пользователям. Перед установкой такого упакованного файла AIR необходимо установить Adobe AIR.

Чтобы обеспечить защиту приложения, все файлы установки AIR должны содержать цифровую подпись. С помощью ADT или другого инструмента генерации сертификатов можно создать простой, самозаверяющий сертификат, используемый в целях разработки. Можно также приобрести коммерческий сертификат подписи кода в коммерческом центре сертификации. Если пользователь устанавливает файл AIR с самозаверяющим сертификатом, то в ходе процесса установки показывается «неизвестный» издатель. Это объясняется тем, что самозаверяющие сертификаты гарантируют лишь то, что файл AIR не изменялся с момента своего создания. Ничто не сможет помешать злоумышленнику сделать файл AIR с самозаверяющим сертификатом и представить его как ваше приложение. Поэтому в выпускаемых для широкого использования файлах AIR рекомендуется использовать обеспечивающие возможность проверки коммерческие сертификаты. Краткие сведения о проблемах системы защиты AIR см. в разделе [Система защиты AIR](#) (для разработчиков ActionScript) или [Система защиты AIR](#) (для разработчиков HTML).

Создание самозаверяющих подписывающих сертификатов и пары ключей

- ❖ В командной строке введите следующую команду (выполняемый файл ADT находится в каталоге `bin` пакета Flex SDK):

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

В этом примере используется минимальный набор атрибутов, которые могут быть установлены для сертификата. Ключ должен иметь тип `1024-RSA` или `2048-RSA` (см. раздел «[Подписание приложений AIR](#)» на странице 203»).

Создание пакета AIR

- ❖ Из командной строки выполните следующую команду (вводится одной строкой):

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

Будет выдано предложение ввести пароль для файла ключей. Введите пароль и нажмите клавишу Enter. Символы пароля не отображаются с целью обеспечения безопасности.

Аргумент HelloWorld.air — это файл AIR, создаваемый ADT. HelloWorld-app.xml — это файл дескриптора приложения. Последующими аргументами являются файлы, используемые приложением. В этом примере используются только три файла, но можно включить любое число файлов и каталогов.

После создания пакета AIR можно установить и выполнить приложение, дважды щелкнув этот упакованный файл. Также можно ввести имя файла AIR, используя команду в оболочке ОС или консоль управления.

Дополнительные сведения см. в разделе «[Упаковка файла установки AIR для настольных систем](#)» на странице 55».

Создание первого приложения AIR для Android с использованием пакета Flex SDK

Чтобы начать работу, установите и настройте AIR SDK и Flex SDK. В данном учебном пособии используется компилятор *AMXMLC* из пакета Flex SDK, а также *AIR Debug Launcher* (ADL) и *AIR Developer Tool* (ADT) из пакета AIR SDK. См. раздел «[Настройка Flex SDK](#)» на странице 19».

Также необходимо загрузить и установить Android SDK с веб-сайта Android, как описано на странице [Разработчики Android: установка SDK](#).

Примечание. Дополнительные сведения о разработке приложений для iPhone см. в разделе «[Создание приложения Hello World для iPhone с помощью Flash Professional CS5](#)».

Создание файла дескриптора приложения AIR

В этом разделе описаны принципы создания дескриптора приложения, представляющего собой XML-файл со следующей структурой:

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

1 Создайте XML-файл HelloWorld-app.xml и сохраните его в каталоге проекта.

2 Добавьте элемент <application>, включая атрибут пространства имен AIR:

```
<application xmlns="http://ns.adobe.com/air/application/2.7">
```

 Последний сегмент пространства имен (2.7) задает версию среды выполнения, которая требуется для приложения.

3 Добавьте элемент <id>:

`<id>samples.android.HelloWorld</id>` Идентификатор приложения однозначно идентифицирует приложение в сочетании с идентификатором издателя (который AIR извлекает из сертификата, используемого для подписи пакета приложения). Рекомендуемым форматом является обратная запись строки DNS с использованием точки в качестве разделителя, например "com.company.AppName".

- 4 Добавьте элемент `<versionNumber>`:

`<versionNumber>0.0.1</versionNumber>` Помогает пользователям определить, какую версию приложения они устанавливают.

- 5 Добавьте элемент `<filename>`:

`<filename>HelloWorld</filename>` Имя, используемое для выполняемого файла приложения, каталога установки и аналогичных ссылок на приложение в операционной системе.

- 6 Добавьте элемент `<initialWindow>`, содержащий следующие дочерние элементы, чтобы указать свойства исходного окна приложения:

`<content>HelloWorld.swf</content>` Определяет корневой HTML-файл приложения AIR для загрузки.

- 7 Добавьте элемент `<supportedProfiles>`.

`<supportedProfiles>mobileDevice</supportedProfiles>` Указывает, что приложение запускается только в профиле мобильного устройства.

- 8 Сохраните файл. По завершении файл дескриптора приложения должен иметь следующий вид:

```
<?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>
```

В этом примере устанавливается всего несколько возможных свойств приложения. В файле дескриптора приложения также можно использовать другие настройки. Например, можно добавить значение `<fullScreen>true</fullScreen>` в элемент `initialWindow`, чтобы создать полноэкранное приложение. Для включения удаленной отладки и функций управляемого доступа на платформе Android в дескриптор приложения необходимо добавить разрешения Android. Данное простое приложение не требует разрешений, поэтому сейчас их добавлять не нужно.

Дополнительные сведения см. в разделе «[Настройка свойств мобильного приложения](#)» на странице 74».

Написание кода приложения

Создайте файл с именем `HelloWorld.as` и в текстовом редакторе добавьте следующий код:

```
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 );
        }
    }
}
```

Компиляция приложения

Перед выполнением и отладкой приложения скомпилируйте код MXML в SWF-файл с использованием компилятора `amxmlc`. Компилятор `amxmlc` находится в каталоге `bin` пакета Flex SDK. При необходимости можно включить каталог `bin` пакета Flex SDK в переменную среды для пути на компьютере. Настройка пути делает выполнение утилит в командной строке более простым.

- 1 Откройте командную среду или терминал команд и перейдите к папке проекта приложения AIR.
- 2 Введите следующую команду:

```
amxmlc HelloWorld.as
```

При выполнении компилятора `amxmlc` создается файл `HelloWorld.swf`, содержащий скомпилированный код приложения.

***Примечание.** Если не удастся скомпилировать приложение, исправьте синтаксические или орфографические ошибки. Ошибки и предупреждения отображаются в окне консоли, используемом для выполнения компилятора `amxmlc`.*

Дополнительные сведения см. в разделе «[Компиляция исходных файлов MXML и ActionScript для AIR](#)» на странице 166».

Проверка приложения

Чтобы выполнить и протестировать приложение из командной строки, используйте инструмент AIR Debug Launcher (ADL) для запуска приложения с помощью соответствующего файла дескриптора приложения. (Инструмент ADL находится в каталоге `bin` пакетов AIR SDK и Flex SDK.)

- ❖ В командной строке введите следующую команду:

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

Дополнительные сведения см. в разделе «[Моделирование устройства с помощью ADL](#)» на странице 106».

Создание файла пакета APK

Если приложение успешно запущено, можно использовать утилиту ADT, чтобы упаковать приложение в файл пакета APK. Файл пакета APK — это собственный формат файлов приложений Android, которые используются для распространения приложениям пользователям.

Все приложения Android должны быть подписаны. В отличие от файлов AIR приложения Android должны быть подписаны с помощью самозаверяющего сертификата. Операционная система Android не пытается установить личность разработчика приложения. Для подписания пакетов Android можно использовать сертификат, сгенерированный с помощью ADT. Сертификаты, используемые для приложений, которые размещаются на Android Маркете, должны иметь срок действия не менее 25 лет.

Создание самозаверяющих подписывающих сертификатов и пары ключей

- ❖ В командной строке введите следующую команду (выполняемый файл ADT находится в каталоге bin пакета Flex SDK):

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

В этом примере используется минимальный набор атрибутов, которые могут быть установлены для сертификата. Необходимо использовать ключ *1024-RSA* или *2048-RSA* (см. раздел «Команда ADT certificate» на странице 186).

Создание пакета AIR

- ❖ Из командной строки выполните следующую команду (вводится одной строкой):

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

Будет выдано предложение ввести пароль для файла ключей. Введите пароль и нажмите клавишу Enter.

Дополнительные сведения см. в разделе «[Упаковка приложения AIR для мобильных устройств](#)» на странице 99».

Установка среды выполнения AIR

Последнюю версию среды выполнения AIR на устройство можно установить с Android Маркета. Кроме того, можно установить включенную в SDK среду выполнения на устройство или в эмулятор Android.

- ❖ Из командной строки выполните следующую команду (вводится одной строкой):

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

В качестве значения флага `-platformsdk` укажите путь к Android SDK (родительский каталог папки tools).

ADT устанавливает файл `Runtime.apk`, включенный в SDK.

Дополнительные сведения см. в разделе «[Установка среды выполнения AIR и приложений для разработки](#)» на странице 115».

Установка приложения AIR

- ❖ Из командной строки выполните следующую команду (вводится одной строкой):

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

В качестве значения флага `-platformsdk` укажите путь к Android SDK (родительский каталог папки tools).

Дополнительные сведения см. в разделе «[Установка среды выполнения AIR и приложений для разработки](#)» на странице 115».

Чтобы запустить приложение, нажмите на значок приложения на экране устройства или эмулятора.

Глава 6. Разработка приложений AIR для настольных систем

Процедура разработки приложений AIR для настольных систем

При создании приложений AIR применяется такая же базовая процедура, как в традиционных моделях разработки: кодирование, компиляция, тестирование, а в завершении цикла выполняется упаковка для создания файла упаковщика.

Для создания кода приложения используется Flash, Flex и ActionScript. Компиляция выполняется с помощью Flash Professional, Flash Builder или компиляторов командной строки mxmhc и compc. Код приложения также можно создавать с помощью HTML и JavaScript. В этом случае этап компиляции можно пропустить.

Тестирование приложений AIR для настольных систем можно выполнить с помощью инструмента ADL, который позволяет запускать приложения без предварительной упаковки и установки. Все среды разработки Flash Professional, Flash Builder, Dreamweaver и Aptana предоставляют поддержку отладчика Flash. Инструмент отладки FDB также можно запустить вручную, если используется ADL с интерфейсом командной строки. ADL показывает ошибки и результаты выполнения трассировочных инструкций.

Все приложения AIR должны быть упакованы в файл установки. Рекомендуется использовать межплатформенный формат файлов AIR. Исключением являются следующие случаи.

- Необходимо получать доступ к зависящим от платформы API-интерфейсам, таким как класс NativeProcess.
- В приложении используются собственные расширения.

В таких случаях приложение AIR можно упаковать в исходный файл установщика конкретной платформы.

Приложения на основе SWF

- 1 Создайте код MXML или ActionScript.
- 2 Создайте нужные ресурсы, такие как файлы растровых значков.
- 3 Создайте дескриптор приложения.
- 4 Скомпилируйте код ActionScript.
- 5 Проверьте приложение.
- 6 Создайте пакет и подпишите файл в формате AIR с использованием цели *air*.

Приложения на основе HTML

- 1 Создайте код HTML и JavaScript.
- 2 Создайте нужные ресурсы, такие как файлы растровых значков.
- 3 Создайте дескриптор приложения.
- 4 Проверьте приложение.
- 5 Создайте пакет и подпишите файл в формате AIR с использованием цели *air*.

Создание исходных установщиков для приложений AIR

- 1 Создайте код (ActionScript или HTML и JavaScript).
- 2 Создайте нужные ресурсы, такие как файлы растровых значков.
- 3 Создайте дескриптор приложения, указав профиль *extendedDesktop*.
- 4 Скомпилируйте код ActionScript.
- 5 Проверьте приложение.
- 6 Упакуйте приложение на каждой целевой платформе с использованием цели *native*.

Примечание. На целевой платформе должен быть создан собственный установщик для этой платформы. Например, нельзя создать установщик Windows на платформе Mac. Виртуальную машину, например VMWare, можно использовать для выполнения нескольких платформ на одном и том же компьютерном оборудовании.

Создание приложений AIR с использованием связанного пакета среды выполнения

- 1 Создайте код (ActionScript или HTML и JavaScript).
- 2 Создайте нужные ресурсы, такие как файлы растровых значков.
- 3 Создайте дескриптор приложения, указав профиль *extendedDesktop*.
- 4 Скомпилируйте код ActionScript.
- 5 Проверьте приложение.
- 6 Упакуйте приложение на каждой целевой платформе с использованием цели *bundle*.
- 7 Создайте программу установки с использованием файлов пакета. (Пакет AIR SDK не предоставляет инструменты для создания таких установщиков, но для этого можно использовать множество наборов инструментов сторонних производителей.)

Примечание. На целевой платформе должен быть создан пакет для этой платформы. Например, нельзя создать пакет Windows на платформе Mac. Виртуальную машину, например VMWare, можно использовать для выполнения нескольких платформ на одном и том же компьютерном оборудовании.

Настройка свойств приложения для настольных систем

Настройте базовые свойства приложения в файле дескриптора приложения. В данном разделе описаны свойства, относящиеся к приложениям AIR для настольных систем. Подробное описание элементов файла дескриптора приложения представлено в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220.

Требуемая версия среды выполнения AIR

Укажите версию среды выполнения AIR, которая требуется для приложения, использующего пространство имен файла дескриптора приложения.

Пространство имен, назначенное элементом `application`, в большей степени определяет функции, которые используются в приложении. Например, если в приложении используется пространство имен AIR 1.5, а пользователь установил AIR 3.0, приложению будет доступно только поведение, соответствующее версии AIR 1.5 (даже если в AIR 3.0 поведение было изменено). Приложение получит доступ к новому поведению и функциям только после изменения пространства имен и публикации обновления. Безопасность и изменения WebKit являются основными исключениями из этого правила.

Укажите пространство имен с помощью атрибута `xmlns` корневого элемента `application`:

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

Дополнительные разделы справки

«[application](#)» на странице 226

Идентификация приложения

Некоторые настройки публикуемых приложений могут быть уникальными. К уникальным настройкам относятся идентификатор, название и имя файла.

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

Дополнительные разделы справки

«[id](#)» на странице 243

«[filename](#)» на странице 237

«[name](#)» на странице 251

Версия приложения

В версиях AIR до AIR 2.5 укажите элемент `version`. Можно использовать любую строку. Среда выполнения AIR не интерпретирует строку. Версия «2.0» не рассматривается как более поздняя версия по сравнению с «1.0».

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

В AIR 2.5 и более поздних версиях укажите версию приложения с помощью элемента `versionNumber`. Элемент `version` больше не используется. При указании значения для элемента `versionNumber` необходимо использовать последовательность из трех цифр, разделенных точками, например «0.1.2». Каждый сегмент номера версии может содержать до трех цифр (то есть максимальным номером версии может быть «999.999.999»). Номер не обязательно должен включать все три сегмента. Также допускается устанавливать версии «1» и «1.0».

Кроме того, можно задать метку для версии с помощью элемента `versionLabel`. Если добавлена метка версии, она отображается вместо номера версии, например в диалоговых окнах установщика приложения AIR.

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

Дополнительные разделы справки

«[version](#)» на странице 259

«[versionLabel](#)» на странице 260

«[versionNumber](#)» на странице 260

Свойства основного окна

Когда AIR запускает приложение в настольной системе, она создает окно и загружает в него основной SWF-файл или страницу HTML. AIR с помощью дочерних элементов `initialWindow` управляет изначальным видом и поведением данного окна приложения.

- **content** — основной SWF-файл приложения в дочернем элементе `content` элемента `initialWindow`. Если для устройства задан профиль настольной системы, можно использовать файл SWF или HTML.

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

Файл должен быть включен в пакет AIR (с помощью ADT или ИСР). Если только указать имя в дескрипторе приложения, это не позволит автоматически включить файл в пакет.

- **depthAndStencil** указывает на необходимость использования глубины или трафаретного буфера. Такой тип буферов обычно используется при работе с трехмерным содержимым.

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

- **height** — высота исходного окна.
- **maximizable** определяет, отображается ли системный хром для разворачивания окна на весь экран.
- **maxSize** — максимально разрешенный размер.
- **minimizable** определяет, отображается ли системный хром для сворачивания окна.
- **minSize** — минимально разрешенный размер.
- **renderMode** — в среде AIR 3 или более поздних версий для приложений для настольных систем можно установить режим визуализации *auto*, *cpu*, *direct* или *gpu*. В более ранних версиях AIR этот параметр игнорируется на настольных платформах. Параметр `renderMode` нельзя изменить во время выполнения.
 - *auto* — по существу совпадает с режимом *cpu*.
 - *cpu* — экранные объекты визуализируются и копируются в видеопамять в программном обеспечении. Класс `StageVideo` доступен только в полноэкранном режиме окна. `Stage3D` использует программное средство визуализации.
 - *direct* — экранные объекты визуализируются программным обеспечением среды выполнения, но копирование визуализированных кадров в видеопамять (блиторование) выполняется с аппаратным ускорением. Класс `StageVideo` доступен. `Stage3D` использует аппаратное ускорение, если при всех прочих условиях это возможно. Если для прозрачности окна установлено значение `true`, то для окна восстанавливается программная визуализация и блиторование.

***Примечание.** Чтобы использовать возможности графического ускорителя в содержимом Flash с AIR для мобильных платформ, компания Adobe рекомендует использовать параметр `renderMode="direct"` (т. е. `Stage3D`), а не `renderMode="gpu"`. Компания Adobe официально поддерживает и рекомендует среды разработки *Starling* (2D) и *Away3D* (3D). Дополнительные сведения о `Stage3D` и *Starling/Away3D* см. на странице <http://gaming.adobe.com/getstarted/>.*

- *gpu* — аппаратное ускорение используется, если этот режим доступен.

- **requestedDisplayResolution** — Должно ли приложение перейти в режим стандартного (*standard*) или высокого (*high*) разрешения на компьютерах MacBook Pro с экранами высокого разрешения. На всех прочих платформах это значение игнорируется. При значении *standard* (стандартное разрешение) каждый пиксел рабочей области занимает четыре пиксела на экране. При значении *high* (высокое разрешение) каждый пиксел рабочей области занимает один физический пиксел экрана. Указанное значение действительно для всех окон приложения. Использование элемента `requestedDisplayResolution` для настольных приложений AIR (в качестве дочернего элемента `initialWindow`) допускается в версии AIR 3.6 и более поздних.
- **resizable** определяет, отображается ли системный хром для изменения размера окна.
- **systemChrome** определяет, используется ли стандартное оформление окна операционной системы. Настройки окна `systemChrome` не могут быть изменены во время выполнения.
- **title** — заголовок окна.
- **transparent** определяет, применяется ли альфа-блендинг при наложении окна на фоновый рисунок. Окно не может использовать системный хром, если прозрачность включена. Параметр окна `transparent` не может быть изменен во время выполнения.
- **visible** определяет, является ли окно видимым сразу после его создания. По умолчанию окно является изначально невидимым, что позволяет приложению отобразить его содержимое, прежде чем оно станет видимым.
- **width** — ширина окна.
- **x** — положение окна по горизонтали.
- **y** — положение окна по вертикали.

Дополнительные разделы справки

«[content](#)» на странице 231

«[depthAndStencil](#)» на странице 233

«[height](#)» на странице 241

«[maximizable](#)» на странице 249

«[maxSize](#)» на странице 250

«[minimizable](#)» на странице 250

«[minimizable](#)» на странице 250

«[minSize](#)» на странице 250

«[renderMode](#)» на странице 253

«[requestedDisplayResolution](#)» на странице 254

«[resizable](#)» на странице 255

«[systemChrome](#)» на странице 257

«[title](#)» на странице 258

«[transparent](#)» на странице 259

«[visible](#)» на странице 260

«[width](#)» на странице 261

[«x»](#) на странице 261

[«y»](#) на странице 262

Функции настольной системы

Следующие элементы управляют функциями установки и обновления на настольных системах.

- `customUpdateUI` позволяет использовать собственные диалоговые окна для обновления приложения. Если по умолчанию установлено значение `false`, используются стандартные диалоговые окна AIR.
- `fileTypes` указывает типы файлов, при открытии которых по умолчанию будет запускаться данное приложение. Если для этого типа файлов уже настроено другое приложение по умолчанию, AIR не будет изменять существующие настройки. Однако приложение сможет переопределить регистрацию во время выполнения, используя метод `setAsDefaultApplication()` объекта `NativeApplication`. Хорошим тоном будет, если приложение будет запрашивать разрешение пользователя, прежде чем переопределять существующие ассоциации типов файлов.

***Примечание.** Регистрация типа файла пропускается при упаковке приложения в качестве связанного пакета среды выполнения (с использованием цели `-bundle`). Чтобы зарегистрировать данный тип файла, необходимо создать программу установки, выполняющую регистрацию.*

- `installFolder` — путь к стандартному каталогу установки приложения, в котором установлено приложение. С помощью этого параметра можно указать другую папку, а также сгруппировать несколько приложений в общей папке.
- `programMenuFolder` — определяет иерархию меню в меню Windows «Все программы». С помощью этого параметра несколько приложений можно разместить в одном меню. Если папка меню не указана, ярлык приложения добавляется в основное меню.

Дополнительные разделы справки

[«customUpdateUI»](#) на странице 232

[«fileTypes»](#) на странице 239

[«installFolder»](#) на странице 246

[«programMenuFolder»](#) на странице 252

Поддерживаемые профили

Если приложение предназначено только для настольных систем, можно предотвратить установку приложения на устройствах с другими профилями, исключив эти профили из списка поддерживаемых. Если приложение использует класс `NativeProcess` или собственное расширение, оно должно поддерживать профиль `extendedDesktop`.

Если элемент `supportedProfile` не указан в дескрипторе приложения, предполагается, что приложение поддерживает все определенные профили. Чтобы настроить поддержку в приложении только определенных профилей, укажите список профилей, разделив их пробелами:

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

Список классов `ActionScript`, поддерживаемых в профилях `desktop` и `extendedDesktop`, представлен в разделе [«Возможности различных профилей»](#) на странице 265.

Дополнительные разделы справки

«[supportedProfiles](#)» на странице 256

Необходимые собственные расширения

Приложения, поддерживающие профиль `extendedDesktop`, могут использовать собственные расширения.

В дескрипторе приложения объявите все собственные расширения, которые используются в этом приложении AIR. В следующем примере показан синтаксис, применяемый для указания требуемых собственных расширений:

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

Значение элемента `extensionID` совпадает со значением элемента `id` в файле дескриптора расширения. Файл дескриптора расширения — это файл XML с именем `extension.xml`. Он упакован в ANE-файл, получаемый от разработчика собственного расширения.

Значки приложения

В настольной системе значки, указанные в дескрипторе приложения, используются в качестве значков для файла приложения, ярлыка и меню программы. Значок приложения должен быть предоставлен в виде набора изображений PNG с размерами 16x16, 32x32, 48x48 и 128x128 пикселей. Укажите путь к файлам значков, используя элемент `icon` в файле дескриптора приложения:

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

Если значок нужного размера отсутствует, используется следующий по размеру значок, который уменьшается до нужного размера. Если не предоставлен ни один значок, используется системный значок по умолчанию.

Дополнительные разделы справки

«[icon](#)» на странице 242

«[imageNxN](#)» на странице 243

Игнорируемые настройки

Приложения для настольных систем игнорируют настройки приложения, которые применяются для мобильных профилей. Игнорируются следующие настройки:

- `android`
- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `iPhone`
- `renderMode` (в версиях, предшествующих AIR 3)

- requestedDisplayResolution
- softKeyboardBehavior

Отладка приложения AIR для настольной системы

Если приложение создается в интегрированной среде разработки, например Flash Builder, Flash Professional или Dreamweaver, инструменты отладки обычно встроены в среду. Для выполнения отладки достаточно запустить приложение в режиме отладки. Если используется ИСП, которая не поддерживает отладку, можно использовать AIR Debug Launcher (ADL) и Flash Debugger (FDB).

Дополнительные разделы справки

[De Monsters: Monster Debugger](#)

«[Отладка с помощью AIR HTML Introspector](#)» на странице 304

Запуск приложения с помощью ADL

С помощью ADL приложение AIR можно запустить без упаковки и установки. Передайте файл дескриптора приложения в ADL в качестве параметра, как показано в следующем примере (необходимо сначала скомпилировать код ActionScript в приложении):

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

ADL выводит трассировочные инструкции, исключения среды выполнения и ошибки анализа HTML на экран терминала. Если процесс FDB ожидает входящего подключения, ADL установит соединение с отладчиком.

Для отладки приложения AIR, в котором используются собственные расширения, можно также использовать инструмент ADL. Например:

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

Дополнительные разделы справки

«[AIR Debug Launcher \(ADL\)](#)» на странице 170

Печать трассировочных инструкций

Для печати трассировочных инструкций в консоли, используемой для запуска ADL, добавьте трассировочные инструкции в код с помощью функции `trace`.

Примечание. Если инструкции `trace()` не отображаются на консоли, убедитесь, что в файле `tt.cfg` не задано `ErrorReportingEnable` или `TraceOutputFileEnable`. Дополнительные сведения о том, где находится этот файл в зависимости от платформы, см. в документе [Редактирование файла tt.cfg](#).

Пример ActionScript:

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

Пример JavaScript:

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

В коде JavaScript можно использовать функции `alert()` и `confirm()` для отображения сообщений отладки из своего приложения. Кроме того, в консоль печатаются номера строк с синтаксическими ошибками и все необработанные исключения JavaScript.

Примечание. Чтобы применить префикс `air`, показанный в примере JavaScript, необходимо импортировать на страницу файл `AIRAliases.js`. Это файл находится внутри каталога `frameworks` в пакете AIR SDK.

Подключение к отладчику Flash Debugger (FDB)

Для отладки приложений AIR с помощью Flash Debugger запустите сеанс FDB и затем запустите приложение, используя ADL.

Примечание. В приложениях AIR на основе SWF исходные файлы ActionScript должны быть скомпилированы с флагом `-debug`. (Во Flash Professional, установите флажок «Разрешить отладку» в диалоговом окне «Параметры публикации».)

- 1 Запустите FDB. Программу FDB можно найти в каталоге `bin` пакета Flex SDK.

В консоли выводится запрос FDB: `<fdb>`

- 2 Выполните команду `run: <fdb>run [Enter]`

- 3 Запустите отладочную версию приложения в другой командной строке или оболочке:

```
adl myApp.xml
```

- 4 С помощью команд FDB задайте контрольные точки.

- 5 Введите: `continue [Enter]`

Если приложение AIR выполнено в виде SWF-файла, то отладчик управляет только выполнением кода ActionScript. Если приложение AIR выполнено в виде HTML-файла, то отладчик управляет только выполнением кода JavaScript.

Чтобы выполнить ADL без подключения к отладчику, включите параметр `-nodebug`:

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

Чтобы получить краткие сведения о командах FDB, выполните команду `help`:

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

Дополнительные сведения о командах FDB см. на странице [Использование команд отладчика из командной строки](#) в документации по Adobe Flex.

Упаковка файла установки AIR для настольных систем

Каждое приложение AIR должно как минимум иметь файл дескриптора и основной SWF- или HTML-файл. Прочие активы, устанавливаемые вместе с приложением, необходимо также упаковать в AIR-файл.

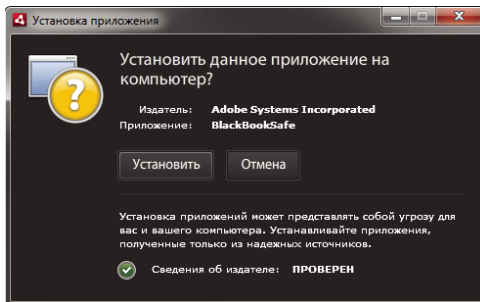
В данной статье описана процедура упаковки приложения AIR с помощью инструментов командной строки, включенных в SDK. Дополнительные сведения об упаковке приложения с использованием одного из инструментов Adobe можно найти в следующих статьях:

- Adobe® Flex® Builder™, см. веб-страницу [Упаковка приложений AIR в Flex Builder](#).
- Adobe® Flash® Builder™, см. веб-страницу [Упаковка приложений AIR в Flash Builder](#).

- Adobe® Flash® Professional, см. веб-страницу [Публикация для Adobe AIR](#).
- Adobe® Dreamweaver®, см. раздел [Создание приложения AIR в Dreamweaver](#).

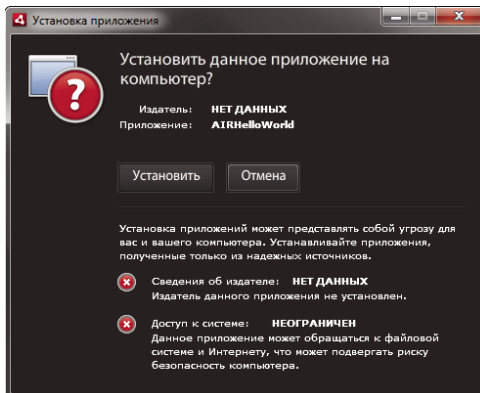
Все установочные файлы AIR необходимо подписывать цифровым сертификатом. Программа установки AIR использует подпись для гарантии того, что файл приложения не изменялся с момента подписания. Можно использовать сертификат для подписания кода, выданный уполномоченным центром сертификации, или применять самозаверяющий сертификат.

Сертификат, выданный сертифицирующим органом, вызовет у пользователей большее доверие к вам как к издателю. Во время установки отобразится диалоговое окно со сведениями о том, что удостоверение издателя проверено сертифицирующим органом.



Диалоговое окно подтверждения для приложения, подписанного доверенным сертификатом

При использовании самоподписанного сертификата пользователи не смогут проверить ваше удостоверение издателя. Кроме этого, пользователь не сможет быть уверен в том, что содержимое пакета не было изменено. (Исходный установочный файл может быть подменен до того, как конечный пользователь откроет его.) Во время установки отобразится диалоговое окно со сведениями о том, что удостоверение издателя не может быть проверено. Устанавливая такие приложения пользователи подвергают систему безопасности большому риску:



Диалоговое окно подтверждения для приложения с самоподписанным сертификатом

Файл AIR можно упаковать и подписать в один этап с помощью команды `-package` средства ADT. Также можно создать промежуточный, неподписанный пакет с помощью команды `-prepare` и отдельно подписать промежуточный пакет с помощью команды `-sign`.

Примечание. Java 1.5 и более поздние версии не позволяют использовать в паролях для защиты файлов сертификатов PKCS12 нестандартные символы ASCII. При создании или экспорте файла сертификата кода используйте в пароле только стандартные символы ASCII.

При подписании установочного пакета ADT автоматически связывается с сервером органа, поставившего временную отметку, для ее проверки. Информация о временной отметке включена в файл AIR. Файл AIR, содержащий проверенную временную отметку, в будущем может быть установлен в любой момент. Если средство ADT не может связаться с сервером временных отметок, упаковка не производится. Эту функцию можно отменить, но без временной отметки приложение AIR нельзя будет установить после того, как истечет срок действия сертификата подписи.

При создании пакета для обновления существующего приложения AIR он должен быть подписан тем же сертификатом, что и исходное приложение. Если исходный сертификат за последние 180 дней обновлялся или истекал, или при необходимости замены сертификата на новый можно использовать подпись переноса. При использовании подписи переноса AIR-файл приложения подписывается как старым, так и новым сертификатом. Используйте команду `-migrate`, чтобы применить подпись переноса, как описано в разделе «Команда ADT migrate» на странице 185.

Важная информация. По окончании срока действия сертификата и по прошествии 180-дневного льготного периода применить подпись переноса невозможно. Если не выполнить перенос сертификата, пользователи должны будут удалить текущую версию приложения перед установкой новой. Льготный период применяется только к приложениям, для которых в пространстве имен дескриптора приложения указана версия AIR 1.5.3 или более поздняя версия. Обратите внимание, что льготный период не предусмотрен в более ранних версиях среды выполнения AIR.

В версиях, предшествующих AIR 1.1, поддержка подписей переноса отсутствует. Для применения подписи переноса необходимо создать пакет приложения с помощью SDK версии 1.1 или более поздней версии.

Приложения, развертываемые с помощью файлов AIR, называются приложениями с профилем настольного компьютера. Нельзя использовать ADT для упаковки исходного установщика для приложения AIR, если файл дескриптора приложения не поддерживает профиль настольного компьютера. Этот профиль можно запретить путем использования элемента `supportedProfiles` в файле дескриптора приложения. См. раздел «Профили устройств» на странице 263 и «`supportedProfiles`» на странице 256.

Примечание. Параметры в файле дескриптора определяют идентификационные данные приложения AIR и путь установки по умолчанию. См. раздел «Файлы дескриптора приложения AIR» на странице 220.

Идентификаторы издателя

Начиная с версии AIR 1.5.3, идентификатор издателя не используется. В новых приложениях (изначально опубликованных с помощью AIR 1.5.3 или более поздней версии) не требуется и не должен использоваться идентификатор издателя.

При обновлении приложений, опубликованных с помощью более ранних версий AIR, необходимо указать исходный идентификатор издателя в файле дескриптора приложения. В противном случае установленная версия приложения и его обновленная версия будут считаться различными приложениями. Если используется другой идентификатор издателя (или если он не используется), пользователи должны будут удалить текущую версию приложения перед установкой новой.

Чтобы определить исходный идентификатор издателя, найдите файл `publisherid` в подкаталоге META-INF/AIR, в котором установлено исходное приложение. Строка в этом файле является идентификатором издателя. Чтобы указать идентификатор издателя вручную, в дескрипторе приложения должна быть указана среда выполнения AIR 1.5.3 (или более поздней версии) в объявлении пространства имен файла дескриптора приложения.

Для приложений, опубликованных с помощью версии AIR, предшествующей 1.5.3, или опубликованных с помощью AIR 1.5.3 SDK, при указании в пространстве имен дескриптора приложения более ранней версии AIR — идентификатор издателя вычисляется на основе сертификата подписи. Этот идентификатор используется совместно с идентификатором приложения для определения подлинности приложения. При наличии идентификатора издателя он используется в следующих целях:

- Проверка того, является ли устанавливаемый файл AIR обновлением или новым приложением
- как часть ключа шифрования для зашифрованного локального хранилища;
- как часть пути к каталогу хранения приложения;
- как часть строки для локальных соединений;
- как часть строки учетных данных, используемой для вызова приложения в API-интерфейсе обозревателя AIR;
- как часть идентификатора OSID (используется при создании пользовательских программ установки/удаления).

В версиях, предшествующих AIR 1.5.3, идентификатор издателя приложения мог изменяться, если обновление для приложения было подписано подписью переноса с использованием нового или обновленного сертификата. При изменении идентификатора издателя также изменяется поведение функций AIR, зависящих от идентификатора. Например, данные в существующем зашифрованном локальном хранилище становятся недоступными, и в любых экземплярах Flash или AIR, создающих локальное соединение с приложением, в строке подключения должен использоваться новый идентификатор.

В AIR 1.5.3 или более поздней версии идентификатор разработчика не связан с сертификатом подписи и присваивается только в том случае, если в дескрипторе приложения содержится тег `publisherID`. Приложение невозможно обновить, если указанный в пакете обновления AIR идентификатор издателя не совпадает с текущим идентификатором издателя.

Упаковка с помощью ADT

Упаковку приложения AIR можно выполнить с помощью инструмента командной строки AIR ADT. Перед упаковкой необходимо скомпилировать код ActionScript, MXML, а также все коды расширения. Кроме того, требуется сертификат для подписи кода.

Подробная информация о командах и параметрах ADT представлена в разделе «[AIR Developer Tool \(ADT\)](#)» на странице 176.

Создание пакета AIR

Чтобы создать пакет AIR, используйте команду ADT `package`, установив для сборок типа пакета *air*.

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

В данном примере предполагается, что путь к инструменту ADT указан в определении пути оболочки командной строки. (Инструкции см. в разделе «[Переменные среды Path](#)» на странице 327».)

Команда должна быть запущена из каталога с файлами приложения. В данном примере представлены файлы приложений `myApp-app.xml` (файл дескриптора приложения), `myApp.swf` и каталог значков.

Если приложение запускается, как показано в примере, ADT предложит ввести пароль для хранилища ключей (при вводе пароль не отображается, поэтому после завершения ввода требуется просто нажать Enter).

Создание пакета AIR на основе файла AIRI

Для создания устанавливаемого пакета подпишите файл AIRI:

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

Упаковка собственного установщика для настольной системы

Начиная с версии AIR 2, можно использовать ADT для создания собственных установщиков приложения для распространения приложений AIR. Например, можно создать EXE-файл установщика для установки приложения AIR в ОС Windows. Можно создать DMG-файл установщика для установки приложения AIR в ОС Mac OS. В AIR 2.5 и AIR 2.6 можно создать DEB- или RPM-файл установщика для установки приложения AIR в ОС Linux.

Приложения, устанавливаемые с помощью собственной программы установки, называются приложениями с профилем расширенного рабочего стола. Нельзя использовать ADT для упаковки собственного установщика для приложения AIR, если файл дескриптора приложения не поддерживает расширенный профиль настольного компьютера. Этот профиль можно запретить путем использования элемента `supportedProfiles` в файле дескриптора приложения. См. раздел «[Профили устройств](#)» на странице 263 и «[supportedProfiles](#)» на странице 256.

Создать версию собственного установщика приложения AIR можно двумя основными способами:

- Собственный установщик можно создать на основе файла дескриптора приложения и других исходных файлов. (В число других исходных файлов могут входить SWF-файлы, HTML-файлы и прочие активы.)
- Собственный установщик можно создать на основе AIR- или AIRI-файла.

Необходимо использовать ADT в той же ОС, для которой создается собственный файл установщика. Таким образом, чтобы создать EXE-файл для ОС, необходимо запустить ADT в ОС Windows. Чтобы создать DMG-файл для ОС Mac OS, необходимо запустить ADT в ОС Mac OS. Чтобы создать DEB- или RPM-файл для ОС Linux, необходимо запустить ADT из AIR 2.6 SDK в ОС Linux.

При создании собственного установщика для распространения приложения AIR оно приобретает следующие свойства.

- Оно может запускать и взаимодействовать с собственными процессами, используя класс `NativeProcess`.
Дополнительные сведения см. в следующих ресурсах.
 - [Взаимодействие с собственными процессами в AIR](#) (для разработчиков ActionScript)
 - [Взаимодействие с собственными процессами в AIR](#) (для разработчиков HTML)
- Оно может использовать собственные расширения.
- Оно может вызывать метод `File.openWithDefaultApplication()` для открытия любого файла в предназначенном для его открытия приложении независимо от его типа. (Существуют ограничения на приложения, которые *не* устанавливаются собственным установщиком. Дополнительные сведения см. в справочнике ActionScript® 3.0 для Adobe® Flash® Professional CS5 в разделе, посвященном методу `File.openWithDefaultApplication()`.)

Однако при упаковке в качестве собственного установщика теряются некоторые преимущества формата файлов AIR. Больше нельзя будет использовать один файл для распространения приложения на все настольные системы. Встроенная функция обновления (а также платформа обновления) не работает.

При двойном нажатии файла собственного установщика производится установка приложения AIR. Если на компьютере не установлена требуемая версия среды Adobe AIR, перед установкой приложения производится ее загрузка и установка. Если подключение к сети отсутствует, и загрузить требуемую версию Adobe AIR (если требуется) нельзя, установка прекращается. Установка также невозможна, если операционная система не поддерживается Adobe AIR 2.

***Примечание.** Если необходимо, чтобы файл в установленном приложении являлся исполняемым, то при создании пакета приложения убедитесь, что он является исполняемым в данной файловой системе. (В ОС Mac и Linux для установки флага executable можно использовать команду `chmod`.)*

Создание собственного установщика из исходных файлов приложения

Для создания собственного установщика из исходных файлов приложения используйте команду `-package` со следующим синтаксисом (в единой командной строке):

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

Этот синтаксис аналогичен синтаксису упаковки файла AIR (без собственного установщика). Однако существуют некоторые отличия:

- К команде добавляется параметр `-target native`. (Если указать параметр `-target air`, ADT создаст файл AIR вместо файла собственного установщика.)
- Целевой DMG- или EXE-файл необходимо указать в качестве `installer_file`.
- К тому же в ОС Windows можно добавить второй набор параметров подписи (см. `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` в примере синтаксиса). Помимо подписи файла AIR в ОС Windows можно подписать файл установщика Windows. Используйте тот же тип сертификата и синтаксис параметров подписи, что и для подписи файла AIR (см. раздел «[Параметры подписания кода ADT](#)» на странице 192»). Для подписи файла AIR и файла установщика можно использовать один и тот же или разные сертификаты. Когда пользователь загружает подписанный файл установщика Windows из Интернета, ОС Windows определяет источник файла по его сертификату.

Подробные сведения о параметрах ADT, кроме параметра `-target`, представлены в разделе «[AIR Developer Tool \(ADT\)](#)» на странице 176.

Ниже приведены примеры создания DMG-файла (файла собственного установщика в ОС Mac OS):

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

Ниже приведены примеры создания EXE-файла (файла собственного установщика в ОС Windows):

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

Ниже приведен пример создания и подписи EXE-файла:

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

Создание собственного установщика для приложения, в котором используются собственные расширения

Собственный установщик можно создать на основе исходных файлов для приложения и пакетов собственных расширений, необходимых в приложении. Используйте команду `-package` со следующим синтаксисом (в единой командной строке):

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

Синтаксис совпадает с синтаксисом, используемым для создания пакета собственного установщика, но имеет два дополнительных параметра. Используйте параметр `-extdir extension-directory` для указания каталога, в котором содержатся ANE-файлы (собственные расширения), используемые в приложении. Дополнительный флаг `-migrate` вместе с параметрами `MIGRATION_SIGNING_OPTIONS` служит для подписи обновленного приложения с помощью подписи переноса, в случае когда основной сертификат подписи отличен от сертификата, использованного для предыдущей версии. Дополнительные сведения см. в разделе «[Подписание обновленной версии приложения AIR](#)» на странице 213».

Дополнительные сведения о параметрах ADT см. в разделе «[AIR Developer Tool \(ADT\)](#)» на странице 176».

В следующем примере создается файл DMG (собственный файл установщика для ОС Mac OS X) для приложения, в котором используются собственные расширения:

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

Создание собственного установщика из файла AIR или AIRI

С помощью ADT можно создать собственный установщик на основе файла AIR или AIRI. Для создания собственного установщика на основе файла AIR используйте команду ADT `-package` со следующим синтаксисом (в единой командной строке):

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

Этот синтаксис аналогичен синтаксису для создания собственного установщика на основе исходных файлов приложений AIR. Однако существуют некоторые отличия.

- В качестве источника указывается файл AIR, а не файл дескриптора приложения и прочие исходные файлы приложения AIR.
- Так как файл AIR уже подписан, параметры подписи для него указывать не надо

Для создания собственного установщика на основе файла AIRI используйте команду ADT `-package` со следующим синтаксисом (в одной командной строке):

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

Этот синтаксис аналогичен синтаксису для создания собственного установщика на основе файла AIR. Однако существуют некоторые отличия.

- В качестве источника указывается файл AIRI.
- Необходимо указать параметры подписи для целевого приложения AIR.

Ниже приведен пример создания DMG-файла (файла собственного установщика в ОС Mac OS) на основе файла AIR:

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

Ниже приведен пример создания EXE-файла (файла собственного установщика в ОС Windows) на основе файла AIR:

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

Ниже приведен пример создания (на основе файла AIR) и подписи EXE-файла:

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

Ниже приведен пример создания DMG-файла (файла собственного установщика в ОС Mac OS) на основе файла AIRI:

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

Ниже приведен пример создания EXE-файла (файла собственного установщика в ОС Windows) на основе файла AIRI:

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

В следующем примере создается файл EXE (на основе файла AIR), который подписывается с использованием подписи AIR и исходной подписи Windows:

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

Упаковка связанных пакетов среды выполнения для настольных компьютеров

Связанный пакет среды выполнения — это пакет, который включает код приложения, а также специальную версию среды выполнения. Упакованное таким образом приложение использует включенную в пакет среду выполнения вместо общей среды выполнения, установленной на компьютере пользователя.

Созданный пакет является автономной папкой файлов приложения на платформе Windows и пакетом .app в ОС Mac OS. Пакет для целевой операционной системы необходимо создавать в этой операционной системе. (Виртуальную машину, например VMWare, можно использовать для выполнения нескольких операционных систем на одном компьютере.)

Приложение можно выполнить из этой папки или этого пакета без установки.

Преимущества

- Создание автономной папки
- Для установки не требуется подключение к Интернету
- Приложение изолировано от обновлений среды выполнения
- Организации могут сертифицировать определенную комбинацию приложения и среды выполнения
- Поддержка традиционной модели развертывания программного обеспечения
- Отдельное перераспределение среды выполнения не требуется
- Использование API-интерфейса NativeProcess
- Использование собственных расширений
- Использование функции `File.openWithDefaultApplication()` без ограничений
- Выполнение с USB-накопителя или оптического диска без установки

Недостатки

- Важные исправления системы защиты недоступны пользователям автоматически после публикации исправления системы защиты компанией Adobe
- Отсутствие поддержки файлов формата .air
- При необходимости требуется создавать собственный установщик
- API-интерфейс и среда обновления AIR не поддерживаются
- Встроенный в браузер API-интерфейс AIR для установки и запуска приложения AIR с веб-страницы не поддерживается
- В ОС Windows регистрация файла должна обрабатываться специальным установщиком
- Приложение занимает больше места на диске

Создание связанного пакета среды выполнения в ОС Windows

Чтобы создать связанный пакет среды выполнения для ОС Windows, необходимо создать пакет приложения в операционной системе Windows. Упакуйте приложение с использованием цели ADT *bundle*:

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

Эта команда создает пакет в каталоге myApp. В каталоге содержатся файлы приложения, а также файлы среды выполнения. Программу можно выполнить непосредственно из папки. Однако для создания записи меню для программы, регистрации типов файлов или обработчиков URI-схемы необходимо создать программу установки, задающую требуемые записи реестра. Пакет AIR SDK не содержит инструменты создания подобных установщиков, но для этого доступны некоторые компоненты сторонних производителей, включая коммерческие и бесплатные наборы инструментов для создания установщиков с открытым кодом.

Можно подписать собственный исполнимый файл в ОС Windows, указав второй набор параметров подписи после записи `-target bundle` в командной строке. Эти параметры подписи обозначают закрытый ключ и сертификат, используемые при применении собственной подписи Windows. (Обычно можно использовать сертификат цифровой подписи AIR.) Подписывается только главный исполнимый файл. Любые дополнительные исполнимые файлы, упакованные вместе с приложением, не подписываются при выполнении этого процесса.

Связь с типом файла

Чтобы связать приложение с общедоступными или пользовательскими типами файлов в ОС Windows, программа установки должна задавать соответствующие записи реестра. Типы файлов также должны быть перечислены в элементе `fileTypes` файла дескриптора приложения.

Дополнительные сведения о типах файлов Windows см. в статье [«Библиотека MSDN: связи типов файлов и файлов»](#)

Регистрация обработчика URI

Чтобы приложение поддерживало обработку запуска URL с использованием указанной URI-схемы, установщик должен задавать требуемые записи реестра.

Дополнительные сведения о регистрации приложения для обработки URI-схемы см. в статье [«Библиотека MSDN: Регистрация протокола URL для приложения»](#)

Создание связанного пакета среды выполнения в ОС Mac OS X

Чтобы создать связанный пакет среды выполнения для ОС Mac OS X, необходимо создать пакет приложения в операционной системе Macintosh. Упакуйте приложение с использованием цели ADT *bundle*:

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

Эта команда создает пакет приложения с именем myApp.app. В пакете содержатся файлы приложения, а также файлы среды выполнения. Приложение можно запустить, дважды щелкнув значок myApp.app и перетащив приложение в нужное расположение, например в папку «Программы». Однако для регистрации типов файлов или обработчиков URI-схем необходимо отредактировать файл списка свойств в пакете приложения.

В целях распространения можно создать файл образа диска (.dmg). Пакет Adobe AIR SDK не включает инструменты создания файла dmg для связанного пакета среды выполнения.

Связь с типом файла

Чтобы связать приложение с общедоступными или пользовательскими типами файлов в ОС Mac OS X, необходимо отредактировать файл info.plist в пакете для задания свойства CFBundleDocumentTypes. См. веб-страницу [«Библиотека разработчика Mac OS X: справочник по ключам информационного списка свойств, CFBundleURLTypes»](#).

Регистрация обработчика URI

Чтобы приложение поддерживало запуск URL с использованием указанной URI-схемы, необходимо отредактировать файл info.plist в пакете для задания свойства CFBundleURLTypes. См. веб-страницу [«Библиотека разработчика Mac OS X: справочник по ключам информационного списка свойств, CFBundleDocumentTypes»](#).

Распространение пакетов AIR для настольных систем

Приложения AIR можно распространять в виде пакета AIR, в который включен код приложения и все ресурсы. Этот пакет может распространяться любыми привычными средствами: загружаться, отправляться по электронной почте, записываться на компакт-диск. Для установки пользователю нужно дважды щелкнуть по файлу AIR. Используя API-интерфейс AIR, встроенный в браузер (библиотека ActionScript с веб-интерфейсом), можно предоставить пользователям возможность устанавливать приложение AIR (и Adobe® AIR® при необходимости) по нажатию на одну ссылку на веб-странице.

Приложения AIR можно упаковывать и распространять в виде исходных установщиков (то есть в виде файлов EXE в Windows, файлов DMG в Mac и файлов DEB или RPM в Linux). Исходные файлы установщика можно распространять и устанавливать в соответствии со стандартами платформы. При распространении приложения в виде исходного пакета теряются некоторые преимущества формата AIR. В числе прочего больше нельзя использовать один файл для большинства платформ, использовать платформу обновления AIR, а также API-интерфейс, встроенный в браузер.

Установка и запуск приложения AIR в настольной системе

Файл AIR можно просто отправить получателю. Например, файл AIR может быть отправлен как вложение в письме или в виде ссылки на веб-страницу.

После загрузки приложения AIR пользователь может установить его, следуя приведенным ниже инструкциям.

- 1 Дважды щелкните по файлу AIR.

К этому моменту на компьютере должна быть установлена среда Adobe AIR.

- 2 В окне установки оставьте все параметры по умолчанию и нажмите кнопку «Продолжить».

В ОС Windows AIR автоматически выполняет следующее:

- устанавливает приложение в каталог Program Files;
- создает ярлык приложения на рабочем столе;
- создает ярлык в меню «Пуск»;
- записывает приложение в раздел панели управления «Установка и удаление программ».

В Mac OS приложение по умолчанию добавляется в каталог Applications.

Если приложение уже установлено, программа установки предлагает открыть существующую версию или обновить приложение до версии загруженного файла AIR. Программа установки находит приложение по его ID и ID издателя в файле AIR.

3 После завершения установки нажмите кнопку «Готово».

В Mac OS для установки новой версии приложения пользователю потребуются соответствующие системные привилегии. В Windows и Linux пользователю нужны права администратора.

Также приложение может быть обновлено до новой версии с помощью ActionScript или JavaScript. Дополнительные сведения см. в разделе «[Обновление приложений AIR](#)» на странице 277.

После установки приложения AIR пользователь просто дважды щелкает по значку приложения, и оно запускается.

- В Windows дважды щелкните по значку приложения (на рабочем столе или в папке) или выберите приложение в меню «Пуск».
- В Linux дважды щелкните значок приложения (на рабочем столе или в папке) или выберите приложение в меню программ.
- В Mac OS дважды щелкните по приложению в папке, куда оно было установлено. По умолчанию приложения устанавливаются в каталог /Applications.

***Примечание.** В ОС Linux можно устанавливать только приложения AIR, разработанные для AIR 2.6 или более ранних версий.*

Непрерывная установка позволяет установить приложение AIR одним щелчком по ссылке на веб-странице. *Вызов из обозревателя* позволяет запустить установленное приложение AIR одним щелчком по ссылке на веб-странице. Эти функции будут рассмотрены в следующем разделе.

Установка и запуск приложений AIR для настольной системы с веб-страницы

API-интерфейс AIR, встроенный в браузер, позволяет устанавливать и запускать приложения AIR с веб-страницы. API-интерфейс AIR, встроенный в браузер, включен в библиотеку SWF *air.swf*, которая предоставлена компанией Adobe. SDK AIR содержит пример приложения *badge*, которое использует эту библиотеку для установки, обновления или запуска приложения AIR (и при необходимости среды выполнения). Пример приложения *badge* можно изменить или создать собственное веб-приложение, которое будет использовать библиотеку *air.swf*.

Любое приложение AIR может быть установлено с помощью значка веб-страницы. Однако с помощью значка веб-страницы можно запускать только приложения, в дескриптор которых включен элемент `<allowBrowserInvocation>true</allowBrowserInvocation>`.

Дополнительные разделы справки

«AIR.SWF, встроенный в браузер интерфейс API» на странице 268

Корпоративное развертывание на настольные компьютеры

ИТ-администраторы могут выполнять «тихую» установку среды выполнения и приложений AIR с помощью стандартных инструментов развертывания. ИТ-администраторы могут:

- выполнить «тихую» установку среды выполнения Adobe AIR с помощью таких инструментов, как Microsoft SMS, IBM Tivoli, или любого другого инструмента, поддерживающего «тихую» установку и использующего средства загрузки;
- выполнить «тихую» установку приложения AIR с помощью тех же инструментов, которые используются для развертывания среды выполнения.

Дополнительные сведения см. в [руководстве администратора Adobe AIR](http://www.adobe.com/go/learn_air_admin_guide_ru) (http://www.adobe.com/go/learn_air_admin_guide_ru).

Журналы установки на настольных компьютерах

Журналы установки сохраняются при установке среды выполнения AIR или приложения AIR. Журнал установки позволяет определить причину проблемы при установке или обновлении.

Файлы журналов создаются в следующих папках:

- ОС Mac: стандартный системный журнал (/private/var/log/system.log)
Чтобы просмотреть системный журнал Mac, откройте приложение Console (обычно находится в папке Utilities).
- Windows XP: C:\Documents and Settings\\Local Settings\Application Data\Adobe\AIR\logs\Install.log
- Windows Vista, Windows 7: C:\Users\\AppData\Local\Adobe\AIR\logs\Install.log
- Linux: /home/<username>/ .appdata/Adobe/AIR/Logs/Install.log

Примечание. Эти файлы журналов не создаются при использовании версий, предшествующих AIR 2.

Глава 7. Разработка приложений AIR для мобильных устройств

Приложения AIR развертываются на мобильных устройствах как собственные приложения. Они используют формат приложений устройства, не формат файлов AIR. В настоящее время AIR поддерживает пакеты Android APK и пакеты iOS IPA. После создания версии выпуска приложение можно распространять, используя стандартный механизм платформы. Для Android в общем случае — это Android Market, для iOS — Apple App Store.

Создавать приложения AIR для мобильных устройств можно с помощью [AIR SDK](#) и Flash Professional, Flash Builder, а также других инструментов разработки ActionScript. Приложения AIR для мобильных устройств на основе HTML в настоящее время не поддерживаются.

***Примечание.** Research In Motion (RIM) BlackBerry Playbook обеспечивает собственный пакет SDK для разработки приложений AIR. Дополнительные сведения о разработке Playbook см. на веб-странице [RIM: BlackBerry Tablet OS Development](#).*

***Примечание.** В настоящем документе описана процедура разработки приложений iOS с помощью AIR 2.6 SDK или более поздней версии. Приложения, созданные с помощью среды AIR 2.6 или более поздней версии, можно установить на устройствах iPhone 3Gs, iPhone 4 и iPad, работающих под управлением ОС iOS 4 и более поздних версий. Для создания приложений AIR для более ранних версий iOS необходимо использовать AIR 2 Packager для iPhone, как описано в разделе «Создание приложений для iPhone».*

Дополнительные сведения о методах обеспечения конфиденциальности см. в [руководстве по обеспечению конфиденциальности Adobe AIR SDK](#).

Полный список системных требований для выполнения приложений AIR см. в публикации [Системные требования Adobe AIR](#).

Настройка среды разработки

При установке приложений на мобильные платформы помимо настройки обычной среды разработки AIR, Flex и Flash требуется выполнить дополнительную настройку (дополнительные сведения по настройке базовой среды разработки AIR см. в разделе «[Инструменты Adobe Flash Platform для разработки приложений AIR](#)» на странице 17).

Настройка Android

Для Android среда выполнения AIR 2.6 и более поздних версий, как правило, не требует специальной настройки. Инструмент Android ADB включен в пакет AIR SDK (в папке lib/android/bin). AIR SDK использует инструмент ADB для установки, удаления и запуска пакетов приложений на устройстве. Кроме того, с помощью ADB можно просматривать системные журналы. Чтобы создать и запустить эмулятор Android, необходимо загрузить отдельный Android SDK.

Если приложение добавляет в элемент `<manifestAdditions>` дескриптора приложения такие элементы, которые текущая версия AIR распознает как недопустимые, необходимо установить более новую версию Android SDK. Задайте в качестве значения переменной среды `AIR_ANDROID_SDK_HOME` или параметра командной строки `-platformsdk` путь к файлу SDK. Инструмент упаковки AIR, ADT, использует этот SDK для проверки записей в элементе `<manifestAdditions>`.

В AIR 2.5 необходимо загрузить отдельную копию Android SDK с Google. С помощью переменной среды `AIR_ANDROID_SDK_HOME` можно настроить путь к папке Android SDK. Если эта переменная среды не установлена, путь к Android SDK необходимо указать с помощью аргумента `-platformsdk` в командной строке ADT.

Дополнительные разделы справки

[«Переменные среды ADT»](#) на странице 201

[«Переменные среды Path»](#) на странице 327

Настройка ОС iOS

Для установки и тестирования приложения для iOS на устройстве и распространения этого приложения требуется присоединиться к программе Apple iOS Developer (участие в программе предусматривает оплату). После присоединения к программе iOS Developer вам будет открыт доступ к portalу iOS Provisioning Portal, на котором можно получить файлы и ресурсы от Apple, необходимые для установки приложения на устройстве, проведения тестирования и последующего распространения. Предоставляется доступ к следующим файлам и ресурсам:

- Сертификаты разработки и распространения
- Идентификаторы приложений
- Файлы для разработки и распространения

Принципы разработки мобильных приложений

Рабочая среда и физические характеристики мобильных устройств требуют особо внимательного подхода к программированию и проектированию. Например, крайне важно упростить код, чтобы приложение работало с максимальной скоростью. Однако возможности оптимизации кода ограничены. Интеллектуальное проектирование в рамках ограничений устройства также помогает предотвратить чрезмерную нагрузку на систему визуализации.

Код

Оптимизировать код для повышения скорости работы приложения всегда полезно. При этом на большинстве мобильных устройств процессор работает медленнее, поэтому написание более тонкого кода приносит большие результаты. Кроме того, мобильные устройства практически всегда работают от аккумуляторов. Достижение тех же самых результатов при меньших усилиях требует меньшего расхода энергии.

Дизайн

При разработке приложения необходимо учитывать такие факторы, как меньший размер экрана, режим взаимодействия с пользователем посредством сенсорного экрана, а также постоянное изменение окружения, в котором работают пользователи.

Код и дизайн

Если в приложении применяется анимация, оптимизация визуализации имеет крайне важное значение. Однако зачастую недостаточно оптимизировать только код. Необходимо продумать все визуальные аспекты приложения, чтобы отрисовка приложения осуществлялась максимально эффективно.

В руководстве [Оптимизация производительности для платформы Flash Platform](#) представлены важные техники оптимизации. Данные техники относятся ко всем типам содержимого Flash и AIR, и их применение позволит гарантировать, что создаваемые приложения будут хорошо работать на мобильных устройствах.

- Пол Трани (Paul Trani). [Советы и приемы по разработке мобильного содержимого Flash](#)
- roguish: [Программа для тестирования графического процессора в приложениях AIR for Mobile](#)
- Джонатан Кампос (Jonathan Campos): [Методы оптимизации приложений AIR for Android](#)
- Чарльз Шульц (Charles Schulze). [Разработка игр AIR 2.6 для ОС iOS](#)

Цикл разработки приложений

Когда приложение теряет фокус, AIR снижает частоту кадров до 4 кадров в секунду и останавливает процесс визуализации графики. При более низкой частоте кадров происходит разрыв соединений с сетями потоковой передачи данных и сокетных соединений. Если в приложении такие подключения не используются, частоту кадров можно уменьшить еще больше.

При необходимости следует остановить воспроизведение и удалить прослушиватели для датчиков географического положения и акселерометров. Объект AIR NativeApplication отправляет события активации и отключения. Используйте эти события для управления переходами между активным и фоновым состоянием.

Большинство операционных систем без предупреждения завершают работу фоновых приложений. Если статус приложения сохраняется регулярно, приложение можно будет восстановить до приемлемого состояния после изменения статуса с фоновый на активный или при повторном запуске приложения.

Плотность информации

Экраны мобильных устройств имеют меньшие физические размеры, чем экраны настольных компьютеров, однако плотность пикселей (количество пикселей на дюйм) выше. При одном и том же размере шрифта буквы на экране мобильного устройства будут меньше, чем на экране компьютера. Часто необходимо использовать шрифт большего размера, чтобы текст был хорошо читаемым. В общем случае минимальный размер шрифта, который легко читается на экранах мобильных устройств, равен 14 пунктам.

Мобильные устройства часто используются на ходу и в условиях плохой освещенности. Учитывайте также то, какой объем информации реально можно отобразить на экране без ущерба удобочитаемости. Информации может поместиться меньше, чем на экране настольного компьютера с тем же разрешением.

Также следует помнить, что когда пользователь касается экрана, его пальцы и рука блокируют часть экрана. Разместите интерактивные элементы по краям и в нижней части экрана, если предполагается более долгое взаимодействие с ними, чем просто краткое нажатие.

Ввод текста

На многих устройствах для ввода текста используется виртуальная клавиатура. Виртуальные клавиатуры скрывают часть экрана и зачастую являются крайне громоздкими. Старайтесь избегать использования клавиатурных событий (за исключением программируемых клавиш).

Попробуйте найти альтернативы использованию текстовых полей для ввода. Например, чтобы пользователь мог ввести число, не требуется текстовое поле. Можно добавить две кнопки для увеличения или уменьшения значения.

Программируемые клавиши

На мобильных устройствах имеются программируемые клавиши, количество которых может отличаться для разных устройств. Это клавиши, для которых можно программным способом настроить выполнение разных функций. При использовании этих клавиш в приложении придерживайтесь требований соответствующей платформы.

Изменение ориентации экрана

Мобильное содержимое можно просматривать с использованием книжной или альбомной ориентации. Предусмотрите поведение приложения при изменении ориентации экрана. Дополнительные сведения см. на веб-странице [«Ориентация рабочей области»](#).

Затемнение экрана

При воспроизведении видео AIR не блокирует автоматически затемнение экрана устройства. Для управления переходом устройства в режим энергосбережения используйте свойство `systemIdleMode` объекта AIR `NativeApplication` (на некоторых платформах требуется запросить разрешение для работы этой функции).

Входящие звонки

Среда выполнения AIR автоматически приглушает звук, когда поступает входящий вызов или пользователь совершает звонок. В Android в дескрипторе приложения необходимо настроить разрешение `READ_PHONE_STATE`, если приложение должно воспроизводить звук при работе в фоновом режиме. В противном случае Android не позволяет среде выполнения обнаруживать телефонные вызовы и автоматически приглушать звук. См. раздел [«Разрешения Android»](#) на странице 81.

Мишени

Учитывайте размер мишени при разработке кнопок и других элементов интерфейса, которые должны нажимать пользователи. Эти элементы должны быть достаточно большими, чтобы их можно было удобно активировать с помощью пальца на сенсорном экране. Кроме того, между мишенями должно быть достаточно места. На стандартных экранах телефонов с высоким разрешением область мишени должна иметь размер около 44 пикселей на 57 пикселей с каждой стороны.

Размер установочного пакета приложения

На мобильных устройствах обычно доступно намного меньше пространства для установки приложений и хранения данных по сравнению с настольными системами. Удалите неиспользуемые ресурсы и библиотеки, чтобы максимально сократить размер пакета.

В Android при установке приложения извлечение отдельных файлов из пакета не выполняется. Вместо этого при обращении к ресурсам они распаковываются во временное хранилище. Чтобы максимально сократить размер хранилища ресурсов в распакованном состоянии, закройте файл и потоки URL после того, как ресурсы будут полностью загружены.

Доступ к файловой системе

В различных операционных системах действуют разные ограничения файловой системы, и такие ограничения обычно отличаются от ограничений, накладываемых операционными системами настольных компьютеров. Таким образом, место хранения файлов и данных может отличаться на разных платформах.

Одно из последствий отличия файловых систем заключается в том, что не всегда будут доступны ярлыки на общие папки, предоставляемые классом AIR File. Следующая таблица содержит ярлыки, которые можно использовать в Android и iOS.

	Android	iOS
File.applicationDirectory	Только чтение по URL (не исходный путь)	Только чтение
File.applicationStorageDirectory	Доступно	Доступно
File.cacheDirectory	Доступно	Доступно
File.desktopDirectory	Корневой каталог SD-карты	Недоступно
File.documentsDirectory	Корневой каталог SD-карты	Доступно
File.userDirectory	Корневой каталог SD-карты	Недоступно
File.createTempDirectory()	Доступно	Доступно
File.createTempFile()	Доступно	Доступно

Рекомендации Apple в отношении приложений iOS задают строгие правила по размещению файлов в различных ситуациях. К примеру, одна из рекомендаций гласит, что в каталоге, подлежащем резервному копированию на удаленное хранилище, должны размещаться только файлы, содержащие пользовательские или иные данные, которые невозможно восстановить или загрузить с сервера заново. Сведения о рекомендациях Apple в отношении резервного копирования и кэширования файлов приведена в разделе «Управление резервным копированием и кэшированием файлов».

Компоненты пользовательского интерфейса

Компанией Adobe разработана специальная версия Flex, оптимизированная для мобильных платформ. Дополнительные сведения об этом см. в разделе [Разработка мобильных приложений с использованием Flex и Flash Builder](#).

Также доступны разработанные участниками сообщества компоненты, которые подходят для мобильных приложений. Вот некоторые из них:

- Джош Тыньяла (Josh Tynjala), [Элементы управления пользовательского интерфейса Feathers для Starling](#)
- Деррик Григг (Derrick Grigg) [версия Minimal Comps, поддерживающая схемы оформления](#)
- Тодд Андерсон (Todd Anderson), [компоненты as3flobile](#)

Ускоренная визуализация графики Stage3D

Начиная с AIR 3.2 профиль AIR для мобильных устройств поддерживает ускоренную визуализацию графики Stage 3D. API-интерфейсы [Stage3D](#) в ActionScript представляют собой API низкого уровня, использующие аппаратное ускорение с использованием графического процессора и обеспечивающие дополнительные возможности 2D- и 3D-визуализации. Эти API-интерфейсы низкого уровня предоставляют разработчикам возможность использования аппаратного ускорения графического процессора для значительного улучшения производительности. Также можно использовать игровые станции, которые поддерживают API-интерфейсы Stage3D в ActionScript.

Дополнительные сведения см. в документе [Gaming engines, 3D, and Stage 3D](#) (Игровые станции, 3D и Stage3D).

Сглаживание видео

Для повышения производительности сглаживание видео в среде AIR отключено.

Собственные функции

AIR 3.0+

Многие мобильные платформы предоставляют функции, которые пока не доступны через стандартный API-интерфейс AIR. Как и в AIR 3, вы можете расширить AIR, добавив свои библиотеки собственных кодов. Такие библиотеки расширений позволят получить доступ к функциям, доступным в операционных системах, или даже на определенных устройствах. Собственные расширения могут быть написаны на языке C в iOS и на Java или C в Android. Дополнительные сведения по разработке собственных расширений см. в разделе [Представление собственных расширений для Adobe AIR](#).

Процедура создания приложений AIR для мобильных устройств

В целом процедура создания приложений AIR для мобильных (или других) устройств очень схожа с процедурой разработки приложений для настольных систем. Основные отличия возникают на этапах упаковки, отладки и установки приложения. Например, приложения AIR for Android используют собственный формат пакетов Android APK вместо формата пакетов AIR. Соответственно, в этом случае применяется стандартный механизм установки и обновления Android.

AIR for Android

Типовыми этапами процесса разработки приложений AIR for Android являются следующие:

- Создание кода ActionScript или MXML.
- Создание файла дескриптора приложения AIR (с использованием пространства имен версии 2.5 или более поздней).
- Компиляция приложения.
- Упаковка приложения в пакет Android (.apk).
- Установка среды выполнения AIR на устройство или эмулятор Android (если используется внешняя среда выполнения; в AIR 3.7 и более поздних версиях по умолчанию используется связанная среда выполнения).
- Установка приложения на устройство (или в эмулятор Android).

- Запуск приложения на устройстве.

Для выполнения этих шагов можно использовать Adobe Flash Builder, Adobe Flash Professional CS5 или инструменты командной строки.

После того как приложение AIR будет создано и упаковано в файл APK, его можно будет разместить на Android Маркете или распространять другими способами.

AIR for iOS

Типовыми этапами процесса разработки приложений AIR for iOS являются следующие:

- Установка программы iTunes.
- Генерация требуемых файлов и идентификаторов разработчика на портале Apple iOS Provisioning Portal. К таким элементам относятся:
 - Сертификат разработчика
 - Идентификатор приложения
 - Профиль поставки

При создании профиля поставки необходимо включить в список идентификаторы всех тестовых устройств, на которые планируется установить приложение.

- Преобразование сертификата разработчика и закрытого ключа в файл хранилища ключей P12.
- Создание кода приложения ActionScript или MXML.
- Компиляция приложения с помощью компилятора ActionScript или MXML.
- Создайте изображения для значка и начального экрана приложения.
- Создание дескриптора приложения (с использованием пространства имен версии 2.6 или более поздней).
- Упаковка файла IPA с помощью ADT.
- С помощью iTunes добавьте профиль поставки на тестовое устройство.
- Выполните установку и тестирование приложения на устройстве с ОС iOS. Установить файл IPA можно с помощью программы iTunes или ADT через USB-подключение (поддерживается в AIR 3.4 и более поздних версий).

Когда разработка приложения AIR будет завершена, выполните повторную упаковку, используя сертификат распространения и профиль поставки. После этого его можно будет разместить в Apple App Store.

Настройка свойств мобильного приложения

Как и при разработке других приложений AIR в файле дескриптора приложения задаются базовые свойства приложения. Мобильные приложения игнорируют некоторые настройки, относящиеся к настольным системам, например размер окна и прозрачность. Кроме того, для мобильных приложений имеются свойства, относящиеся к отдельным платформам. Например, для приложений Android можно включить элемент `android`, а для приложений iOS — элемент `iPhone`.

Общие настройки

Некоторые настройки в дескрипторе приложения важны для всех приложений для мобильных устройств.

Требуемая версия среды выполнения AIR

Укажите версию среды выполнения AIR, которая требуется для приложения, использующего пространство имен файла дескриптора приложения.

Пространство имен, назначенное элементом `application`, в большей степени определяет функции, которые используются в приложении. Например, если в приложении используется пространство имен AIR 2.7, а пользователь установил более позднюю версию, приложение будет по-прежнему видеть поведение AIR 2.7 (даже если поведение в более поздней версии было изменено). Приложение получит доступ к новому поведению и функциям только после изменения пространства имен и публикации обновления. Исправления безопасности являются важным исключением из этого правила.

На устройствах, которые используют среду выполнения отдельно от приложения, например на Android с AIR 3.6 и более ранних версий, пользователю будет предложено установить или обновить AIR, если требуемая версия не установлена. На устройствах со встроенной средой выполнения, таких как iPhone, этого не происходит, так как требуемая версия включена в пакет приложения.

Примечание. (AIR 3.7 и более поздних версий) По умолчанию ADT добавляет среду выполнения в пакеты приложений Android.

Укажите пространство имен с помощью атрибута `xmlns` корневого элемента `application`. Для мобильных приложений необходимо использовать следующие пространства имен (в зависимости от целевой мобильной платформы):

```
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">
```

Примечание. Поддержку для устройств с ОС iOS 3 представляет *Packager* для iPhone SDK, построенный на базе AIR 2.0 SDK. Дополнительные сведения о создании приложений AIR для iOS 3 см. в разделе «Создание приложений для iPhone». Пакет AIR 2.6 SDK (и более поздних версий) поддерживает ОС iOS 4 и более поздних версий на устройствах iPhone 3Gs, iPhone 4 и iPad.

Дополнительные разделы справки

«[application](#)» на странице 226

Идентификация приложения

Некоторые настройки публикуемых приложений могут быть уникальными. К таким настройкам относятся идентификатор, имя и имя файла.

Идентификаторы приложений для Android

На устройствах Android для преобразования идентификатора в имя пакета Android к идентификатору AIR добавляется префикс «air.». Например, если имеется идентификатор AIR `com.example.MyApp`, пакет Android будет иметь имя `air.com.example.MyApp`.

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

Кроме того, если идентификатор не является надлежащим именем пакета в операционной системе Android, он преобразуется в надлежащее имя. Символы дефиса заменяются подчеркиваниями, а перед цифрами в начале идентификатора добавляется заглавная буква «А». Например, идентификатор `3-goats.1-boat` будет преобразован в имя пакета `air.A3_goats.A1_boat`.

***Примечание.** Префикс, добавляемый к идентификатору приложения, можно использовать для идентификации приложений AIR на Android Маркете. Если идентификация приложения AIR на основе префикса не требуется, необходимо распаковать файл APK, изменить идентификатор приложения и повторно упаковать его, как описано в статье [Отключение анализа приложений AIR for Android](#).*

Идентификаторы приложений для iOS

Установите идентификатор приложения AIR, соответствующий идентификатору приложения, который был создан на портале Apple iOS Provisioning Portal.

Идентификатор приложения iOS состоит из идентификатора начального числа пакета, за которым следует идентификатор пакета. Идентификатор начального числа пакета — это строка символов, например 5RM86Z4DJM, которую Apple присваивает идентификатору приложения. Идентификатор пакета содержит строку с обратным написанием доменного имени, которое выбирает разработчик. Идентификатор пакета может заканчиваться звездочкой (*), указывающей на подстановочный шаблон идентификатора. Если идентификатор пакета заканчивается подстановочным символом, его можно заменить на любую допустимую строку.

Например:

- Для идентификатора приложения Apple 5RM86Z4DJM.com.example.helloWorld в дескрипторе приложения следует использовать идентификатор com.example.helloWorld.
- Если приложение Apple имеет идентификатор 96LPVWEASL.com.example.* (идентификатор с подстановочным символом), можно использовать com.example.helloWorld, com.example.anotherApp, а также любой другой идентификатор, который начинается со строки com.example.
- Наконец, если идентификатор приложения Apple состоит только из идентификатора начального числа пакета и подстановочного шаблона, например 38JE93KJL.*, в AIR можно использовать любой идентификатор приложения.

При указании идентификатора приложения не включайте в него идентификатор начального числа пакета.

Дополнительные разделы справки

«id» на странице 243

«filename» на странице 237

«name» на странице 251

Версия приложения

В AIR 2.5 и более поздних версиях укажите версию приложения с помощью элемента `versionNumber`. Элемент `version` больше не используется. При указании значения для элемента `versionNumber` необходимо использовать последовательность из трех цифр, разделенных точками, например «0.1.2». Каждый сегмент номера версии может содержать до трех цифр (то есть максимальным номером версии может быть «999.999.999»). Номер не обязательно должен включать все три сегмента. Также допускается устанавливать версии «1» и «1.0».

Кроме того, можно задать метку для версии с помощью элемента `versionLabel`. Если добавлена метка версии, она отображается вместо номера версии, например в информационных окнах приложения Android. Метку версии необходимо установить для приложений, которые распространяются через Android Маркет. Если в дескрипторе приложения AIR значение элемента `versionLabel` не указано, для метки версии Android устанавливается значение элемента `versionNumber`.

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

На платформе Android номер версии AIR `versionNumber` преобразуется в целое число Android `versionCode` по формуле: $a \cdot 1000000 + b \cdot 1000 + c$, где a , b и c — это компоненты номера версии AIR: $a.b.c$.

Дополнительные разделы справки

«[version](#)» на странице 259

«[versionLabel](#)» на странице 260

«[versionNumber](#)» на странице 260

Основной SWF-файл приложения

Укажите основной SWF-файл приложения в дочернем элементе `content` элемента `initialWindow`. Если целевое устройство имеет мобильный профиль, необходимо использовать SWF-файл (приложения на основе HTML не поддерживаются).

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

Файл должен быть включен в пакет AIR (с помощью ADT или ИСП). Если только указать имя в дескрипторе приложения, это не позволит автоматически включить файл в пакет.

Свойства главного окна

Исходный вид и поведение главного окна приложения определяют несколько дочерних элементов элемента `initialWindow`.

- **aspectRatio** — указывает, будет ли приложение первоначально отображаться в *книжном* формате (высота больше ширины), *альбомном* формате (высота меньше ширины) или *любом* формате (с автоматической настройкой по всем ориентациям).

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

- **autoOrients** — указывает, изменяется ли ориентация рабочей области автоматически, когда пользователь поворачивает устройство или выполняет другую операцию, которая влияет на ориентацию, например открывает или закрывает выдвижную клавиатуру. Если для параметра установлено значение *false*, которое является значением по умолчанию, ориентация рабочей области не будет изменяться вместе с ориентацией устройства.

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

- **depthAndStencil** указывает на необходимость использования глубины или трафаретного буфера. Такой тип буферов обычно используется при работе с трехмерным содержимым.

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

- **fullScreen** — указывает, заполняет ли приложение всю область отображения на устройстве или на экране также отображаются системные элементы, например строка состояния.

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

- **renderMode** — указывает, использует ли среда выполнения графический или центральный процессор для визуализации приложения. В общем случае визуализация с помощью графического процессора повышает скорость визуализации, однако в этом режиме недоступны некоторые функции, например определенные режимы наложения и фильтры PixelBender. Кроме того, возможности и ограничения графического процессора отличаются для разных устройств и разных драйверов устройств. По возможности приложение следует протестировать на как можно большем количестве моделей устройств, особенно при использовании режима визуализации с помощью графического процессора.

Для режима визуализации можно установить значение *gpu*, *cpu*, *direct* или *auto*. По умолчанию установлено значение *auto*, при котором в настоящее время используется режим центрального процессора.

Примечание. Чтобы использовать возможности графического ускорителя в содержимом Flash с AIR для мобильных платформ, компания Adobe рекомендует использовать параметр `renderMode="direct"` (т. е. Stage3D), а не `renderMode="gpu"`. Компания Adobe официально поддерживает и рекомендует среды разработки Starling (2D) и Away3D (3D). Дополнительные сведения о Stage3D и Starling/Away3D см. на странице <http://gaming.adobe.com/getstarted/>.

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

Примечание. Не используйте `renderMode="direct"` для приложений, выполняемых в фоновом режиме.

Режим графического процессора имеет следующие ограничения:

- Платформа Flex не поддерживает режим визуализации с помощью графического процессора.
- Фильтры не поддерживаются.
- Наложения PixelBender и заполнения не поддерживаются.
- Не поддерживаются следующие режимы наложения: слой, альфа, стереть, перекрытие, жесткий свет, замена светлым и замена темным.
- Не рекомендуется использовать режим визуализации с помощью графического процессора в приложении, которое воспроизводит видео.
- При визуализации с помощью графического процессора текстовые поля некорректно перемещаются в видимую область, если открыта виртуальная клавиатура. Чтобы текстовое поле оставалось видимым, когда пользователь вводит текст, используйте свойство рабочей области `softKeyboardRect` и события клавиатуры, чтобы переместить текстовое поле в видимую область.
- Если графический процессор не может выполнить визуализацию объекта отображения, он не отображается совсем. Например, если к объекту отображения применяется фильтр, объект не показывается.

Примечание. Реализация режима работы графического процессора для ОС iOS в AIR 2.6 и более поздних версий существенно отличается от реализации в предыдущей версии AIR 2.0. Применяются другие критерии оптимизации.

Дополнительные разделы справки

«[aspectRatio](#)» на странице 229

«[autoOrients](#)» на странице 229

«[depthAndStencil](#)» на странице 233

«[fullScreen](#)» на странице 241

«[renderMode](#)» на странице 253

Поддерживаемые профили

Чтобы указать, какие профили устройств поддерживает приложение, можно добавить элемент `supportedProfiles`. Для мобильных устройств используйте профиль `mobileDevice`. Когда приложение запускается с помощью Adobe Debug Launcher (ADL), ADL в качестве активного профиля использует первый профиль из списка. Кроме того, при запуске ADL можно использовать `-profile`, чтобы выбрать определенный профиль из списка поддерживаемых. Если приложение работает во всех профилях, элемент `supportedProfiles` можно не указывать. В этом случае в качестве активного профиля по умолчанию ADL использует профиль настольной системы.

Чтобы настроить поддержку в приложении профилей мобильных устройств и настольной системы и протестировать приложение в профиле мобильных устройств, добавьте следующий элемент:

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

Дополнительные разделы справки

«[supportedProfiles](#)» на странице 256

«[Профили устройств](#)» на странице 263

«[AIR Debug Launcher \(ADL\)](#)» на странице 170

Необходимые собственные расширения

Приложения, поддерживающие профиль `mobileDevice`, могут использовать собственные расширения.

В дескрипторе приложения объявите все собственные расширения, которые используются в этом приложении AIR. В следующем примере показан синтаксис, применяемый для указания требуемых собственных расширений:

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

Значение элемента `extensionID` совпадает со значением элемента `id` в файле дескриптора расширения. Файл дескриптора расширения — это файл XML с именем `extension.xml`. Он упакован в ANE-файл, получаемый от разработчика собственного расширения.

Поведение виртуальной клавиатуры

Установите для элемента `softKeyboardBehavior` значение `none`, чтобы отключить функции автоматического панорамирования и изменение размера, с помощью которых среда выполнения обеспечивает, что поле, в которое вводится текст, остается видимым после открытия виртуальной клавиатуры. Если автоматическая функция отключена, эту роль берет на себя приложение, обеспечивая видимость области ввода текста или другого релевантного содержимого после вызова клавиатуры. Свойство рабочей области `softKeyboardRect` можно использовать вместе с событием `SoftKeyboardEvent`, чтобы определять, когда клавиатура открывается, и какую область она закрывает.

Для включения автоматического поведения установите для элемента значение `pan`:

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

Поскольку `pan` является значением по умолчанию, автоматическое поведение клавиатуры также можно включить, если не указывать элемент `softKeyboardBehavior`.

Примечание. В режиме визуализации с использованием графического процессора поведение `pan` не поддерживается.

Дополнительные разделы справки

«[softKeyboardBehavior](#)» на странице 255

[Stage.softKeyboardRect](#)

[SoftKeyboardEvent](#)

Настройки Android

На платформе Android можно использовать элемент дескриптора приложения `android`, чтобы добавить информацию в манифест приложения Android, который представляет собой файл свойств приложения, используемый в операционной системе Android. ADT автоматически генерирует файл `Manifest.xml` при создании пакета APK. AIR задает свойства значений, которые требуются для работы определенных функций. Все остальные свойства, настроенные в разделе `android` дескриптора приложения AIR, добавляются в соответствующий раздел файла `Manifest.xml`.

***Примечание.** Для большинства приложений AIR необходимо настроить разрешения Android, требуемые для приложения, с помощью элемента `android`. Однако в общем случае не требуется настраивать никакие другие свойства.*

Настроить можно только атрибуты, значения которых имеют строковый, целый и логический тип. Настройка ссылок на ресурсы в пакете приложения не поддерживается.

***Примечание.** Среда выполнения требует установки параметра «минимальная версия SDK» выше 14. Если требуется создать приложение только для более поздних версий, необходимо убедиться, что файл манифеста содержит строку `<uses-sdk android:minSdkVersion=" "></uses-sdk>` с указанием правильной версии.*

Зарезервированные настройки манифеста Android

AIR настраивает несколько параметров в созданном манифесте Android, чтобы обеспечить корректную работу приложения и среды выполнения. Можно определить следующие настройки:

Элемент `manifest`

Следующие атрибуты элемента `manifest` не доступны для настройки:

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

Элемент `activity`

Следующие атрибуты элемента `activity` не доступны для настройки:

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

Элемент `application`

Следующие атрибуты элемента `application` не доступны для настройки:

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

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

Разрешения Android

В соответствии с моделью безопасности Android требуется, чтобы приложения запрашивали разрешение на использование функций, связанных с безопасностью и конфиденциальностью. Эти разрешения необходимо указать при упаковке приложений. Они не могут быть изменены во время выполнения. При установке пользователем приложения операционная система Android сообщает, какие разрешения приложение запрашивает. Если требуемое для функции разрешение не запрашивается, операционная система Android может выдать исключение при попытке приложения получить доступ к функции, но исключение не гарантируется. Исключения передаются средой выполнения в приложение. В режиме безопасного возврата сообщения об ошибках, связанных с разрешениями, добавляются в системный журнал Android

В среде AIR разрешения Android задаются в дескрипторе приложения с помощью элемента `android`. При добавлении разрешений используется следующий формат (здесь `PERMISSION_NAME` — это имя разрешения Android):

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

Инструкции по использованию разрешений, настроенные с помощью элемента `manifest`, добавляются непосредственно в манифест Android.

Для использования функций AIR требуются следующие разрешения:

ACCESS_COARSE_LOCATION Разрешает приложению доступ к данным о местоположении, предоставляемым сетью WIFI или сотовой сетью, которые получаются через класс `Geolocation`.

ACCESS_FINE_LOCATION: Разрешает приложению доступ к данным GPS через класс `Geolocation`.

ACCESS_NETWORK_STATE и ACCESS_WIFI_STATE: Разрешает приложению доступ к сведениям о сети через класс `NetworkInfo`.

CAMERA: Разрешает приложению доступ к камере.

Примечание. Если запрашивается разрешение на использование функции камеры Android предполагает, что приложению также требуется камера. Если функция камеры является необязательной, добавьте в манифест элемент `uses-feature` для камеры и установите для атрибута `required` значение `false`. См. раздел «[Фильтры совместимости Android](#)» на странице 83».

INTERNET: Разрешает приложению выполнять запросы по сети. Также разрешает удаленную отладку.

READ_PHONE_STATE: Разрешает среде выполнения AIR приглушать звук во время телефонных вызовов. Данное разрешение необходимо настроить, если приложение воспроизводит видео в фоновом режиме.

RECORD_AUDIO: Разрешает приложению доступ к микрофону.

WAKE_LOCK и DISABLE_KEYGUARD Разрешает приложению предотвращать переход устройства в режим сна с помощью параметров класса SystemIdleMode.

WRITE_EXTERNAL_STORAGE: Разрешает приложению выполнять запись на внешнюю карту памяти устройства.

Например, чтобы настроить разрешения для приложения, которому требуются все доступные разрешения, в дескриптор приложения можно добавить следующий код:

```
<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>
```

Дополнительные разделы справки

[Безопасность и разрешения Android](#)

[Класс Android Manifest.permission](#)

Пользовательские схемы URI Android

Для запуска приложения AIR с веб-страницы или собственного приложения Android можно применять пользовательскую схему URI. Поддержка пользовательских URI реализована с помощью фильтров намерений, настроенных в манифесте Android, поэтому данный метод недоступен на других платформах.

Для применения пользовательского URI добавьте элемент intent-filter в раздел <android> дескриптора приложения. В следующем примере должны быть указаны оба элемента intent-filter. Измените инструкцию <data android:scheme="my-customuri" /> в соответствии со строкой URI для пользовательской схемы.

```
<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>
```

Фильтр намерений указывает операционной системе Android, что приложение доступно для выполнения определенного действия. В случае пользовательского URI это означает, что пользователь нажал на ссылку, использующую данную схему URI (и браузер не знает, как обработать это событие).

Когда приложение вызывается с помощью пользовательского URI, объект `NativeApplication` отправляет событие `invoke`. URL ссылки с параметрами запроса помещается в массив `arguments` объекта `InvokeEvent`. Можно настроить любое количество фильтров намерений.

***Примечание.** Ссылки в экземпляре `StageWebView` не поддерживают открытие URL-адресов, для которых используется пользовательская URI-схема.*

Дополнительные разделы справки

[Фильтры намерений Android](#)

[Действия и категории Android](#)

Фильтры совместимости Android

Операционная система Android использует количество элементов в файле манифеста приложения для определения того, совместимо ли приложение с данным устройством. Добавление этой информации в манифест не является обязательным. Если эти элементы не включены, приложение может быть установлено на любом устройстве Android. Однако оно не обязательно будет работать на всех устройствах Android. Например, приложение камеры будет не слишком полезно на телефоне без камеры.

Для фильтрации применяются следующие теги манифеста Android:

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

Приложения камеры

Если запрашивается разрешение на использование камеры в приложении, Android предполагает, что приложению требуются все функции камеры, включая автофокусировку и вспышку. Если приложению требуются не все функции камеры или камера является необязательной функцией, для камеры необходимо задать различные элементы `uses-feature`, чтобы указать, какие не являются обязательными. В противном случае пользователи, устройства которых не поддерживают какую-либо из функций или на которых нет камеры, не смогут найти ваше приложение на Android Маркете.

В следующем примере показано, как запросить разрешения для камеры и указать, что все функции камеры являются необязательными:

```
<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>
```

Приложения звукозаписи

Если запрашивается разрешение для записи звука, Android предполагает, что приложению требуется микрофон. Если функция запись звука является необязательной, можно добавить тег `uses-feature` и указать, что микрофон не требуется. В противном случае пользователи, на устройствах которых нет микрофона, не смогут найти ваше приложение на Android Маркете.

В следующем примере показано, как запросить разрешение на использование микрофона и указать, что наличие микрофона не является обязательным:

```
<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>
```

Дополнительные разделы справки

[Разработчики Android: совместимость Android](#)

[Разработчики Android: константы имен функций Android](#)

Расположение для установки

Чтобы разрешить установку или перенос приложения на внешнюю карту памяти, задайте для атрибута `installLocation` элемента Android `manifest` значение `auto` или `preferExternal`:

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

Операционная система Android не гарантирует, что приложение будет установлено во внешнюю память. Пользователь может переносить приложение из внутренней во внешнюю память и обратно с помощью системных настроек.

Даже когда приложение установлено во внешнюю память, кэш приложения и пользовательские данные, включая содержимое каталога `app-storage`, общие объекты и временные файлы, по-прежнему остаются во внутренней памяти. Чтобы не занимать слишком большой объем внутренней памяти, внимательно выбирайте данные, которые будут помещены в каталог хранилища приложения. Большие объемы данных следует сохранять на SD-карте, используя местоположения `File.userDirectory` или `File.documentsDirectory` (которые ссылаются на корневой каталог SD-карты на Android).

Включение проигрывателя Flash Player и других подключаемых модулей в объекте StageWebView

В ОС Android 3.0 и более поздних версий приложение должно включать аппаратное ускорение в элементе приложения Android, чтобы содержимое подключаемого модуля отображалось в объекте `StageWebView`. Чтобы включить визуализацию для подключаемого модуля, задайте атрибуту `android:hardwareAccelerated` элемента `application` значение `true`:

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

Глубина цвета

AIR 3+

В AIR 3 и более поздних версиях среда выполнения настраивает дисплей для визуализации цветов с глубиной 32 бита. В более ранних версиях AIR среда выполнения использует глубину цвета 16 бит. Чтобы указать среде выполнения на необходимость использования глубины цвета 16 бит, используйте элемент `<colorDepth>` дескриптора приложения.

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

При использовании глубины цвета 16 бит увеличивается скорость визуализации, но ухудшается точность цветопередачи.

Настройки iOS

Настройки, которые применяются только для устройств с ОС iOS, указываются в элементе `<iPhone>` дескриптора приложения. Элемент `iPhone` может иметь следующие дочерние элементы: `InfoAdditions`, `requestedDisplayResolution`, `Entitlements`, `externalSwfs` и `forceCpuRenderModeForDevices`.

Элемент `InfoAdditions` позволяет указывать пары «ключ-значение», которые добавляются в файл настроек `Info.plist` для приложения. Например, следующие значения задают стиль строки состояния приложения и указывают, что приложению не требуется постоянный доступ к Wi-Fi.

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

Настройки `InfoAdditions` заключены в тег `CDATA`.

Элемент `Entitlements` позволяет указать пары «ключ-значение», добавляемые в файл настроек приложения `Entitlements.plist`. Настройки `Entitlements.plist` предоставляют приложению доступ к различным функциям ОС iOS, например push-уведомлениям.

Дополнительные сведения о настройках `Info.plist` и `Entitlements.plist` см. в документации для разработчиков Apple.

Поддержка в iOS задач, выполняемых в фоновом режиме

AIR 3.3

Adobe AIR 3.3 и более поздних версий поддерживает многозадачность в iOS, обеспечивая выполнение определенных действий в фоновом режиме:

- Аудио
- Обновления сведений о расположении
- Компьютерные сети
- Отказ от работы приложения в фоновом режиме

Примечание. Для SWF версии 21 и более ранних AIR не поддерживает выполнение в фоновом режиме на платформах iOS и Android, когда выбран прямой режим визуализации. Из-за этого приложения на основе Stage3D не могут выполнять фоновые задачи, в частности, воспроизведение аудио, отправку местоположения, передачу данных по сети и пр. Системой iOS запрещены вызовы, обращенные к графическому интерфейсу OpenGL или визуализации в фоновом режиме. Работа приложений, предпринимающих попытку вызова OpenGL в фоновом режиме, прерывается системой iOS. Система Android разрешает приложениям выполнять как вызовы OpenGL в фоновом режиме, так и другие фоновые задачи (такие как воспроизведение аудио). Для SWF версии 22 или более поздней мобильные приложения AIR могут выполняться в фоновом режиме, когда выбран прямой режим визуализации. В среде выполнения AIR iOS

возникает ошибка `ActionScript (3768 — API-интерфейс Stage3D нельзя использовать при выполнении в фоновом режиме)`, если вызовы `OpenGLES` отправляются в фоновом режиме. Однако на платформе `Android` ошибки не возникают, так как ее собственным приложениям разрешается вызывать `OpenGLES` в фоновом режиме. Чтобы обеспечить оптимальное использование ресурсов мобильных устройств, не вызывайте методы визуализации, пока приложение работает в фоновом режиме.

Воспроизведение и запись аудио в фоновом режиме

Чтобы включить воспроизведение и запись аудио в фоновом режиме, добавьте следующую пару «ключ-значение» в элемент `InfoAdditions`:

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

Обновления сведений о расположении в фоновом режиме

Чтобы включить обновления сведений о расположении в фоновом режиме, добавьте следующую пару «ключ-значение» в элемент `InfoAdditions`:

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

Примечание. Используйте данную функцию только при необходимости, так как API-интерфейсы определения расположения в немалой степени расходуют заряд батареи.

Работа с сетями в фоновом режиме

Для выполнения несложных задач в фоновом режиме приложение задает свойство `NativeApplication.nativeApplication.executeInBackground` равным `true`.

Например, приложение может начать операцию отправки файла, после которой пользователь переводит другое приложение в активный режим. Когда приложение получает сведения о событии отправки, оно может задать свойство `NativeApplication.nativeApplication.executeInBackground` равным `false`.

Настройка свойства `NativeApplication.nativeApplication.executeInBackground` со значением `true` не гарантирует, что приложение будет работать неопределенно долго, так как `iOS` накладывает ограничение по времени выполнения задач в фоновом режиме. Когда `iOS` останавливает обработку в фоновом режиме, AIR отправляет событие `NativeApplication.suspend`.

Отказ от работы в фоновом режиме

Приложение может явным образом отклонить выполнение в фоновом режиме, добавив следующую пару «ключ-значение» в элемент `InfoAdditions`:

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

Зарезервированные настройки iOS InfoAdditions

AIR настраивает несколько параметров в созданном файле Info.plist, чтобы обеспечить корректную работу приложения и среды выполнения. Можно определить следующие настройки:

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

Примечание. Можно определить *MinimumOSVersion*. Определение *MinimumOSVersion* учитывается в Air 3.3 и более поздних версий.

Поддержка разных моделей устройств с ОС iOS

Для поддержки iPad добавьте корректные настройки пар «ключ-значение» для `UIDeviceFamily` в элемент `InfoAdditions`. Параметр `UIDeviceFamily` является массивом строк. Каждая строка определяет поддерживаемые устройства. Параметр `<string>1</string>` определяет поддержку iPhone и iPod Touch. Параметр `<string>2</string>` определяет поддержку iPad. Параметр `<string>3</string>` определяет поддержку tvOS. Если указать только одну из этих строк, будет поддерживаться только одно семейство устройств. Например, следующий параметр означает, что поддерживается только iPad:

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

Следующий параметр поддерживает оба семейства устройств (iPhone/iPod Touch и iPad):

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

Кроме того, в AIR 3.7 и более поздних версиях, можно использовать тег `forceCPURenderModeForDevices` для принудительного включения режима визуализации с использованием ЦП для указанного набора устройств и включения режима визуализации с использованием графического процессора для остальных устройств iOS.

Этот тег необходимо добавить к тегу `iPhone` в качестве дочернего тега и указать перечень моделей устройств, разделенный пробелами. Список допустимых наименований моделей устройств см. в описании [«forceCPURenderModeForDevices»](#) на странице 240.

Например, чтобы использовать режим ЦП в старых устройствах iPod, iPhone и iPad и включить режим графического процессора для всех остальных устройств, в дескрипторе приложения укажите следующее:

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

Экраны с высоким разрешением

Элемент `requestedDisplayResolution` указывает, какой режим разрешения следует использовать в приложении (*стандартное* или *высокое* разрешение) на устройствах iOS с высоким разрешением.

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

В режиме высокого разрешения каждый пиксел экрана с высоким разрешением можно обрабатывать отдельно. В режиме стандартного разрешения экран устройства для приложения представляется экраном со стандартным разрешением. При отрисовке отдельного пикселя в этом режиме задается цвет для четырех пикселей на экране с высоким разрешением.

Параметр по умолчанию — `standard`. Обратите внимание, что при разработке для устройств под управлением iOS элемент `requestedDisplayResolution` должен быть дочерним к элементу `iPhone` (не к элементам `InfoAdditions` или `initialWindow`).

Если вы планируете использовать различные настройки для различных устройств, в качестве элемента `requestedDisplayResolution` укажите значение по умолчанию. Задайте устройства, для которых значение будет противоположным, с помощью атрибута `excludeDevices`. К примеру, в следующем коде режим высокого разрешения применяется для всех устройств, поддерживающих его, кроме планшетов iPad 3-го поколения, для которых задан стандартный режим:

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

Атрибут `excludeDevices` доступен в версии AIR 3.6 и более поздних.

Дополнительные разделы справки

[«requestedDisplayResolution»](#) на странице 254

[Ренаун Эриксон \(Renaun Erickson\). Разработка для сетчаточных и несетчаточных экранов ОС iOS с использованием AIR 2.6](#)

Пользовательские URI-схемы для iOS

Можно зарегистрировать пользовательскую схему URI, чтобы разрешить вызов приложения по ссылке на веб-странице или другим исходным приложением на устройстве. Чтобы зарегистрировать схему URI, добавьте ключ `CFBundleURLTypes` в элемент `InfoAdditions`. В следующем примере выполняется регистрация схемы URI с именем `com.example.app`, чтобы разрешить вызов приложения по URL, которые имеют вид `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>
```

Когда приложение вызывается с помощью пользовательского URI, объект `NativeApplication` отправляет событие `invoke`. URL ссылки с параметрами запроса помещается в массив `arguments` объекта `InvokeEvent`. Можно использовать любое количество пользовательских схем URI.

Примечание. Ссылки в экземпляре `StageWebView` не поддерживают открытие URL-адресов, для которых используется пользовательская URI-схема.

Примечание. Если схема уже была зарегистрирована для другого приложения, ваше приложение не может его заменить, так как для этой схемы URI имеется зарегистрированное приложение.

Фильтры совместимости iOS

Добавьте параметры в массив `UIRequiredDeviceCapabilities` в элементе `InfoAdditions`, если ваше приложение может работать только на устройствах с определенными аппаратными или программными возможностями. Например, следующий параметр указывает, что приложению требуется фотокамера и микрофон:

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

Если соответствующие возможности на устройстве отсутствуют, приложение не может быть установлено. Для приложений AIR можно задать следующие параметры совместимости:

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

Среда AIR 2.6 и более поздних версий автоматически добавляет `armv7` и `opengles-2` в список обязательных возможностей.

Примечание. Эти возможности не требуется включать в дескриптор приложения, чтобы приложение могло их использовать. Параметры `UIRequiredDeviceCapabilities` следует использовать, только чтобы предотвратить установку приложения на устройства, на которых оно не сможет работать корректно.

Выход вместо приостановки

Когда пользователь переключается из приложения AIR в другую программу, приложение переходит в фоновый режим, и его работа приостанавливается. Если необходимо, чтобы в этом случае вместо приостановки осуществлялся выход из приложения, установите для свойства `UIApplicationExitsOnSuspend` значение `YES`:

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

Уменьшение загружаемого объема путем загрузки внешних ресурсных SWF

AIR 3.7

Можно уменьшить первоначальный размер загружаемого приложения путем упаковки подмножества SWF-файлов, используемых приложением, и загрузки оставшихся SWF-файлов (содержащих только ресурсы) во время выполнения с помощью метода `Loader.load()`. Для использования этой функции необходимо упаковать приложение таким образом, чтобы ADT переместил весь код ActionScript ByteCode (ABC) из внешне загружаемых SWF-файлов в главный SWF-файл приложения, оставляя SWF-файл, содержащий только ресурсы. Этого требует правило Apple Store, запрещающее загрузку любого кода после установки приложения.

ADT выполняет следующие действия для поддержки внешне загружаемых SWF-файлов (также называемых очищенными SWF):

- Считывает текстовый файл, указанный в подэлементе `<externalSwfs>` элемента `<iPhone>`, для доступа к списку SWF-файлов (по одному на строке), которые требуется загрузить во время выполнения:

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

- Переносит код ABC из каждого внешне загружаемого SWF-файла в основной исполняемый модуль.
- Удаляет внешне загружаемые SWF-файлы из файла `ipa`.
- Копирует очищенные SWF в каталог `.remoteStrippedSWFs`. Эти SWF-файлы размещаются на веб-сервере, и приложение загружает их по мере необходимости во время выполнения.

Указать, какие SWF-файлы необходимо загрузить во время выполнения, можно, перечислив их в текстовом файле (по одному в строке), как показано в следующем примере:

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

Указанный путь к файлу является относительным к файлу дескриптора приложения. Также необходимо перечислить эти SWF-файлы как ресурсы в команде `adt`.

Примечание. Данная функция относится только к стандартной упаковке. В случае быстрой упаковки (например, при использовании интерпретатора, симулятора или отладчика) ADT не создает очищенные SWF.



Дополнительные сведения об этой функции, включая пример кода, см. в статье [Внешнее размещение вторичных SWF для программ AIR для iOS](#) в блоге инженера Adobe Абхинава Дхадха (Abhinav Dhandh).

Поддержка определения геоположения

Чтобы обеспечить поддержку определения геоположения, добавьте одну из следующих пар «ключ-значение» в элемент 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>
```

Значки приложения

В следующей таблице представлены размеры значков, которые используются на мобильных платформах:

Размер значков	Платформа
29 x 29	iOS
36 x 36	Android
40x40	iOS
48 x 48	Android, iOS
50 x 50	iOS
57 x 57	iOS
58 x 58	iOS
60x60	iOS
72 x 72	Android, iOS
75x75	iOS
76x76	iOS
80x80	iOS
87x87	iOS
96x96	Android
100 x 100	iOS
114 x 114	iOS
120x120	iOS
144 x 144	Android, iOS
152x152	iOS
167 x 167	iOS
180x180	iOS

Размер значков	Платформа
192x192	Android
512x512	Android, iOS
1024 x 1024	iOS

Укажите путь к файлам значков, используя элемент `icon` в файле дескриптора приложения:

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

Если значок нужного размера отсутствует, используется следующий по размеру значок, который уменьшается до нужного размера.

Значки на Android

На платформе Android значки, указанные в дескрипторе приложения, используются в качестве значка Launcher. Значок Launcher должен быть предоставлен в виде набора изображений PNG с размерами 36x36, 48x48, 72x72, 96x96 и 192x192 пикселей. Эти значки используются для экранов с низкой, средней и высокой плотностью пикселей.

Разработчики должны отправлять значок размером 512x512 пикселей во время оформления программы на Google Play Store.

Значки в iOS

Значки, определенные в дескрипторе приложения, используются в следующих местах приложений iOS:

- Значок 29 x 29 пикселей: значок поиска Spotlight для устройств iPhone или iPod с более низким разрешением и значок настроек для устройств iPad с более низким разрешением.
- Значок 40 x 40 пикселей: значок Spotlight для устройств iPad с более низким разрешением.
- Значок 48 x 48 пикселей: AIR добавляет границу к изображению и использует его в качестве значка размером 50 x 50 пикселей для поиска Spotlight на устройствах iPad с более низким разрешением.
- Значок 50 x 50 пикселей: поиск Spotlight для устройств iPad с более низким разрешением.
- Значок 57 x 57 пикселей: значок приложения для устройств iPhone или iPod с более низким разрешением.
- Значок 58 x 58 пикселей: значок Spotlight для устройств iPhone или iPod с дисплеем Retina и значок настроек для устройств iPad с дисплеем Retina.
- Значок 60 x 60 пикселей: значок приложения для устройств iPhone или iPod с более низким разрешением.
- Значок 72 x 72 пикселя (необязательно): значок приложения для устройств iPad с более низким разрешением.
- Значок 76 x 76 пикселей (необязательно): значок приложения для устройств iPad с более низким разрешением.
- Значок 80 x 80: значок Spotlight для устройств iPhone, iPod или iPad с высоким разрешением.
- Значок 100 x 100 пикселей: поиск Spotlight для устройств iPad с дисплеем Retina.
- Значок 114 x 114 пикселей: значок приложения для устройств iPhone или iPod с дисплеем Retina.
- Значок 120 x 120 пикселей: значок приложения для устройств iPhone или iPod с высоким разрешением.

- Значок 152 x 152 пикселя: значок приложения для устройств iPad с высоким разрешением.
- Значок 167 x 167 пикселей: значок приложения для устройств iPad Pro с высоким разрешением.
- Значок 512 x 512 пикселей: значок приложения для устройств iPhone, iPod или iPad с более низким разрешением. Этот значок отображается в программе iTunes. PNG-файл изображения 512 x 512 пикселей используется только для тестирования отладочных версий приложения. При отправке окончательной версии приложения в Apple App Store, изображение 512 x 512 пикселей передается отдельно в формате JPG. Оно не включается в IPA-файл.
- Значок 1024 x 1024 пикселя: значок приложения для устройств iPhone, iPod или iPad с дисплеем Retina.

В ОС iOS значок приобретает эффект блеска. Нет необходимости применять эффект для исходного изображения. Чтобы убрать эффект блеска, используемый по умолчанию, добавьте следующий код в элемент `InfoAdditions` в файле дескриптора приложения:

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

Примечание. На платформе iOS метаданные приложения включаются в формате PNG в состав значков приложения. Это необходимо для того, чтобы компания Adobe могла отслеживать количество приложений AIR, предлагаемых в магазине App Store компании Apple. Если вы не желаете, чтобы ваше приложение распознавалось по метаданным значков как приложение AIR, следует распаковать файл IPA, удалить метаданные значков, затем упаковать файлы заново. Эта процедура освещается в статье [Отказ от анализа приложений AIR для iOS](#).

Дополнительные разделы справки

«`icon`» на странице 242

«`imageNxN`» на странице 243

[Разработчики Android: руководство по созданию значков](#)

[Рекомендации по пользовательскому интерфейсу iOS: создание пользовательского значка и изображения](#)

Изображения для запуска iOS

Помимо значков приложения также требуется предоставить хотя бы одно изображение для запуска: `Default.png`. Дополнительно можно включить отдельные изображения для запуска при различных ориентациях, различных разрешениях (включая высокое разрешение дисплеев Retina и дисплеи с пропорциями 16:9) и на различных устройствах. Также можно включить разные изображения для запуска, которые будут использоваться при вызове приложения по URL.

Ссылка на файлы изображений для запуска не включается в дескриптор приложения. Файлы следует поместить в корневой каталог приложения. (не помещайте эти файлы в подкаталог).

Схема именования файлов

При выборе имен изображений следует использовать следующую схему:

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

Необходимой частью имени файла является только один фрагмент — *basename*. В качестве него указывается либо *Default* (с заглавной буквы D), либо имя, указанное с помощью ключа `UILaunchImageFile` в элементе `InfoAdditions` дескриптора приложения.

Фрагмент *screen size modifier* определяет размер экрана, если он не относится к стандартным. Этот модификатор применим только для моделей iPhone и iPod touch с пропорциями экрана 16:9, таким как iPhone 5 и iPod touch (5-го поколения). Для этого модификатора поддерживается только одно значение: `-568h`. Поскольку эти устройства оснащены дисплеями Retina высокого разрешения, модификатор размера экрана всегда применяется вместе с модификатором масштаба `@2x`. Таким образом, полное имя изображений для запуска на таких устройствах (без дополнительных параметров) будет `Default-568h@2x.png`.

Фрагмент *urischeme* представляет собой строку, используемую для идентификации схемы URL. Этот фрагмент применим только в том случае, если приложение поддерживает одну или несколько пользовательских схем URL. Например, если приложение можно вызвать по такой ссылке, как `example://foo`, в качестве фрагмента схемы в имени файла изображения для запуска следует использовать строку `-example`.

Фрагмент *orientation* позволяет отображать различные изображения для запуска в зависимости от ориентации устройства при запуске приложения. Этот фрагмент применим только к изображениям для приложений iPad. В качестве него можно указать одно из следующих значений, соответствующих ориентации устройства при запуске приложения:

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

Фрагмент *scale* равен `@2x` (iPhone 4, iPhone 5 и iPhone 6) или `@3x` (iPhone 6 plus) для изображений запуска, используемых для дисплеев с высоким разрешением (Retina). (не указывайте фрагмент *scale* для изображений, используемых для устройств со стандартным разрешением). В случае изображений для запуска на более длинных устройствах, таких как iPhone 5 или iPod touch (5-го поколения), следует также указать после фрагмента *basename* и перед любыми другими фрагментами имени модификатор размера экрана `-528h`.

Фрагмент *device* предусмотрен для назначения отдельных изображений для запуска на карманных устройствах и телефонах. Этот фрагмент используется для универсальных приложений, поддерживающих в одном бинарном файле `app` как карманные устройства, так и планшетные компьютеры. Допустимые значения: `~ipad` и `~iphone` (для устройств iPhone и iPod Touch).

Для iPhone можно включать только изображения с портретным соотношением сторон. Однако для iPhone 6 plus также можно добавлять изображения с альбомной ориентацией. Для устройств со стандартным разрешением размер изображения в пикселях должен быть равен 320x480, с высоким разрешением — 640x960, а для устройств с пропорциями 16:9, таких как iPhone 5 и iPod touch (5-го поколения) — 640x1136.

Для iPad следует добавить изображения по следующей схеме:

- AIR 3.3 и более ранней версии — не полноэкранные изображения альбомной ориентации (1024x748 для нормального разрешения, 2048x1496 для высокого разрешения) и книжной ориентации (768x1004 для нормального разрешения, 1536x2008 для высокого разрешения).

- AIR 3.4 и более поздней версии — полноэкранные изображения альбомной ориентации (1024x768 для нормального разрешения, 2048x1536 для высокого разрешения) и книжной ориентации (768x1024 для нормального разрешения, 1536x2048 для высокого разрешения). При добавлении в пакет полноэкранного изображения для не полноэкранного приложения верхние 20 пикселей (или 40 пикселей в высоком разрешении) будут перекрыты строкой состояния. Не используйте эту часть изображения для вывода важной информации.

Примеры

В следующей таблице показан пример набора изображений для запуска, которые можно включать для гипотетического приложения, поддерживающего максимальное число устройств и ориентаций, и которое можно запускать по URL со схемой `example://`:

Имя файла	Размер изображения	Применение
Default.png	320 x 480	iPhone, стандартное разрешение
Default@2x.png	640 x 960	iPhone, высокое разрешение
Default-568h@2x.png	640 x 1136	iPhone, высокое разрешение, пропорции 16:9
Default-Portrait.png	768 x 1004 (AIR 3.3 и более ранних версий) 768 x 1024 (AIR 3.4 и более поздних версий)	iPad, портретная ориентация
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 и более ранних версий) 1536 x 2048 (AIR 3.4 и более поздних версий)	iPad, высокое разрешение, книжная ориентация
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 и более ранних версий) 768 x 1024 (AIR 3.4 и более поздних версий)	iPad, перевернутая портретная ориентация
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 и более ранних версий) 1536 x 2048 (AIR 3.4 и более поздних версий)	iPad, высокое разрешение, перевернутая книжная ориентация
Default-Landscape.png	1024 x 768	iPad, альбомная ориентация с поворотом влево
Default-LandscapeLeft@2x.png	1536 x 2048	iPad, высокое разрешение, альбомная ориентация по левому краю
Default-LandscapeRight.png	1024 x 768	iPad, альбомная ориентация с поворотом вправо
Default-LandscapeRight@2x.png	1536 x 2048	iPad, высокое разрешение, альбомная ориентация по правому краю
Default-example.png	320 x 480	example:// URL на стандартном iPhone

Имя файла	Размер изображения	Применение
Default-example@2x.png	640 x 960	example:// URL на iPhone с высоким разрешением
Default-example~ipad.png	768 x 1004	example:// URL на iPad с портретной ориентацией
Default-example-Landscape.png	1024 x 768	example:// URL на iPad с альбомной ориентацией

Этот пример иллюстрирует только один подход. Например, можно использовать изображение Default.png для устройства iPad и указать определенные изображения для запуска для устройств iPhone и iPod с именами Default~iphone.png и Default@2x~iphone.png.

См. также

[Руководство по программированию приложений iOS: изображения для запуска приложений](#)

Изображения для запуска, которые требуется упаковать для устройств iOS

Устройства	Разрешение (пиксели)	Имя изображения для запуска	Ориентация
iPhone			
iPhone 4 (не Retina)	640 x 960	Default~iphone.png	Книжная
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Книжная
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	Книжная
iPhone6, iPhone 7	750 x 1334	Default-375w-667h@2x~iphone.png	Книжная
iPhone 6+, iPhone 7+	1242 x 2208	Default-414w-736h@3x~iphone.png	Книжная
iPhone 6+, iPhone 7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	Альбомная
iPad			
iPad 1, 2	768 x 1024	Default-Portrait~ipad.png	Книжная
iPad 1, 2	768 x 1024	Default-PortraitUpsideDown~ipad.png	Книжная перевернутая
iPad 1, 2	1024 x 768	Default-Landscape~ipad.png	Левая альбомная
iPad 1, 2	1024 x 768	Default-LandscapeRight~ipad.png	Правая альбомная
iPad 3, Air	1536x2048	Default-Portrait@2x~ipad.png	Книжная
iPad 3, Air	1536x2048	Default-PortraitUpsideDown@2x~ipad.png	Книжная перевернутая
iPad 3, Air	1536 x 2048	Default-LandscapeLeft@2x~ipad.png	Левая альбомная
iPad 3, Air	1536 x 2048	Default-LandscapeRight@2x~ipad.png	Правая альбомная
iPad Pro	2048 x 2732	Default-Portrait@2x.png	Книжная
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Альбомная

Рекомендации по содержимому

В изображениях для запуска может использоваться любая графика при условии сохранения правильных размеров. Однако в большинстве случаев рекомендуется, чтобы изображение соответствовало начальному экрану приложения. При создании таких изображений можно использовать снимок экрана с окном запуска приложения.

- 1 Откройте приложение на устройстве с iOS. Когда появится первый экран пользовательского интерфейса, нажмите и удерживайте клавишу «Домой» (под экраном). Удерживая кнопку «Домой», нажмите кнопку «Питание/Режим сна» (вверху на устройстве). В результате будет сделан снимок экрана и отправлен в приложение «Фотопленка».
- 2 Перенесите изображение на рабочий компьютер с помощью функции переноса фотографий из iPhoto или другого приложения для перемещения фотографий.

Не добавляйте текст в изображение для запуска, если приложение локализовано на несколько языков. Изображение для запуска является статическим, и текст не будет соответствовать другим языкам.

См. также

[Рекомендации по пользовательскому интерфейсу iOS: изображения для запуска](#)

Игнорируемые настройки

Приложения на мобильных устройствах игнорируют настройки приложения, которые относятся к собственным окнам или функциям операционной системы настольного компьютера. Игнорируются следующие настройки:

- allowBrowserInvocation
- customUpdateUI
- fileTypeTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

Упаковка приложения AIR для мобильных устройств

С помощью команды `ADT -package` создайте пакет приложения AIR, предназначенный для мобильных устройств. Параметр `-target` определяет мобильную платформу, для которой создается пакет.

Пакеты Android

Приложения AIR на Android вместо формата пакетов AIR используют формат пакетов Android (APK).

Пакеты, созданные с помощью ADT для целевой платформы *APK*, имеют формат, подходящий для размещения приложения на Android Маркете. Для размещения на Android Маркет приложения должны удовлетворять определенным требованиям. Перед созданием окончательной версии пакета следует ознакомиться с актуальными требованиями. См. статью [Разработчики Android: публикация в Маркете](#).

В отличие от приложений iOS для подписания приложения Android можно использовать обычный сертификат для подписания кода AIR, однако для отправки приложений на Android Маркет сертификат должен соответствовать правилам Маркета, согласно которым сертификат должен быть действителен по крайней мере до 2033 года. Такой сертификат можно создать с помощью команды `ADT -certificate`.

Чтобы разместить приложение на альтернативном рынке, на котором приложениям не разрешено запрашивать загрузку AIR с рынка Google, можно указать альтернативный URL загрузки, используя параметр `ADT -airDownloadURL`. Когда пользователи, не имеющие требуемой версии среды выполнения AIR, будут запускать приложение, они будут перенаправлены по указанному URL. Дополнительные сведения см. в разделе «[Команда ADT package](#)» на странице 177».

По умолчанию ADT упаковывает приложение Android вместе с общей средой выполнения. Поэтому для запуска приложения пользователю необходимо установить отдельную среду выполнения AIR на устройстве.

Примечание. Чтобы с помощью ADT создать APK, использующий связанную среду выполнения, используйте `target apk-captive-runtime`.

Пакеты iOS

Приложения AIR в iOS вместо собственного формата AIR используют пакетов iOS (IPA).

Если пакеты, созданные с помощью ADT для целевой платформы *ipa-app-store*, имеют правильный сертификат для подписи кода и надлежащий профиль поставки, такие пакеты можно размещать в Apple App Store. Используйте тип целевой платформы *ipa-ad-hoc*, чтобы создать пакет приложения для специального развертывания.

Для подписания приложения необходимо использовать правильный сертификат разработчика, выданный Apple. Другие сертификаты применяются для создания тестовых сборок, которые используются при генерации окончательной версии пакета перед отправкой приложения.

Пример создания пакета приложения для iOS с использованием Ant см. в публикации [Петр Вальчижин \(Piotr Walczyszyn\)](#). Создание пакета приложения AIR для устройств iOS с помощью команды ADT и сценария ANT

Упаковка с помощью ADT

Среда AIR SDK 2.6 и более поздних версий поддерживает создание пакетов для ОС iOS и Android. Перед упаковкой необходимо скомпилировать код ActionScript, MXML, а также все коды расширения. Кроме того, требуется сертификат для подписи кода.

Подробная информация о командах и параметрах ADT представлена в разделе «[AIR Developer Tool \(ADT\)](#)» на странице 176.

Пакеты Android APK

Создание пакета APK

Для создания пакета APK используйте команду ADT `package`, установите тип целевой платформы `apk` для рабочих версий, `apk-debug` — для отладочных версий или `apk-emulator` — для сборок, запускаемых в рабочем режиме на эмуляторе.

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

Введите команду в одной строке. Разрывы строки в приведенном выше примере служат только для упрощения восприятия. Кроме того, в данном примере предполагается, что путь к инструменту ADT указан в определении пути оболочки командной строки. (Инструкции см. в разделе «[Переменные среды Path](#)» на странице 327».)

Команда должна быть запущена из каталога с файлами приложения. В данном примере представлены файлы приложений `myApp-app.xml` (файл дескриптора приложения), `myApp.swf` и каталог значков.

Если приложение запускается, как показано в примере, ADT предложит ввести пароль для хранилища ключей (при вводе пароль не отображается, поэтому после завершения ввода требуется просто нажать Enter).

***Примечание.** По умолчанию все приложения AIR Android имеют префикс `air`. в имени пакета. Чтобы изменить такое поведение по умолчанию, задайте переменной среды `AIR_NOANDROIDFLAIR` значение `true` на своем компьютере.*

Создание пакета APK для приложения, в котором используются собственные расширения

Чтобы создать пакет APK для приложения, которое использует собственные расширения, добавьте параметр `-extdir` в дополнение к обычным параметрам упаковки. Если имеется несколько ANE, которые используют общие ресурсы или библиотеки, ADT выбирает только один ресурс или библиотеку и игнорирует другие повторяющиеся записи, прежде чем отправлять предупреждение. Этот параметр указывает каталог, содержащий файлы ANE, используемые приложением. Например:

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

Создание пакета APK, включающего собственную версию среды выполнения AIR

Чтобы создать пакет APK, который содержит и приложение и связанную версию среды выполнения AIR, используйте в качестве цели `apk-captive-runtime`. Этот параметр указывает каталог, содержащий файлы ANE, используемые приложением. Например:

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

Этой технике свойственны следующие недостатки.

- Важные исправления системы защиты недоступны пользователям автоматически после публикации исправления системы защиты компанией Adobe
- Программа использует больше ОЗУ

Примечание. Когда среда выполнения включается в пакет приложения, ADT добавляет в приложения разрешения `INTERNET` и `BROADCAST_STICKY`. Они требуются для среды выполнения AIR.

Создание отладочного пакета APK

Чтобы создать версию приложения, которая будет использоваться с отладчиком, установите в качестве типа целевой платформы `apk-debug` и укажите параметры подключения:

```
adt -package
    -target apk-debug
    -connect 192.168.43.45
    -storetype pkcs12 -keystore ../codesign.p12
    MyApp.apk
    MyApp-app.xml
    MyApp.swf icons
```

Флаг `-connect` указывает среде выполнения AIR на устройстве, когда требуется подключиться по сети к удаленному отладчику. Для выполнения отладки по USB вместо этого следует установить флаг `-listen` и указать порт TCP, который будет использоваться для подключения к отладчику:

```
adt -package
    -target apk-debug
    -listen 7936
    -storetype pkcs12 -keystore ../codesign.p12
    MyApp.apk
    MyApp-app.xml
    MyApp.swf icons
```

Для работы большинства функций отладки необходимо скомпилировать файлы SWF и SWC приложения, для которых включен режим отладки. В разделе «[Параметры подключения к отладчику](#)» на странице 195 приводится подробное описание флагов `-connect` и `-listen`.

Примечание. По умолчанию ADT упаковывает связанную копию среды выполнения AIR в программу для Android, когда упаковывает приложение с заданной целью `apk-debug`. Чтобы с помощью ADT создать APK, использующий внешнюю среду выполнения, установите для переменной среды `AIR_ANDROID_SHARED_RUNTIME` значение `true`.

На платформе Android приложения должны получить разрешение на доступ к Интернету, чтобы установить соединение с компьютером, на котором запущен отладчик. См. раздел «[Разрешения Android](#)» на странице 81».

Создание пакета APK для использования на эмуляторе Android

На эмуляторе Android можно использовать отладочный пакет APK вместо пакета с рабочим режимом. Создать пакет APK с рабочим режимом для эмулятора можно с помощью команды `ADT package`. При этом следует установить тип целевой платформы `apk-emulator`:

```
adt -package -target apk-emulator -storetype pkcs12 -keystore ../codesign.p12 MyApp.apk MyApp-app.xml MyApp.swf icons
```

В данном примере предполагается, что путь к инструменту ADT указан в определении пути оболочки командной строки. (Инструкции см. в разделе «[Переменные среды Path](#)» на странице 327».)

Создание пакета APK из файла AIR или AIRI

Пакет APK можно создать непосредственно на основе существующего файла AIR или AIRI:

```
adt -target apk -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp.air
```

В файле AIR в дескрипторе приложения должно быть указано пространство имен AIR 2.5 или более поздней версии.

Создание пакета APK для платформы Android x86

Начиная с AIR 14, аргумент `-arch` можно использовать с целью упаковки APK для платформы Android x86. Например:

```
adt -package -target apk-debug -listen 7936 -arch x86 -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp-app.xml myApp.swf icons
```

Пакеты iOS

В iOS инструмент ADT преобразует байт-код файла SWF и другие исходные файлы в собственное приложение iOS.

- 1 Создайте SWF-файл с использованием Flash Builder, Flash Professional или компилятора командной строки.
- 2 Откройте командную строку или терминал и перейдите в папку проекта приложения iPhone.
- 3 Затем с помощью инструмента ADT создайте IPA-файл, используя следующий синтаксис:

```
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
```

Измените ссылку на приложение `adt`, указав полный путь к нему. Приложение `adt` установлено в подкаталоге `bin` в каталоге AIR SDK.

Выберите параметр `-target` в соответствии с типом приложения iPhone, которое требуется создать.

- `-target ipa-test`: выберите этот вариант, чтобы быстро скомпилировать версию приложения для тестирования на рабочем устройстве iPhone. Кроме того, можно использовать `ipa-debug-interpreter` для еще более быстрой компиляции или `ipa-debug-interpreter-simulator` для запуска в приложении iOS Simulator.
- `-target ipa-debug`: выберите этот вариант, чтобы скомпилировать отладочную версию приложения для тестирования на рабочем устройстве iPhone. При выборе этого варианта можно использовать отладочный сеанс для получения вывода `trace()` от приложения iPhone.

Для указания IP-адреса компьютера, на котором выполняется отладка, добавьте один из следующих параметров `-connect` (`CONNECT_OPTIONS`):

- `-connect` — программа будет пытаться подключиться через WiFi к сеансу отладки на компьютере, используемом для разработки, для компиляции программы.
- `-connect IP_ADDRESS` — программа будет пытаться подключиться через WiFi к сеансу отладки на компьютере с указанным IP-адресом. Например:

```
-target ipa-debug -connect 192.0.32.10
```
- `-connect HOST_NAME` — программа будет пытаться подключиться через WiFi к сеансу отладки на компьютере с указанным именем хоста. Например:

```
-target ipa-debug -connect bobroberts-mac.example.com
```

Параметр `-connect` не является обязательным. Если он не задан, программа отладки не будет пытаться подключиться к отладчику на хосте. Также можно использовать `-listen` вместо `-connect`, чтобы включить отладку USB, как описано в разделе «Удаленная отладка с помощью FDB через USB» на странице 112.

Если не удастся подключиться к отладчику, программа отображает диалоговое окно, в котором пользователь должен ввести IP-адрес хоста, на котором производится отладка. Если устройство не подключено к сети Wi-Fi, установить подключение может быть невозможно. Это возможно и в случае, если устройство подключено, но не защищено брандмауэром хоста, на котором производится отладка.

Кроме того, можно использовать `ipa-debug-interpreter` для ускоренной компиляции или `ipa-debug-interpreter-simulator` для запуска в приложении iOS Simulator.

Дополнительные сведения см. в разделе «Отладка приложений AIR для мобильных устройств» на странице 106».

- `-target ipa-ad-hoc`: выберите этот вариант, чтобы создать приложения для специального развертывания. Сведения см. в центре разработчиков Apple iPhone
- `-target ipa-app-store`: выберите этот вариант, чтобы создать окончательную версию IPA-файла для развертывания в Apple App Store.

Замените строку `PROFILE_PATH` на путь к файлу профиля обеспечения своего приложения. Дополнительные сведения о профилях поставки см. в разделе «Настройка ОС iOS» на странице 69».

Используйте параметр `-platformsdk`, чтобы указать на набор iOS Simulator SDK при построении для запуска приложения в iOS Simulator.

Вместо `SIGNING_OPTIONS` укажите ссылку на сертификат и пароль разработчика iPhone. Пользуйтесь следующим синтаксисом:

```
-storetype pkcs12 -keystore P12_FILE_PATH -storepass PASSWORD
```

Вместо `P12_FILE_PATH` укажите путь к файлу сертификата P12. Вместо `PASSWORD` укажите пароль сертификата. (См. пример ниже.) Дополнительные сведения о файле сертификата P12 см. в разделе «Преобразование сертификата разработчика в файл хранилища P12» на странице 212».

Примечание. Можно использовать самозаверяющий сертификат при создании пакетов для iOS Simulator.

Вместо `APP_DESCRIPTOR` укажите файл дескриптора приложения.

Вместо `SOURCE_FILES` укажите основной SWF-файл проекта, а затем остальные ресурсы, которые требуется включить. Укажите пути ко всем файлам значков, определенным в диалоговом окне настроек приложения в среде Flash Professional или пользовательском файле дескриптора приложения. Также добавьте изображение начального экрана (Default.png).

Используйте параметр `-extdir extension-directory` для указания каталога, в котором содержатся ANE-файлы (собственные расширения), используемые в приложении. Если собственные расширения не используются в приложении, не включайте этот параметр.

Важная информация. Не создавайте подкаталогов в каталоге приложения `Resources`. Среда выполнения автоматически создает папку с этим именем для соответствия структуре пакета IPA. Создание папки `Resources` вручную приводит к неустранимому конфликту.

Создание пакета iOS для отладки

Чтобы создать пакет iOS для установки на тестовые устройства, используйте команду ADT `package`, установив в качестве типа целевой платформы `ios-debug`. Прежде чем выполнять эту команду, необходимо получить сертификат подписи кода для разработки и профиль поставки от 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
```

Примечание. Кроме того, можно использовать `ipa-debug-interpreter` для ускоренной компиляции или `ipa-debug-interpreter-simulator` для запуска в приложении iOS Simulator.

Введите команду в одной строке. Разрывы строки в приведенном выше примере служат только для упрощения восприятия. Кроме того, в данном примере предполагается, что путь к инструменту ADT указан в определении пути оболочки командной строки. (Инструкции см. в разделе «[Переменные среды Path](#)» на странице 327.)

Команда должна быть запущена из каталога с файлами приложения. В данном примере представлены файлы приложений `myApp-app.xml` (файл дескриптора приложения), `myApp.swf`, каталог значков и файл `Default.png`.

Приложение следует подписать с помощью надлежащего сертификата распространения, выданного Apple. Использовать другие сертификаты для подписи кода не разрешается.

Используйте параметр `-connect` для отладки WiFi-подключений. Приложение пытается начать сеанс отладки с помощью средства Flash Debugger (FDB), запущенного для указанного IP-адреса или имени хоста. Используйте параметр `-listen` для отладки USB-подключений. Прежде всего следует запустить приложение, а затем открыть отладчик FDB, который начнет сеанс отладки для выполняющегося приложения. Дополнительные сведения см. в разделе «[Подключение к отладчику Flash Debugger](#)» на странице 110.

Создание пакета iOS для распространения через Apple App Store.

Чтобы создать пакет iOS для размещения в Apple App Store, используйте команду ADT `package`, установив тип целевой платформы `ios-app-store`. Прежде чем выполнять эту команду, необходимо получить сертификат подписи кода для распространения и профиль поставки от 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
```

Введите команду в одной строке. Разрывы строки в приведенном выше примере служат только для упрощения восприятия. Кроме того, в данном примере предполагается, что путь к инструменту ADT указан в определении пути оболочки командной строки. (Инструкции см. в разделе «[Переменные среды Path](#)» на странице 327».)

Команда должна быть запущена из каталога с файлами приложения. В данном примере представлены файлы приложений myApp-app.xml (файл дескриптора приложения), myApp.swf, каталог значков и файл Default.png.

Приложение следует подписать с помощью надлежащего сертификата распространения, выданного Apple. Использовать другие сертификаты для подписи кода не разрешается.

Важная информация. Apple требует, чтобы загрузка приложений в App Store выполнялась через программу Apple Application Loader. Apple предоставляет только Application Loader для Mac OS X. То есть приложения AIR для iPhone можно создавать на компьютере с ОС Windows, однако отправка приложения в App Store должна выполняться с компьютера с OS X 10.5.3 или более поздней версией. Программу Application Loader можно загрузить из центра разработчиков Apple iOS.

Создание пакета iOS для специального развертывания

Чтобы создать пакет iOS для установки на специального развертывания, используйте команду ADT package, установив в качестве типа целевой платформы ios-ad-hoc. Прежде чем выполнять эту команду, необходимо получить соответствующий сертификат подписи кода для специального развертывания и профиль поставки от 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
```

Введите команду в одной строке. Разрывы строки в приведенном выше примере служат только для упрощения восприятия. Кроме того, в данном примере предполагается, что путь к инструменту ADT указан в определении пути оболочки командной строки. (Инструкции см. в разделе «[Переменные среды Path](#)» на странице 327».)

Команда должна быть запущена из каталога с файлами приложения. В данном примере представлены файлы приложений myApp-app.xml (файл дескриптора приложения), myApp.swf, каталог значков и файл Default.png.

Приложение следует подписать с помощью надлежащего сертификата распространения, выданного Apple. Использовать другие сертификаты для подписи кода не разрешается.

Создание пакета iOS для приложения, в котором используются собственные расширения

Чтобы создать пакет iOS для приложения, в котором используются собственные расширения, воспользуйтесь командой создания пакета ADT с параметром -extdir. Используйте команду ADT соответствующим образом для цели (ipa-app-store, ipa-debug, ipa-ad-hoc, ipa-test). Например:

```
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
```

Введите команду в одной строке. Разрывы строки в приведенном выше примере служат только для упрощения восприятия.

Относительно собственных расширений в примере подразумевается, что каталог `extensionsDir` находится в каталоге, в котором выполняется команда. Каталог `extensionsDir` содержит ANE-файлы, используемые приложением.

Отладка приложений AIR для мобильных устройств

Отладку мобильных приложений AIR можно выполнять несколькими способами. Самым простым способом выявления ошибок логики приложения является выполнение отладки с помощью ADL или iOS Simulator на компьютере, на котором приложение было разработано. Кроме того, приложение можно установить на устройство и выполнить отладку удаленно, используя отладчик Flash Debugger, запущенный на настольном компьютере.

Моделирование устройства с помощью ADL

Самым простым и быстрым способом тестирования и отладки функций мобильных приложений является запуск приложения на компьютере, на котором приложение было создано, с использованием утилиты Adobe Debug Launcher (ADL). ADL определяет используемый профиль на основании элемента `supportedProfiles`, который указан в дескрипторе приложения. Если указано несколько профилей, ADL использует первый профиль из списка. Кроме того, используя параметр `ADL -profile`, можно выбрать другой профиль из списка `supportedProfiles` (если элемент `supportedProfiles` не включен в дескриптор приложения, для аргумента `-profile` можно указать любой профиль). Например, следующая команда позволяет запустить приложение для моделирования профиля мобильного устройства:

```
adl -profile mobileDevice myApp-app.xml
```

Когда моделирование мобильного профиля на настольном компьютере осуществляется таким образом, приложение запускается в среде, которая наиболее точно соответствует среде целевого мобильного устройства. API-интерфейсы ActionScript, которые не относятся к мобильному профилю, будут недоступны. Однако ADL не делает различий между возможностями разных мобильных устройств. Например, можно отправлять события нажатия программных клавиш в приложение, даже если на фактическом целевом устройстве программные клавиши отсутствуют.

ADL поддерживает моделирование изменений ориентации устройства и нажатие программных клавиш через команды меню. Когда инструмент ADL запускается с использованием профиля мобильных устройств, ADL отображает меню (в окне приложения или на панели меню), с помощью которого можно генерировать события вращения устройства и нажатия программных клавиш.

Нажатие программных клавиш

ADL моделирует нажатие программных клавиш «Назад», «Меню» и «Поиск» мобильного устройства. Отправлять события нажатия этих клавиш в смоделированную среду устройства можно с помощью меню, которое отображается при запуске ADL с профилем мобильного устройства.

Поворот устройства

ADL позволяет моделировать поворот устройства, используя меню, которое отображается при запуске ADL с профилем мобильного устройства. Смоделированное устройство можно поворачивать вправо и влево.

Моделирование поворота влияет только на приложения, которые поддерживают функцию автоматического изменения ориентации. Чтобы включить эту функцию, в дескрипторе приложения установите для элемента `autoOrients` значение `true`.

Размер экрана

Выполните тестирование приложения на экранах разных размеров, изменяя значение параметра `ADL-screensize`. Передайте код для одного из predefined типов экрана или строку, содержащую четыре значения, представляющие размеры в пикселях для обычного и развернутого окна.

Всегда указывайте для портретной ориентации размеры в пикселях, задавая для ширины меньшее значение, чем для высоты. Например, следующая команда позволяет открывать ADL для моделирования экрана устройства Motorola Droid:

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Список predefined типов экранов представлен в разделе «[Использование ADL](#)» на странице 170».

Ограничения

Некоторые API-интерфейсы не поддерживаются в профиле настольной системы, которая не может быть смоделирована с помощью ADL. Не выполняется моделирование следующих API-интерфейсов:

- Accelerometer
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- Функция «мультитач» и жесты в операционных системах настольных компьютеров, которые не поддерживают эти функции
- SystemIdleMode

Если в приложении используются эти классы, данные функции следует протестировать на реальном устройстве или в эмуляторе.

Аналогичным образом, доступны API-интерфейсы, работающие в среде ADL на настольных компьютерах, которые поддерживают не все типы мобильных устройств. Вот некоторые из них:

- аудиокодек Speex и AAC.
- Поддержка специальных возможностей и чтения с экрана
- RTMPE
- Загрузка SWF-файлов, содержащих байт-код ActionScript
- Шейдеры PixelBender

Протестируйте приложения, в которых используются эти функции, на целевых устройствах, так как ADL не позволяет моделировать все возможности среды выполнения.

Моделирование устройства с помощью iOS Simulator

Приложение iOS Simulator (только для компьютеров Mac) предлагает быстрый способ запуска и отладки приложений для ОС iOS. При тестировании с помощью iOS Simulator не требуется использовать сертификат разработчика или профиль подготовки к работе. Но по-прежнему необходимо создать сертификат p12, хотя он может быть самозаверяющим.

По умолчанию ADT всегда запускает симулятор iPhone. Чтобы изменить устройство-симулятор, выполните следующие действия.

- Используйте команду ниже, чтобы увидеть доступные симуляторы.

```
xcrun simctl list devices
```

Вывод выглядит примерно так, как показано ниже.

```
== 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)
....
```

- Можно выбрать конкретный симулятор, задав значение переменной AIR_IOS_SIMULATOR_DEVICE следующим образом.

```
export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
```

Перезапустите процесс после установки переменной среды и запустите приложение на выбранном симуляторе устройства.

Примечание. Если вместе с приложением iOS Simulator используется ADT, необходимо всегда добавлять параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.

Чтобы запустить приложение в iOS Simulator, выполните следующие действия.

- Используйте команду `adt -package` с параметром `-target ipa-test-interpreter-simulator` или `-target ipa-debug-interpreter-simulator`, как показано в следующем примере:

```
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
```

Примечание. Сейчас параметры подписи больше не требуются, поэтому в флаге `-keystore` можно передать любое значение, так как ADT не будет его учитывать.

- 2 Команда `adt -installApp` позволяет установить приложение в iOS Simulator, как показано в следующем примере:

```
adt -installApp
    -platform ios
    -platformsdk
    /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
    -device ios-simulator
    -package sample_ipa_name.ipa
```

- 3 Команда `adt -launchApp` позволяет запустить приложение в iOS Simulator, как показано в следующем примере:

Примечание. По умолчанию команда `adt -launchApp` запускает приложение в симуляторе iPhone. Чтобы запустить приложение в симуляторе iPad, экспортируйте переменную среды, `AIR_IOS_SIMULATOR_DEVICE = «iPad»`, а затем используйте команду `adt -launchApp`.

```
adt -launchApp
    -platform ios
    -platformsdk
    /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
    -device ios-simulator
    -appid sample_ipa_name
```

Чтобы проверить собственное расширение в iOS Simulator, используйте имя платформы `iPhone-x86` в файле `extension.xml` и укажите `library.a` (статическую библиотеку) в элементе `nativeLibrary`, как показано в следующем примере `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>
```

Примечание. При тестировании собственного расширения в iOS Simulator не используйте статическую библиотеку (файл `.a`), скомпилированный для устройства. Вместо этого используйте статическую библиотеку, скомпилированную для средства моделирования.

Трассировочные инструкции

При запуске мобильного приложения на настольном компьютере данные трассировки выводятся в окне отладчика или терминала, из которого был запущен инструмент ADL. Если приложение запускается на устройстве или эмуляторе, для просмотра данных трассировки можно установить сеанс удаленной отладки. Кроме того, данные трассировки можно просматривать с помощью инструментов разработки программного обеспечения, предоставляемых производителем устройства или операционной системы, если они поддерживают такую функцию.

Во всех случаях SWF-файлы приложения можно компилировать со встроенным отладчиком, чтобы просматривать трассировочные инструкции во время выполнения.

Удаленные трассировочные инструкции на Android

При запуске приложения на устройстве Android или в эмуляторе данные трассировки можно просматривать в системном журнале Android, используя утилиту Android Debug Bridge (ADB), которая включена в Android SDK. Чтобы просмотреть результаты выполнения приложения, используйте следующую команду в интерфейсе командной строки или окне терминала на компьютере:

```
tools/adb logcat air.MyApp:I *:S
```

где *MyApp* — это идентификатор приложения AIR. Аргумент **:S* подавляет вывод результатов для всех остальных процессов. Чтобы помимо данных трассировки также выводилась системная информация о приложении, в фильтр *logcat* можно включить *ActivityManager*:

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

Данные примеры команд предполагают, что ADB запускается из папки Android SDK или папка SDK определена с помощью переменной среды *Path*.

Примечание. В AIR 2.6+ утилита ADB включена в AIR SDK и находится в папке *lib/android/bin*.

Удаленные трассировочные инструкции в iOS

Чтобы просмотреть данные трассировки для приложений, запущенных на устройстве с ОС iOS, необходимо установить сеанс удаленной отладки с помощью Flash Debugger (FDB).

Дополнительные разделы справки

[Android Debug Bridge: использование журнала logcat](#)

«[Переменные среды Path](#)» на странице 327

Подключение к отладчику Flash Debugger

Для выполнения отладки приложения на мобильном устройстве можно запустить на компьютере отладчик Flash Debugger и подключиться к нему по сети. Выполните следующие действия, чтобы включить удаленную отладку:

- На платформе Android в дескрипторе приложения укажите разрешение `android.permission.INTERNET`.
- Скомпилируйте SWF-файлы приложения с поддержкой отладки.
- Создайте пакет приложения с помощью параметра `-target apk-debug` для Android или `-target ipa-debug` для iOS и флага `-connect` (отладка WiFi-подключений) или `-listen` (отладка USB-подключений).

Для удаленной отладки через WiFi устройство должно иметь доступ к TCP-порту 7935 компьютера, на котором запущен отладчик Flash Debugger. Для установки подключения требуется указать IP-адрес или полное имя домена. Для удаленной отладки через USB устройство должно иметь доступ к TCP-порту 7936 или порту, указанному в флаге `-listen`.

При использовании ОС iOS можно также указать параметр `-target ipa-debug-interpreter` или `-target ipa-debug-interpreter-simulator`.

Удаленная отладка с помощью Flash Professional

После завершения подготовки приложения для отладки и настройки разрешений в дескрипторе приложения выполните следующие действия:

- 1 Откройте диалоговое окно «Настройки AIR Android».
- 2 Перейдите на вкладку «Развертывание».
 - Выберите тип «Развертывание на устройство».
 - В качестве операции, выполняемой после публикации, выберите вариант «Установить приложение на подключенное устройство Android».
 - Из раздела операций, выполняемых после публикации, исключите вариант «Запустить приложение на подключенное устройство Android».
 - При необходимости настройте путь к Android SDK.
- 3 Нажмите кнопку «Опубликовать».
Приложение будет установлено и запущено на устройстве.
- 4 Закройте диалоговое окно «Настройки AIR Android».
- 5 В меню Flash Professional выберите Debug > Begin Remote Debug Session > ActionScript 3 (Отладка > Начать сеанс удаленной отладки > ActionScript 3).
В панели вывода Flash Professional будет показано сообщение «Waiting for Player to connect» (Ожидание подключения проигрывателя).
- 6 Запуск приложения на устройстве.
- 7 В диалоговом окне подключения Adobe AIR введите IP-адрес или имя хоста компьютера, на котором запущен отладчик Flash Debugger и нажмите кнопку «ОК».

Удаленная отладка с помощью FDB по сети

Чтобы выполнить отладку запущенного на устройстве приложения с помощью отладчика Flash Debugger (FDB) в режиме командной строки, сначала запустите отладчик на компьютере, на котором выполнялась разработка, а затем запустите приложение на устройстве. В следующей процедуре инструменты AMXMLC, FDB и ADT используются для компиляции, упаковки и отладки приложения на устройстве. В примерах предполагается, что используются Flex и AIR SDK, а путь к каталогу bin определен в переменной среды Path (данное предположение сделано исключительно с целью упрощения примеров команд).

- 1 Откройте окно терминала или интерфейс командной строки и перейдите в каталог, в котором находится исходный код приложения.
- 2 Скомпилируйте приложение с помощью amxmlc, включив поддержку отладки:

```
amxmlc -debug DebugExample.as
```

- 3 Упакуйте приложения, установив в качестве типа целевой платформы apk-debug или ipa-debug:

```
Android
adobe-air -package -target apk-debug -connect -storetype
pkcs12 -keystore ../../AndroidCert.p12 DebugExample.apk DebugExample-app.xml
DebugExample.swf

iOS
adobe-air -package -target ipa-debug -connect -storetype
pkcs12 -keystore ../../AppleDeveloperCert.p12 -provisioning-profile test.mobileprovision
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

Если для отладки используется одно и то же имя хоста или IP-адрес, это значение можно указать после флага `-connect`. Приложение попытается автоматически подключиться к этому IP-адресу или хосту. В противном случае данную информацию потребуется вводить на устройстве каждый раз при запуске отладки.

4 Установка приложения.

На Android можно использовать команду ADT `-installApp`:

```
adt -installApp -platform android -package DebugExample.apk
```

В ОС iOS приложение можно установить с использованием команды ADT `-installApp` или программы iTunes.

5 Откройте второе окно терминала или командной строки и запустите FDB:

```
fdb
```

6 В окне FDB введите команду `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 Запуск приложения на устройстве.

8 После запуска приложения на устройстве или в эмуляторе откроется диалоговое окно соединения Adobe AIR (если имя хоста или IP-адрес были заданы с помощью параметра `-connect` при создании пакета, приложение попытается автоматически подключиться к этому адресу). Введите соответствующий адрес и нажмите ОК.

Чтобы подключиться к отладчику в этом режиме, устройство должно разрешить IP-адрес или имя хоста и установить подключение к TCP-порту 7935. Требуется наличие сетевого подключения.

9 Когда удаленная среда выполнения подключается к отладчику, с помощью команды FDB `break` можно определить точки прерывания, а затем начать выполнение с помощью команды `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
```

Удаленная отладка с помощью FDB через USB

AIR 2.6 (Android) AIR 3.3 (iOS)

Для отладки приложения через USB-подключение создавать его пакет следует с помощью параметра `-listen` вместо параметра `-connect`. Если указан параметр `-listen`, при запуске приложения среда выполнения прослушивает соединение с отладчиком Flash Debugger (FDB) через TCP-порт 7936. Затем следует запустить отладчик FDB с параметром `-p`, чтобы он инициировал подключение.

Процедура отладки USB-подключений для Android

Для того чтобы отладчик Flash Debugger, запущенный на настольном компьютере, мог подключиться к среде выполнения AIR, запущенной на устройстве или в эмуляторе, необходимо использовать утилиту Android Debug Bridge (ADB) из Android SDK или утилиту iOS Debug Bridge (IDB) из набора AIR SDK, чтобы перенаправить порт устройства на порт настольного компьютера.

- 1 Откройте окно терминала или интерфейс командной строки и перейдите в каталог, в котором находится исходный код приложения.

- 2 Скомпилируйте приложение с помощью antxmlc, включив поддержку отладки:

```
antxmlc -debug DebugExample.as
```

- 3 Создайте пакет приложения, установив соответствующую цель отладки (например, apk-debug), и укажите параметр -listen:

```
adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12  
DebugExample.apk DebugExample-app.xml DebugExample.swf
```

- 4 Подключите устройств для отладки к компьютеру с помощью USB-кабеля (данную процедуру также можно использовать для отладки приложений, запущенных в эмуляторе, и в этом случае USB-соединение устанавливать необязательно, и установить его невозможно).

- 5 Установка приложения.

Используйте команду ADT -installApp:

```
adt -installApp -platform android -package DebugExample.apk
```

- 6 С помощью утилиты Android ADB перенаправьте TCP-порт 7936 с устройства или эмулятора на настольный компьютер:

```
adb forward tcp:7936 tcp:7936
```

- 7 Запуск приложения на устройстве.

- 8 В окне терминала или командной строке запустите FDB с параметром -p:

```
fdb -p 7936
```

- 9 В окне FDB введите команду run:

```
Adobe fdb (Flash Player Debugger) [build 14159]  
Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.  
(fdb) run
```

- 10 Утилита FDB попытается подключиться к приложению.

- 11 После установки подключения можно настроить точки останова с помощью команды FDB break, а затем запустить выполнение с помощью команды 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
```

Примечание. Порт 7936 используется по умолчанию для отладки по USB средой выполнения AIR и утилитой FDB. С помощью параметра `ADB -listen` и параметра `FDB -p` можно указать другой порт. В этом случае следует использовать утилиту `Android Debug Bridge`, чтобы перенаправить порт, указанный в ADT, на порт, указанный в FDB: `adb forward tcp:adt_listen_port# tcp:fdb_port#`

Процедура отладки USB-подключений для iOS

Для того чтобы отладчик Flash Debugger, запущенный на настольном компьютере, мог подключиться к среде выполнения AIR, запущенной на устройстве или в эмуляторе, необходимо использовать утилиту `iOS Debug Bridge (IDB)` из набора AIR SDK, чтобы перенаправить порт устройства на порт настольного компьютера.

- 1 Откройте окно терминала или интерфейс командной строки и перейдите в каталог, в котором находится исходный код приложения.

- 2 Скомпилируйте приложение с помощью `amxmlc`, включив поддержку отладки:

```
amxmlc -debug DebugExample.as
```

- 3 Создайте пакет приложения, установив соответствующую цель отладки (например, `ipa-debug` или `ipa-debug-interpreter`), и укажите параметр `-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 Подключите устройств для отладки к компьютеру с помощью USB-кабеля (данную процедуру также можно использовать для отладки приложений, запущенных в эмуляторе, и в этом случае USB-соединение устанавливать необязательно, и установить его невозможно).

- 5 Установите и запустите приложение на устройстве iOS. В AIR 3.4 и более поздних версиях можно использовать команду `adt -installApp` для установки приложения через USB.

- 6 Определите дескриптор устройства с помощью команды `idb -devices` (утилита IDB находится в папке `air_sdk_root/lib/aot/bin/iOSBin/idb`):

```
./idb -devices  
  
List of attached devices  
Handle    UUID  
1         91770d8381d12644df91fbcee1c5bbdacb735500
```

Примечание. (AIR 3.4 и более поздних версий) Для определения дескриптора устройства можно использовать команду `adt -devices` вместо команды `idb -devices`.

- 7 Направьте трафик порта компьютера на порт, указанный в параметре `adt -listen` (в этом случае — 16000; значение по умолчанию — 7936), с помощью утилиты IDB и идентификатора устройства, определенного на предыдущем этапе:

```
idb -forward 7936 16000 1
```

В этом примере 7936 — это порт ПК, 16000 — порт, который прослушивает подключенное устройство, а 1 — идентификатор подключенного устройства.

- 8 В окне терминала или командной строке запустите FDB с параметром `-p`:

```
fdb -p 7936
```

- 9 В окне FDB введите команду `run`:

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                     Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                     (fdb) run
```

10 Утилита FDB попытается подключиться к приложению.

11 После установки подключения можно настроить точки останова с помощью команды `FDB break`, а затем запустить выполнение с помощью команды `continue`:

Примечание. Порт 7936 используется по умолчанию для отладки по USB средой выполнения AIR и утилитой FDB. С помощью параметра `IDB -listen` и параметра `FDB -p` можно указать другой порт.

Установка AIR и приложений AIR на мобильные устройства

Пользователи приложения могут установить среду выполнения AIR и приложения AIR, используя обычный механизм распространения, предусмотренный для их устройства.

Например, на Android, пользователи могут устанавливать приложения с Android Маркета. Если в разделе «Настройки приложения» разрешена установка приложений из неизвестных источников, пользователи могут устанавливать приложения по нажатию ссылки на веб-страницы, а также скопировав пакет приложения на устройство и открыть его. Если пользователь пытается установить приложение Android, но еще не установил среду выполнения AIR, он автоматически будет перенаправлен на Маркет, где он сможет установить среду выполнения.

В iOS предусмотрено два способа распространения приложения для конечных пользователей. Основным методом распространения является Apple App Store. Также можно использовать метод специального распространения, чтобы разрешить определенному числу пользователей установку приложения без перехода в App Store.

Установка среды выполнения AIR и приложений для разработки

Поскольку приложения AIR на мобильных устройствах устанавливаются как собственные пакеты, для установки приложений для тестирования можно пользоваться стандартными инструментами платформы. Для установки среды выполнения AIR и приложений AIR можно использовать команды ADT, если они поддерживаются. В настоящее время данный подход поддерживается на Android.

В iOS установку приложения для тестирования можно выполнить с помощью iTunes. Протестированные приложения должны быть подписаны с помощью сертификата для подписи кода Apple, выданного специально для разработки приложений, и упакованы с помощью профиля поставки. Приложение AIR в iOS представляет собой автономный пакет. Отдельная среда выполнения не используется.

Установка приложений AIR с помощью ADT

При разработке приложений AIR с помощью ADT можно устанавливать и удалять среду выполнения и приложения (эти команды также могут быть интегрированы в ИСП, поэтому запускать ADT отдельно не требуется).

Среду выполнения AIR можно установить на устройство или эмулятор с помощью утилиты AIR ADT. При этом требуется установить SDK для устройства. Используйте команду `-installRuntime`:

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```


Если параметр `-package` не указан, пакет среды выполнения, предназначенный для устройства или эмулятора, выбирается из списка доступных в установленном AIR SDK.

Чтобы установить приложение AIR в ОС Android или iOS (AIR 3.4 и более поздних версий), воспользуйтесь похожей командой `-installApp`:

```
adt -installApp -platform android -device deviceID -package path-to-app
```

Значение аргумента `-platform` должно соответствовать устройству, на которое выполняется установка.

Примечание. *Существующие версии среды выполнения AIR и приложение AIR необходимо удалить перед повторной установкой.*

Установка приложений AIR на устройства с iOS с использованием программы iTunes

Чтобы установить приложение AIR на устройство iOS для тестирования, выполните следующие действия.

- 1 Откройте приложение iTunes.
- 2 Если это еще не сделано, добавьте профиль обеспечения данного приложения в iTunes. В iTunes выберите «Файл» > «Добавить в медиатеку». Затем выберите файл профиля обеспечения (с типом файла `mobileprovision`).
- 3 В некоторых версиях iTunes не выполняется замена, если уже установлена та же версия приложения. В таком случае удалите приложение с устройства и из списка приложений в iTunes.
- 4 Дважды щелкните IPA-файл для приложения. Он должен быть указан в списке приложений в iTunes.
- 5 Подключите устройство к порту USB компьютера.
- 6 В iTunes перейдите на вкладку «Программа» для устройства и проверьте, что программа выбрана в списке устанавливаемых.
- 7 Выберите устройство в списке на левой панели программы iTunes. Затем нажмите кнопку «Синхронизировать». По окончании синхронизации приложение Hello World появится на устройстве iPhone.

Если новая версия не установилась, удалите приложение с устройства и из списка приложений в iTunes, а затем повторите процедуру. Такое происходит, если текущая установленная версия использует тот же идентификатор приложения и номер версии.

Дополнительные разделы справки

[«Команда ADT installRuntime»](#) на странице 189

[«Команда ADT installApp»](#) на странице 186

Запуск приложения AIR на устройстве

Установленные приложения AIR запускаются через пользовательский интерфейс на устройстве. Приложения также можно запускать удаленно с помощью утилиты AIR ADT, если эта функция поддерживается:

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

Чтобы запустить приложение, для аргумента `-appid` следует указать идентификатор приложения AIR. Используйте значение, указанное в дескрипторе приложения AIR (без префикса `air.`, который добавляется при упаковке).

Если подключено и запущено только одно устройство или эмулятор, флаг `-device` можно не указывать. Значение аргумента `-platform` должно соответствовать устройству, на которое выполняется установка. В настоящее время поддерживается только значение `android`.

Удаление среды выполнения AIR и приложений

Для удаления приложений можно использовать стандартные средства, доступные в операционной системе устройства. Кроме того, для удаления среды выполнения и приложений AIR можно использовать утилиту AIR ADT. Для удаления среды выполнения используйте команду `-uninstallRuntime`:

```
adt -uninstallRuntime -platform android -device deviceID
```

Для удаления приложений используйте команду `-uninstallApp`:

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Если подключено и запущено только одно устройство или эмулятор, флаг `-device` можно не указывать. Значение аргумента `-platform` должно соответствовать устройству, на которое выполняется установка. В настоящее время поддерживается только значение *android*.

Настройка эмулятора

Для запуска приложения AIR на эмуляторе устройства обычно требуется с помощью SDK для этого устройства создать и запустить экземпляр эмулятора на компьютере. Затем можно установить версию эмулятора для среды выполнения AIR и приложение AIR в эмулятор. Обратите внимание, что приложения в эмуляторе обычно работают намного медленнее, чем на реальном устройстве.

Создание эмулятора Android

- 1 Запустите Android SDK и приложение AVD Manager:
 - В ОС Windows запустите файл SDK Setup.exe из корневого каталога Android SDK.
 - В ОС Mac OS запустите приложение aandroid из подкаталога tools в каталоге Android SDK.
- 2 Выберите пункт «Settings» (Настройки), а затем «Force https://» (Принудительное использование https://).
- 3 Выберите пункт «Available Packages» (Доступные пакеты). Отобразится список доступных Android SDK.
- 4 Выберите совместимый Android SDK (Android 2.3 или более позднюю версию) и нажмите кнопку «Install Selected» (Установить выбранное).
- 5 Выберите пункт «Virtual Devices» (Виртуальные устройства) и нажмите кнопку «New» (Создать).
- 6 Установите следующие настройки:
 - Имя виртуального устройства
 - Целевой API-интерфейс, например Android 2.3, API level 8
 - Размер SD-карты (например 1024)
 - Схема оформления (например, Default HVGA)
- 7 Нажмите кнопку «Create AVD» (Создать AVD).

Обратите внимание, что создание виртуального устройства может занять длительное время, в зависимости от конфигурации системы.

После этого можно будет запустить новое виртуальное устройство.

- 1 Выберите виртуальное устройство в приложении AVD Manager. Виртуальное устройство, которое было создано ранее, должно быть представлено в списке.
- 2 Выберите виртуальное устройство и нажмите кнопку «Start» (Начать).
- 3 На следующем экране нажмите кнопку «Launch» (Запуск).

На компьютере откроется окно эмулятора. Это может занять несколько секунд. Инициализация операционной системы Android может занять некоторое время. Установить пакет приложения на эмулятор можно с помощью команд `apk-debug` и `apk-emulator`. Приложения, упакованные для целевой платформы `apk`, не работают в эмуляторе.

Дополнительные разделы справки

<http://developer.android.com/guide/developing/tools/othertools.html#android>

<http://developer.android.com/guide/developing/tools/emulator.html>

Обновление мобильных приложений AIR

Мобильные приложения AIR распространяются как исходные пакеты, в связи с чем для них нельзя использовать стандартный механизм обновления, как для других приложений на платформе. Обычно выполняется отправка обновления на тот же рынок или в магазин приложений, через который распространялось исходное приложение.

Мобильные приложения AIR не могут использовать класс AIR Updater или платформу.

Обновление приложений AIR на Android

Приложения, распространяемые через Android Маркет, можно обновлять путем размещения на Маркете новой версии приложения, если соблюдены следующие условия (эти требования предъявляет Маркет, не AIR):

- Пакет APK подписан с использованием такого же сертификата.
- Идентификаторы AIR совпадают.
- В дескрипторе приложения указано большее значение для атрибута `versionNumber` (значение `versionLabel` также следует увеличить, если оно используется).

Пользователи, которые загрузили ваше приложение с Android Маркета, получают уведомление о выходе обновления через интерфейс программы на устройстве.

Дополнительные разделы справки

[Разработчики Android: публикация обновлений на Android Маркете](#)

Обновление приложений AIR в iOS

Приложения AIR, распространяемые через iTunes, можно обновлять путем отправки обновления в магазин, если соблюдаются следующие условия (их определяют Apple App Store, а не среда AIR):

- Код подписи сертификата и профили поставки генерируются для того же самого идентификатора Apple ID.
- В пакете IPA используется тот же идентификатор пакета Apple Bundle ID.
- Обновление не уменьшает список поддерживаемых устройств (то есть, если исходное приложение поддерживает устройства с iOS 3, нельзя создать обновление, в котором поддержка iOS 3 будет исключена).

Важная информация. Поскольку AIR SDK 2.6 и более поздних версий не поддерживает ОС iOS 3, тогда как AIR 2 поддерживает эту версию, нельзя обновить опубликованные приложения для iOS, которые были созданы с помощью AIR 2, используя обновление, созданное с помощью AIR 2.6 и более поздних версий.

Использование push-уведомлений

Push-уведомления позволяют удаленным поставщикам уведомлений отправлять уведомления приложениям, работающим на мобильных устройствах. В AIR 3.4 поддержка push-уведомлений для устройств iOS обеспечивается службой Apple Push Notification Service (APNs).

***Примечание.** Чтобы включить push-уведомления в AIR для приложений Android, используйте соответствующее расширение в машинном коде, например [as3c2dm](#), разработанное специалистом Adobe по пропагандированию технологий Петром Вальчижином (Piotr Walczyszyn).*

В оставшейся части данного раздела описано, как включить push-уведомления в AIR для приложений iOS.

***Примечание.** Предполагается, что читатель располагает идентификатором разработчика Apple и знаком с общим процессом разработки для ОС iOS, а также имеет опыт развертывания хотя бы одного приложения на устройстве iOS.*

Обзор push-уведомлений

Служба Apple Push Notification Service (APNs) позволяет удаленным поставщикам уведомлений отправлять уведомления приложениям, работающим на устройствах iOS. APNs поддерживает следующие типы уведомлений:

- Предупреждения
- Значки
- Звуки

Дополнительные сведения об APNs см. на сайте developer.apple.com.

При использовании в приложениях push-уведомлений выделяются следующие функциональные компоненты:

- **Клиентская программа.** Регистрируется для получения уведомлений и поддерживает связь с удаленными поставщиками уведомлений и получает push-уведомления.
- **ОС iOS.** Управляет взаимодействием клиентской программы и службы APNs.
- **Служба APNs.** Предоставляет клиентской программе в момент регистрации идентификатор `tokenId` и передает в iOS уведомления от удаленных поставщиков уведомлений.
- **Удаленный поставщик уведомлений.** Хранит информацию, связанную с идентификаторами клиентских программ `tokenId` и рассылает push-уведомления службам APNs.

Процесс регистрации

Ниже приведена диаграмма процесса регистрации на сервере для получения push-уведомлений:

- 1 Клиентская программа посылает запрос ОС iOS на включение push-уведомлений.
- 2 ОС iOS перенаправляет запрос службе APNs.
- 3 Сервер APNs возвращает ОС iOS идентификатор `tokenId`.
- 4 ОС iOS передает `tokenId` клиентской программе.
- 5 Клиентская программа, используя собственные механизмы, предоставляет `tokenId` удаленному поставщику уведомлений, который сохраняет его для использования при рассылке push-уведомлений.

Процесс отправки и получения уведомления

Ниже приведена диаграмма процесса отправки и получения уведомления

- 1 Удаленный поставщик уведомлений генерирует уведомление и передает пакет данных, дополненный идентификатором `tokenId`, службе APNs.
- 2 Служба APNs перенаправляет уведомление ОС iOS на устройстве.
- 3 ОС iOS передает содержимое push-уведомления программе.

Прикладной интерфейс push-уведомлений

В AIR 3.4 были добавлены прикладные программные интерфейсы (API) с поддержкой push-уведомлений iOS. Эти интерфейсы API находятся в пакете `flash.notifications` и включают следующие классы:

- `NotificationStyle` — определяет константы для типов уведомлений: `ALERT` (предупреждение), `BADGE` (значок) и `SOUND` (звук).
- `RemoteNotifier` — служит для создания и прекращения подписок на push-уведомления.
- `RemoteNotifierSubscribeOptions` — позволяет выбрать типы принимаемых уведомлений. С помощью свойства `notificationStyles` можно определить вектор строковых значений, задающих регистрацию нескольких типов уведомлений.

В AIR 3.4 также включено событие `flash.events.RemoteNotificationEvent`, которое генерируется объектом `RemoteNotifier` в следующих случаях:

- приложение успешно подписано на уведомление, и от службы APNs получен новый идентификатор `tokenId`;
- получено новое удаленное уведомление.

Помимо этого, объект `RemoteNotifier` генерирует событие `flash.events.StatusEvent` в случае, если в процессе подписки произошла ошибка.

Управление push-уведомлениями из приложения

Чтобы зарегистрировать приложение в качестве получателя push-уведомлений, выполните следующие действия.

- Добавьте в приложение код, создающий подписку на push-уведомления.
- Включите push-уведомления в файле XML приложения.
- Создайте профиль выделения ресурсов и сертификат, которые разрешают использование служб push-уведомлений iOS (iOS Push Services).

Следующий пример кода с примечаниями иллюстрирует процесс подписки на push-уведомления и обработки поступающих push-уведомлений:

```
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());
}
}
}
```

Включение push-уведомлений в файле XML приложения

Чтобы использовать push-уведомления в своем приложении, введите следующее содержимое в элемент Entitlements (находящийся внутри элемента iphone):


```
<iphone>
    ...
    <Entitlements>
    <![CDATA[
        <key>aps-environment</key>
        <string>development</string>
    ]]>
    </Entitlements>
</iphone>
```

Перед непосредственной отправкой приложения в App Store измените содержимое элемента `<string> c development` на `production`:

```
<string>production</string>
```

Если ваше приложение поддерживает локализованные строки, укажите соответствующие языки в элементе `supportedLanguages`, расположенном под элементом `initialWindow`, как в следующем примере:

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

Создайте профиль выделения ресурсов и сертификат, которые разрешают использование служб push-уведомлений iOS (iOS Push Services)

Чтобы приложение могло связываться со службой APNs, необходимо включить в пакет приложения профиль выделения ресурсов и сертификат, разрешающие службу iOS Push Services:

- 1 Войдите в учетную запись разработчика Apple.
- 2 Перейдите на страницу Provisioning Portal (Центр выделения ресурсов).
- 3 Откройте вкладку App IDs (Идентификаторы приложений).
- 4 Нажмите кнопку New App ID (Новый идентификатор приложения).
- 5 Введите описание и укажите идентификатор группы (в этом поле нельзя использовать *).
- 6 Нажмите кнопку Submit (Отправить). Центр выделения ресурсов сгенерирует идентификатор программы, после чего снова откроется страница App IDs (Идентификаторы приложений).
- 7 Нажмите кнопку Configure (Настроить), расположенную справа от идентификатора приложения. Откроется страница Configure App ID (Настройка идентификатора приложения).
- 8 Установите флажок Enable for Apple Push Notification (Разрешить Push-уведомления Apple). Обратите внимание на то, что существует два типа сертификатов SSL, используемых для push-уведомлений: один для разработки и тестирования программ, а другой — для распространения программ.
- 9 Нажмите кнопку Configure (Настроить), расположенную справа от строки Development Push SSL Certificate (Сертификат SSL уровня разработки для push-уведомлений). Откроется страница Generate Certificate Signing Request (Создать запрос на подпись сертификата).
- 10 Сгенерируйте запрос на подпись сертификата с помощью утилиты Keychain Access (Связка ключей), как описано на странице.
- 11 Сгенерируйте сертификат SSL.
- 12 Загрузите и установите сертификат SSL.
- 13 (Необязательно) Повторите пп. 9–12 для пункта Production Push SSL Certificate (Сертификат SSL уровня распространения для push-уведомлений).
- 14 Нажмите Done (Готово). Откроется страница Configure App ID (Настройка идентификатора приложения).

- 15 Нажмите Done (Готово). После этого отобразится страница App IDs (Идентификаторы приложений). Обратите внимание на зеленый кружок напротив строки Push Notification для вашего идентификатора приложения.
- 16 Обязательно сохраните все созданные сертификаты SSL, поскольку они понадобятся в дальнейшем для обеспечения взаимодействия приложения и поставщика уведомлений.
- 17 Откройте вкладку Provisioning (Выделение ресурсов), чтобы отобразить страницу Provisioning Profiles (Профили выделения ресурсов).
- 18 Создайте новый профиль выделения ресурсов для нового идентификатора приложения и загрузите его на свой компьютер.
- 19 Откройте вкладку Certificates (Сертификаты) и загрузите новый сертификат для нового профиля выделения ресурсов.

Использование звуковых push-уведомлений

Чтобы разрешить звуковые push-уведомления в вашем приложении, добавьте звуковые файлы к нему аналогично любым другим ресурсам, однако разместите их в той же папке, что и SWF-файлы и файлы app.xml. Например:

```
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
```

Устройства Apple поддерживают следующие форматы аудио (в файлах aiff, wav или caf):

- Линейный PCM
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Использование локализованных предупреждений

Чтобы использовать в приложении локализованные версии уведомлений-предупреждений, добавьте локализованные строки в виде папок lproj. Например, для поддержки предупреждений на русском языке, сделайте следующее.

- 1 Создайте в проекте папку ru.lproj на том же уровне, на котором находится файл app.xml.
- 2 В папке ru.lproj создайте текстовый файл с именем Localizable.Strings.
- 3 Откройте файл Localizable.Strings в текстовом редакторе и добавьте ключи сообщений и соответствующие им локализованные строки. Например:

```
"PokeMessageFormat" = "La notificación de alertas en español."
```
- 4 Сохраните файл.
- 5 Если в качестве языка устройства выбран «Русский», то при получении приложением предупреждения с заданным значением ключа, будет отображен локализованный вариант текста предупреждения.

Настройка удаленного поставщика уведомлений

Чтобы приложение могло получать push-уведомления, необходим удаленный поставщик уведомлений. Данное серверное приложение работает, как поставщик, принимающий входные данные push-уведомлений и передающий уведомления службе APNs, которая в свою очередь отправляет push-уведомления клиентской программе.

Подробные сведения об отправке push-уведомлений удаленным поставщиком уведомлений см. в разделе «Связь поставщика со службой push-уведомлений Apple» библиотеки разработчика Apple.

Варианты реализации удаленного поставщика уведомлений

Допустимы следующие варианты реализации удаленного поставщика уведомлений.

- Создание собственного поставщика на базе сервера APNS-php с открытым кодом. Для настройки PHP сервера можно воспользоваться проектом <http://code.google.com/p/apns-php/>. Данный проект Google Code позволяет разработать интерфейс, отвечающий требованиям конкретной задачи.
- Использование поставщика услуг. Например, вы можете воспользоваться готовым поставщиком APNs, предлагаемым на сайте <http://urbanairship.com/>. После регистрации в данной службе необходимо предоставить устройству дескриптор. Это можно сделать, например, с помощью следующего кода:

```
private var urlreq:URLRequest;
private var urlLoad:URLLoader = new URLLoader();
private var urlString:String;
//When subscription is successful then only call the
following code
urlString = new
String("https://go.urbanairship.com/api/device_tokens/" + e.tokenId);
urlreq = new URLRequest(urlString);
urlreq.authenticate = true;
urlreq.method = URLRequestMethod.PUT;

URLRequestDefaults.setLoginCredentialsForHost("go.urbanairship.com",
"Application Key","Application Secret");
urlLoad.load(urlreq);

urlLoad.addEventListener(IOErrorEvent.IO_ERROR,iohandler);
urlLoad.addEventListener(Event.COMPLETE,compHandler);

urlLoad.addEventListener(HTTPStatusEvent.HTTP_STATUS,httpHandler);
private function iohandler(e:IOErrorEvent):void{
trace("\n In IOError handler" + e.errorID + " "
+e.type);
}
private function compHandler(e:Event):void{
trace("\n In Complete handler,"+"status: " +e.type +
"\n");
}
private function httpHandler(e:HTTPStatusEvent):void{
tt.appendText("\n in httpstatus handler,"+ "Status:
" + e.status);
}
```

Затем следует отправить тестовые уведомления с помощью инструментов Urban Airship.

Сертификаты для удаленного поставщика уведомлений

Сертификат SSL и закрытый ключ (генерируемый ранее) должны быть скопированы в соответствующее место на сервере удаленного поставщика уведомлений. Обычно эти два файла объединяются в один файл .pem. Это можно сделать следующим образом:

- 1 Откройте окно терминала.
- 2 Создайте файл .pem из сертификата SSL, введя следующую команду:

```
openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
```

3 Создайте файл `.pem` закрытого ключа (`.p12`), введя следующую команду:

```
openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
```

4 Объедините два файла `.pem` в один с помощью следующей команды:

```
cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
```

5 Передайте комбинированный файл `.pem` поставщику серверных услуг во время создания серверного приложения для поддержки push-уведомлений.

Дополнительные сведения см. в разделе «[Установка сертификата SSL и ключа на сервер](#)» руководства программиста Apple по локальным и push-уведомлениям.

Обработка push-уведомлений в приложении

Обработка push-уведомлений в приложении включает следующие составляющие.

- Установка глобальных пользовательских настроек, разрешающих push-уведомления
- Получение от пользователя разрешения на прием конкретных уведомлений
- Обработка push-уведомлений и пересылаемых с ними данных

Настройки, разрешающие push-уведомления

При первом запуске пользователем программы с поддержкой push-уведомлений ОС iOS отображает диалоговое окно с предупреждением: **Программа «Приложение» собирается отправлять Вам push-уведомления** и кнопками «Не разрешать» и «ОК». Выбор варианта «ОК» позволяет программе получать все типы уведомлений, на которые она подписана. Выбор варианта «Не разрешать» запретит программе получать уведомления.

***Примечание.** Также пользователь может открыть раздел «Настройки» > «Уведомления» и выбрать конкретные типы push-уведомлений, которые программа сможет получать от поставщика.*

Рекомендации компании Apple заключаются в том, что программа должна подписываться на push-обновления при каждой активации. Когда приложение вызывает метод `RemoteNotifier.subscribe()`, оно получает событие `RemoteNotificationEvent` типа `token` (дескриптор), содержащее а уникальный 32-байтный идентификатор `tokenId` для заданного приложения на устройстве.

Когда устройство получает push-уведомление, на его экране отображается всплывающее окно с кнопками «Закрыть» и «Запустить». При нажатии кнопки «Закрыть» не выполняется никаких действий. При нажатии кнопки «Запустить» ОС iOS запускает приложение и оно получает событие `flash.events.RemoteNotificationEvent` типа `notification` (уведомление), как описано ниже.

Обработка push-уведомлений и пересылаемых с ними данных

Когда удаленный поставщик уведомлений отправляет уведомление на устройство (используя идентификатор `tokenID`), ваше приложение получает событие `flash.events.RemoteNotificationEvent` типа `notification` (уведомление), независимо от того, запущено оно или нет. В этот момент приложение выполняет специфическую для нее обработку полученного уведомления. Если приложению требуется доступ к данным уведомления, они могут быть получены через свойство `RemoteNotificationEvent.data` в формате JSON.

Глава 8. Разработка приложений AIR для телевизионных устройств

Возможности AIR для телевизионных устройств

Можно создавать приложения Adobe® AIR® для телевизионных устройств, включая телевизоры, цифровые видеомagniтофоны, плееры Blu-ray, если на этих устройствах имеется среда Adobe for TV. Среда AIR for TV оптимизирована для телевизионных устройств и использует, например, аппаратное ускорение для повышения производительности видео и графики.

Приложения AIR for TV представляют собой программы на основе SWF, не на основе HTML. Среда AIR for TV может использовать преимущества аппаратного ускорения, а также другие возможности AIR, которые подходят для среды «гостиная».

Профили устройств

Для определения группы целевых устройств с одинаковыми функциями в AIR используются профили. Приложения AIR for TV могут использовать следующие профили:

- Профиль tv. Этот профиль следует использовать в приложениях, которые предназначены для устройства AIR for TV.
- Профиль extendedtv. Этот профиль следует использовать для приложений for TV, в которых применяются собственные расширения.

Возможности ActionScript, определенные для этих профилей, представлены в разделе «[Профили устройств](#)» на странице 263». Конкретные отличия ActionScript в приложениях AIR for TV рассмотрены в статье «[Справочник по ActionScript 3.0 для платформы Adobe Flash Platform](#)».

Подробные сведения о профилях AIR for TV см. в разделе «[Поддерживаемые профили](#)» на странице 150.

Аппаратное ускорение

Телевизионные устройства поддерживают аппаратное ускорение, благодаря чему существенно повышается производительность графики и видео в приложениях AIR. Чтобы воспользоваться преимуществами аппаратного ускорения, ознакомьтесь с информацией в разделе «[Принципы разработки приложений AIR for TV](#)» на странице 130».

Защита содержимого

Среда AIR for TV предоставляет широкие возможности на основе платного видеосодержимого, от голливудских хитов до независимых фильмов и телесериалов. Поставщики содержимого могут создавать интерактивные приложения с использованием инструментов Adobe. Они могут интегрировать серверные продукты Adobe в инфраструктуру распространения контента, а также работать в сотрудничестве с одним из партнеров экосистемы Adobe.

Защита содержимого является ключевым элементом при распространении платного видео. AIR for TV поддерживает Adobe® Flash® Access™, систему защиты и оплаты содержимого, которая удовлетворяет жестким требованиям по безопасности, предъявляемым владельцами содержимого, включая крупнейшие киностудии.

Flash Access имеет следующие функции:

- Поточковая передача и загрузка видео.
- Различные бизнес-модели, включая поддержку рекламы, подписки, аренду и электронные продажи.
- Различные технологии поставки содержимого, включая HTTP Dynamic Streaming, потоковую передачу через RTMP (Real Time Media Protocol) использование Flash® Media Server и прогрессивную загрузку по HTTP.

AIR for TV также имеет встроенную поддержку RTMPE, зашифрованной версии RTMP, для существующих решений потоковой передачи с более низкими требованиями к безопасности. Flash Media Server предоставляет поддержку RTMPE и соответствующих технологий проверки SWF.

Дополнительные сведения см. в документе [Adobe Flash Access](#) (доступ к Adobe Flash).

Многоканальный звук

Начиная с AIR 3, AIR for TV поддерживает многоканальное аудио для видеороликов, последовательно загружаемых с сервера HTTP. Эта поддержка включает следующие кодеки:

- AC-3 (Dolby Digital)
- E-AC-3 (Enhanced Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

Примечание. Поддержка многоканального аудио для видеороликов, передаваемых потоком с Adobe Flash Media Server, пока недоступна.

Ввод игровых данных

Начиная с AIR 3, AIR for TV поддерживает API-интерфейсы ActionScript, которые позволяют приложениям обмениваться данными с присоединенными игровыми устройствами ввода, такими как джойстики, геймпады и пульты. Хотя эти устройства называются игровыми устройствами ввода, они могут использоваться в любых приложениях AIR for TV, а не только в играх.

На рынке имеется множество игровых устройств ввода с разными возможностями. Поэтому в API-интерфейсе устройства обобщаются, чтобы приложение могло хорошо работать с разными (и возможно неизвестными) типами игровых устройств ввода.

Класс `GameInput` — это точка входа в API-интерфейсы ввода данных игры ActionScript. Дополнительные сведения см. в описании [GameInput](#).

Ускоренная визуализация графики Stage3D

Начиная с AIR 3 профиль AIR for TV поддерживает ускоренную визуализацию графики Stage3D. API-интерфейсы [Stage3D](#) в ActionScript представляют собой API низкого уровня, использующие аппаратное ускорение с использованием графического процессора и обеспечивающие дополнительные возможности 2D- и 3D-визуализации. Эти API-интерфейсы низкого уровня предоставляют разработчикам возможность использования аппаратного ускорения графического процессора для значительного улучшения производительности. Также можно использовать игровые станции, которые поддерживают API-интерфейсы Stage3D в ActionScript.

Дополнительные сведения см. в документе [Gaming engines, 3D, and Stage 3D](#) (Игровые станции, 3D и Stage3D).

Собственные расширения

Если для приложения задан профиль `extendedTV`, оно может использовать пакеты ANE (собственное расширение AIR).

Обычно производитель устройств предоставляет пакеты ANE для реализации доступа к функция устройства, которые в противном случае среда выполнения AIR не поддерживает. Например, собственное расширение может использоваться для переключения каналов на телевизоре или приостановки воспроизведения на видеоплеере.

При упаковке приложения AIR for TV, которое использует пакеты ANE, его можно упаковать в файл AIRN вместо файла AIR.

Собственные расширения для устройств AIR for TV всегда представляют собой собственные расширения, *связанные с устройством*. «Связанные с устройством» означает, что библиотеки расширений установлены на устройстве AIR for TV. Пакет ANE, включаемый в пакет приложения, *никогда* не включает собственные библиотеки расширения. Иногда он содержит версию собственного расширения, в которой используется только ActionScript. Эта содержащая только ActionScript версия является фрагментом кода или симулятором расширения. Производитель устройства устанавливает на него реальное расширение, включающее собственные библиотеки.

Если вы разрабатываете собственные расширения, обратите внимание на следующие факты.

- Разрабатывая собственное расширение для устройства AIR for TV, всегда консультируйтесь с его производителем.
- На некоторых устройствах AIR for TV производитель создает собственные расширения.
- На всех устройствах for TV производитель принимает решение об установке тех или иных собственных расширений.
- Каждый производитель использует свои инструменты для разработки собственных расширений for TV.

Дополнительные сведения об использовании собственных расширений в приложении AIR см. в разделе «[Использование собственных расширений для Adobe AIR](#)» на странице 158.

Дополнительные сведения о создании собственных расширений см. в документе [Разработка собственных расширений для Adobe AIR](#).

Принципы разработки приложений AIR for TV

Вопросы, связанные с видео

Инструкции по шифрованию видео

Adobe рекомендует соблюдать следующие инструкции для организации потоковой передачи видео на ТВ-устройства.

Видеокодек:	H.264, профиль Main или High, прогрессивное шифрование
Разрешение:	720i, 720p, 1080i или 1080p
Частота кадров:	24 кадра в секунду или 30 кадров в секунду

Аудиокодек:	AAC-LC или AC-3, 44,1 кГц, стерео, или эти многоканальные аудиокодеки: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio или DTS-HD Master Audio
Общая скорость потока:	до 8 Мбит/с, в зависимости от пропускной способности потока:
Скорость потока аудио:	до 192 Кбит/с
Попиксельные пропорции:	1 × 1

Для видео, передаваемого на устройство AIR for TV, Adobe рекомендует использовать кодек H.264.

Примечание. AIR for TV также поддерживает видео, зашифрованное с помощью кодеков Sorenson Spark и On2 VP6. Однако оборудование не раскодирует и не отображает видео, закодированное с помощью этих кодеков. Вместо этого среда выполнения сама раскодирует и отображает видео, закодированное с помощью этих кодеков, с использованием программного обеспечения, в связи с чем видео воспроизводится с намного меньшей частотой кадров. Поэтому рекомендуется использовать кодек H.264, когда это возможно.

Класс StageVideo

AIR for TV поддерживает аппаратное раскодирование и отображение видео, закодированного с помощью H.264. Используйте класс StageVideo, чтобы включить эту функцию.

Дополнительные сведения см. в разделе [«Использование класса StageVideo для отображения с аппаратным ускорением»](#) в руководстве разработчика по ActionScript 3.0.

- API-интерфейс класса StageVideo и связанных классов.
- Ограничения использования класса StageVideo.

Для обеспечения наилучшей поддержки приложений AIR, в которых используется объект Video для видео, закодированного с помощью H.264, AIR for TV на *внутреннем уровне* использует объект StageVideo. Такой подход позволяет использовать преимущества аппаратного раскодирования и визуализации при отображении видео. Однако в отношении объекта Video действуют такие же ограничения, как для объекта StageVideo. Например, если приложение пытается повернуть видео, поворота не происходит, так как визуализацию видео выполняет оборудование, а не среда выполнения.

Тем не менее, при создании новых приложений рекомендуется использовать для видео, закодированного с помощью кодека H.264, объект StageVideo.

Пример использования класса StageVideo см. в разделе [«Поставка видео и содержимого для платформы Flash Platform на ТВ-устройствах»](#).

Рекомендации по поставке видео

На устройстве AIR for TV пропускная способность сети может колебаться при воспроизведении видео. Такие колебания возникают, например, когда кто-то другой начинает использовать то же самое подключение к Интернету.

Поэтому Adobe рекомендует, чтобы в системе поставки видео была реализована возможность адаптивной скорости передачи. Например, на серверной стороне адаптивную скорость передачи поддерживает Flash Media Server. На стороне клиента можно использовать Open Source Media Framework (OSMF).

Для передачи по сети видеосодержимого в приложения AIR for TV доступны следующие протоколы:

- HTTP/HTTPS Dynamic Streaming (формат F4F, динамическая потоковая передача)
- RTMP/RTMPE/RTMFP/RTMPT/RTMPTE Streaming (потоковая передача)

- HTTP/HTTPS Progressive Download (последовательная загрузка)

Дополнительные сведения см. в следующих разделах:

- [Adobe Flash Media Server Developer's Guide](#) (руководство разработчика Adobe Flash Media Server).
- [Open Source Media Framework](#).

Вопросы, связанные с аудио

ActionScript для воспроизведения звука в приложениях AIR for TV не отличается от других приложений AIR. Дополнительные сведения см. в разделе «Работа со звуком» в *Руководстве разработчика по ActionScript 3.0*.

В отношении поддержки многоканального аудио для профиля AIR for TV необходимо учитывать следующее.

- AIR for TV поддерживает многоканальное аудио для видеороликов, последовательно загружаемых с сервера HTTP. Поддержка многоканального аудио для видеороликов, передаваемых потоком с Adobe Flash Media Server, пока недоступна.
- И хотя AIR for TV поддерживает многие аудиокодеки, не все *устройства* AIR for TV поддерживают весь набор. Используйте метод `flash.system.Capabilities.hasMultiChannelAudio()` для проверки того, поддерживает ли устройство AIR for TV определенный многоканальный аудиокодек, например AC-3.

Например, рассмотрите приложение, выполняющее прогрессивную загрузку видеофайла с сервера. На сервере содержатся другие видеофайлы H.264, поддерживающие другие многоканальные аудиокодеки. В приложении можно использовать метод `hasMultiChannelAudio()` для определения видеофайла, который необходимо запросить с сервера. Или можно отправить из приложения на сервер строку, содержащуюся в `Capabilities.serverString`. В строке обозначены доступные многоканальные аудиокодеки, благодаря чему сервер может выбрать подходящий видеофайл.

- При использовании одного из аудиокодеков DTS в некоторых ситуациях `hasMultiChannelAudio()` возвращает значение `true`, хотя аудио DTS не воспроизводится.

В качестве примера рассмотрим проигрыватель Blu-ray с выходом S/PDIF, подключенный к старому усилителю. Старый усилитель не поддерживает DTS, а у выхода S/PDIF нет протокола для уведомления проигрывателя Blu-ray об этой ситуации. Если проигрыватель Blu-ray отправляет поток DTS старому усилителю, пользователь ничего не слышит. Поэтому при использовании DTS рекомендуется предусмотреть интерфейс, чтобы пользователь мог сообщить, что отсутствует звук. В таком случае приложение может переключиться на другой кодек.

В следующей таблице описаны ситуации использования разных аудиокодеков в приложениях AIR for TV. В таблице также перечислены условия, при которых на устройстве AIR for TV используются аппаратные ускорители для декодирования аудиокодека. Аппаратное ускорение позволяет повысить производительность и уменьшить нагрузку на центральный процессор.

Аудиокодек	Доступность на устройстве AIR for TV	Аппаратное декодирование	Ситуации использования аудиокодека	Дополнительная информация
AAC	Всегда	Всегда	В видео, закодированных с помощью H.264. Для потоковой передачи аудио, например в службах потоковой передачи музыки через Интернет.	Если выполняется только потоковая передача аудио AAC, закройте аудиопоток в контейнер MP4.
mp3	Всегда	Нет	Для звуков в SWF-файлах приложений. В видеороликах, закодированных с помощью Sorenson Spark или On2 VP6.	Видеоролик H.264, в котором используется аудио формата mp3, не воспроизводится на устройствах AIR for TV.
AC-3 (Dolby Digital) E-AC-3 (Enhanced Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Проверка	Да	В видео, закодированных с помощью H.264.	Как правило, AIR for TV передает многоканальный аудиопоток на внешний аудио- или видеоприемник, на котором происходит декодирование и воспроизведение аудио.
Speex	Всегда	Нет	Получение эфирного голосового потока.	Видеоролик H.264, в котором используется аудио формата Speex, не воспроизводится на устройствах AIR for TV. Используйте Speex только для видеороликов, закодированных с помощью Sorenson Spark или On2 VP6.
NellyMoser	Всегда	Нет	Получение эфирного голосового потока.	Видеоролик H.264, в котором используется аудио формата NellyMoser, не воспроизводится на устройствах AIR for TV. Используйте NellyMoser только для видеороликов, закодированных с помощью Sorenson Spark или On2 VP6.

***Примечание.** Некоторые видеофайлы содержат два аудиопотока. Например, видеофайл может содержать потоки AAC и AC3. AIR for TV не поддерживает такие видеофайлы и использование таких видеофайлов может привести к тому, что видео будет воспроизводиться без звука.*

Аппаратное ускорение графики

Использование аппаратного ускорения графики

Устройства AIR for TV предоставляют аппаратное ускорение для двумерных графических операций. Аппаратные ускорители графики на устройствах разгружают центральный процессор, выполняя следующие операции:

- растровая визуализация;
- растровое масштабирование;
- растровое наложение;

- сплошная заливка прямоугольника.

Использование такого аппаратного графического ускорения позволяет повысить производительность многих графических операций в приложениях AIR for TV. К таким операциям относятся:

- скользящие переходы;
- переходы с масштабированием;
- проявление и затухание;
- совмещение нескольких изображений с помощью альфа-блендинга.

Чтобы воспользоваться преимуществами аппаратного графического ускорения при выполнении таких операций, используйте один из следующих методов:

- Установите для свойства `cacheAsBitmap` значение `true` в объектах `MovieClip` и других отображаемых объектах, содержимое которых преимущественно не изменяется. Затем примените скользящий переход, переход с затуханием и альфа-блендинг для этих объектов.
- Используйте свойство `cacheAsBitmapMatrix` для отображаемых объектов, для которых требуется изменить масштаб или выполнить преобразование (выполните изменение позиции по оси `x` и `y`).

При использовании операций класса `Matrix` для масштабирования и преобразования операции выполняет аппаратный ускоритель устройства. В качестве альтернативного метода можно рассмотреть сценарий, когда изменяются размеры отображаемых объектов, для свойства `cacheAsBitmap` которого установлено значение `true`. При изменении размеров программное обеспечение среды выполнения перерисовывает растровое изображение. При перерисовке с помощью программного обеспечения производительность масштабирования ниже, чем при аппаратном ускорении с использованием операции `Matrix`.

Например, рассмотрите приложение, которое отображает изображение, разворачивающееся, когда пользователь выделяет его. Используйте операцию масштабирования `Matrix` несколько раз, чтобы создать иллюзию развертывания изображения. Однако в зависимости от размера исходного и конечного изображений качество итогового изображения может быть неприемлемым. Поэтому после операций развертывания выполняется сброс размеров отображаемого объекта. Поскольку для свойства `cacheAsBitmap` установлено значение `true`, программное обеспечение среды выполнения перерисовывает отображаемый объект, но только один раз, и отображает высококачественное изображение.

***Примечание.** Обычно устройства AIR for TV не поддерживают поворот и наклон с использованием аппаратного ускорения. Поэтому если в классе `Matrix` указан поворот и наклон, AIR for TV выполняет все операции `Matrix` в программно обеспечении. Такие программные операции могут негативно сказаться на производительности.*

- Используйте класс `BitmapData` для создания пользовательского поведения кэширования растрового изображения.

Дополнительные сведения о кэшировании растровых изображений см. в следующих документах:

- [Кэширование экранных объектов](#)
- [Кэширование растрового изображения](#)
- [Кэширование растрового изображения вручную](#)

Управление графической памятью

Для выполнения графических операций с ускорением в аппаратных ускорителях используется специальная графическая память. Если приложение использует всю графическую память, оно работает медленнее, так как AIR for TV выполняет все графические операции с помощью программного обеспечения.

Порядок управления использованием графической памяти в приложении

- После завершения работы с изображением или другими растровыми данными высвободите соответствующую графическую память. Для этого вызовите метод `dispose()` свойства `bitmapData` объекта `Bitmap`. Например:

```
myBitmap.bitmapData.dispose();
```

Примечание. При высвобождении ссылки на объект `BitmapData` освобождение графической памяти происходит не сразу. Сборщик мусора среды выполнения в конечном итоге высвобождает графическую память, однако вызов метода `dispose()` предоставляет больший контроль.

- Используйте `PerfMaster Deluxe`, приложение AIR от Adobe, чтобы лучше понять, как работает аппаратное графическое ускорение на целевых устройствах. Данное приложение показывает число кадров в секунду при выполнении различных операций. С помощью `PerfMaster Deluxe` можно сравнить разные реализации одной операции. Например, сравните перемещение растрового и векторного изображений. `PerfMaster Deluxe` можно загрузить со страницы «[Платформа Flash Platform для ТВ-устройств](#)».

Управление списком отображения

Чтобы сделать экранный объект невидимым, установите для свойства `visible` значение `false`. После этого объект по-прежнему находится в списке отображения, но среда AIR for TV не выполняет его визуализацию или вывод на экран. Этот метод рекомендуется использовать для объектов, которые часто появляются в области зрения и исчезают, так как это вызывает лишь незначительные затраты ресурсов на обработку. Однако присвоение свойству `visible` значения `false` не освобождает ресурсы объекта. Поэтому по завершении отображения объекта или по прошествии длительного времени после завершения работы с ним удалите объект из списка отображения. Кроме того, установите для всех ссылок на объект значение `null`. Эти действия позволяют сборщику мусора освободить ресурсы объекта.

Использование изображений PNG и JPEG

Форматы PNG и JPEG являются стандартными форматами изображений, используемых в приложениях. При использовании этих форматов изображений в приложениях AIR for TV учитывайте следующие моменты:

- Раскодирование файлов JPEG в AIR for TV обычно выполняется с помощью аппаратного ускорения.
- Раскодирование файлов PNG в AIR for TV обычно выполняется с помощью программного ускорения. Раскодирование PNG-файлов в программном обеспечении выполняется быстро.
- PNG является единственным межплатформенным форматом растровых изображений, поддерживающим прозрачность (альфа-канал).

Поэтому эти форматы изображений в приложениях следует использовать следующим образом:

- Используйте файлы JPEG для фотографий, чтобы воспользоваться преимуществами аппаратного ускорения.
- Используйте PNG-файлы для элементов пользовательского интерфейса. Элементы пользовательского интерфейса могут иметь настройки прозрачности, и программное раскодирование дает достаточно высокую производительность для визуализации элементов графического интерфейса.

Рабочая область в приложениях AIR for TV

Если при создании приложений AIR for TV используется класс `Stage`, учитывайте следующие моменты:

- разрешение экрана;
- безопасная область просмотра;

- режим масштабирования рабочей области;
- выравнивание рабочей области;
- состояние отображения рабочей области;
- разработка для нескольких размеров экрана;
- параметры качества рабочей области;

разрешение экрана.

В настоящее время ТВ-устройства обычно имеют одно из трех разрешений экрана: 540p, 720p или 1080p. Эти разрешения экрана дают следующие значения в классе ActionScript Capabilities:

Разрешение экрана	Capabilities.screenResolutionX	Capabilities.screenResolutionY
540p	960	540
720p	1280	720
1080p	1920	1080

При создании полноэкранных приложений AIR for TV жестко закодируйте `Stage.stageWidth` и `Stage.stageHeight` в соответствии с разрешением экрана устройства. Однако при создании полноэкранных приложений, которые могут работать на нескольких устройствах, используйте свойства `Capabilities.screenResolutionX` и `Capabilities.screenResolutionY`, чтобы задать размеры рабочей области.

Например:

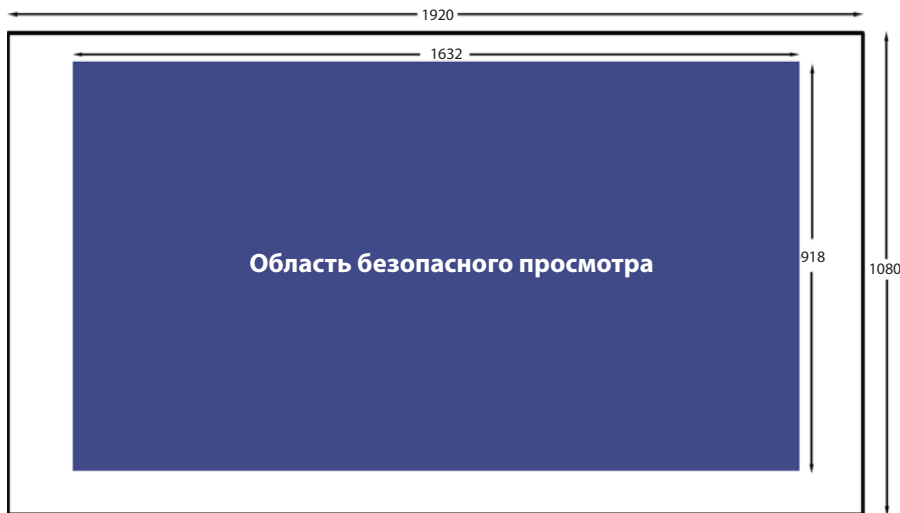
```
stage.stageWidth = Capabilities.screenResolutionX;  
stage.stageHeight = Capabilities.screenResolutionY;
```

Безопасная область просмотра

Безопасная область просмотра на телевизоре — это область экрана с учетом отступов от границ экрана. Доступ является достаточно большим, чтобы конечный пользователь мог видеть всю область, не скрытую рамкой телевизора. Необходимость отступа определяется наличием этой физической рамки, которая может быть разной у разных производителей. Безопасная область просмотра предназначена для того, чтобы гарантировать область экрана, которая будет видимой. Безопасную область просмотра также называют *безопасной областью титров*.

Переразвертка — это область экрана, которая скрыта рамкой.

Adobe рекомендует отступ в размере 7,5 % с каждой стороны экрана. Например:



Безопасная область просмотра для экранов с разрешением 1920 x 1080

При разработке приложений AIR for TV всегда учитывайте безопасную область просмотра.

- Используйте всю область экрана для фона, например для фоновое изображение или фоновое цвета.
- Для важных элементов приложения, таких как текст, графика, видео и элементы пользовательского интерфейса, включая кнопки, используйте только безопасную область просмотра.

В следующей таблице показаны размеры безопасной области просмотра для стандартных разрешений экрана с отступом 7,5 %.

Разрешение экрана	Ширина и высота безопасной области просмотра	Ширина отступа слева и справа	Высота отступа сверху и снизу
960 x 540	816 x 460	72	40
1280 x 720	1088 x 612	96	54
1920 x 1080	1632 x 918	144	81

Однако всегда рекомендуется динамически рассчитывать безопасную область просмотра. Например:

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;  
  
horizontalInset = .075 * Capabilities.screenResolutionX;  
verticalInset = .075 * Capabilities.screenResolutionY;  
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);  
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

Режим масштабирования рабочей области

Установите для свойства `Stage.scaleMode` значение `StageScaleMode.NO_SCALE` и прослушивайте события изменения размеров рабочей области.

```
stage.scaleMode = StageScaleMode.NO_SCALE;  
stage.addEventListener(Event.RESIZE, layoutHandler);
```

Данный параметр устанавливает координаты рабочей области равными пиксельным координатам. В сочетании с состоянием отображения `FULL_SCREEN_INTERACTIVE` и выравниванием рабочей области `TOP_LEFT` данный параметр обеспечивает эффективное использование безопасной области просмотра.

В частности в полноэкранных приложениях этот режим масштабирования означает, что свойства `stageWidth` и `stageHeight` класса `Stage` соответствуют свойствам `screenResolutionX` и `screenResolutionY` класса `Capabilities`.

Кроме того, при изменении размера окна приложения содержимое рабочей области сохраняет заданный размер. Среда выполнения не выполняет автоматической компоновки или масштабирования. Кроме того, при изменении размеров окна среда выполнения отправляет событие `resize` класса `Stage`. Поэтому вы можете полностью управлять корректировкой содержимого приложения при его запуске и изменении размеров окна приложения.

***Примечание.** Поведение `NO_SCALE` такое же, как в любом приложении AIR. Однако в приложениях AIR for TV, в которых используется этот параметр, крайне важно использовать безопасную область просмотра.*

Выравнивание рабочей области

Установите для свойства `Stage.align` значение `StageAlign.TOP_LEFT`:

```
stage.align = StageAlign.TOP_LEFT;
```

При таком выравнивании точка начала координат `0, 0` находится в левом верхнем углу экрана, что довольно удобно для размещения содержимого с использованием `ActionScript`.

В сочетании с режимом масштабирования `NO_SCALE` и состоянием отображения `FULL_SCREEN_INTERACTIVE` данный параметр обеспечивает эффективное использование безопасной области просмотра.

Состояние отображения рабочей области

Установите для параметра `Stage.displayState` полноэкранный приложения AIR for TV значение `StageDisplayState.FULL_SCREEN_INTERACTIVE`:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Данное значение предписывает приложению AIR развернуть рабочую область на весь экран, не отключая возможность ввода.

Adobe рекомендует использовать параметр `FULL_SCREEN_INTERACTIVE`. В сочетании с режимом масштабирования `NO_SCALE` и выравниванием рабочей области `TOP_LEFT` данный параметр обеспечивает эффективное использование безопасной области просмотра.

Поэтому для полноэкранных приложений в обработчике для события `ADDED_TO_STAGE` основного класса документа выполните следующие действия:

```
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;
}
```

Затем в обработчике события `RESIZE` выполните следующую операцию:

- Сравните размеры разрешения экрана с шириной и высотой рабочей области. Если они совпадают, произошло событие `RESIZE`, так как состояние отображения экрана изменилось на `FULL_SCREEN_INTERACTIVE`.

- Рассчитайте и сохраните размеры безопасной области просмотра и соответствующие отступы.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {
        // Calculate and save safe viewing area dimensions.
    }
}
```

Если рабочая область имеет размеры `Capabilities.screenResolutionX` и `screenResolutionY`, AIR for TV работает вместе с оборудованием для обеспечения самого высокого из возможных уровня точности видео и графики.

Примечание. Точность, с которой графика и видео отображаются на ТВ-экране, может отличаться от значений `Capabilities.screenResolutionX` и `screenResolutionY` в зависимости от устройства, на котором работает среда AIR for TV. Например, компьютерная приставка к телевизору, на которой выполняется среда AIR for TV, может иметь разрешение экрана 1280 x 720, а подключенный телевизор — разрешение 1920 x 1080. Однако среда AIR for TV задает обеспечение самого высокого из возможных уровня точности аппаратным обеспечением. Поэтому в этом примере аппаратное обеспечение отображает видео 1080p с экраным разрешением 1920 x 1080.

Разработка для нескольких размеров экрана

Полноэкранные приложения AIR for TV можно создавать таким образом, чтобы они хорошо работали и смотрелись на разных устройствах AIR for TV. Выполните следующие действия.

- 1 Установите для свойств рабочей области `scaleMode`, `align` и `displayState` рекомендованные значения: `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` и `StageDisplayState.FULL_SCREEN_INTERACTIVE` соответственно.
- 2 Задайте безопасную область просмотра на основе `Capabilities.screenResolutionX` и `Capabilities.screenResolutionY`.
- 3 Настройте размер и расположение содержимого в соответствии с шириной и высотой безопасной области просмотра.

Несмотря на то, что объекты содержимого имеют большой размер, особенно по сравнению с объектами в приложениях для мобильных устройств, применяются те же самые концепции динамического размещения, относительного позиционирования и адаптивного содержимого. Дополнительные сведения о поддержке этих концепций в ActionScript см. в статье [«Создание мобильного содержимого Flash для разных размеров экранов»](#).

Качество рабочей области

Свойство `Stage.quality` для приложения AIR for TV всегда имеет значение `StageQuality.High`. Его нельзя изменить.

Это свойство определяет качество визуализации всех объектов Stage.

Обработка ввода с пульта управления

Взаимодействие пользователей с приложением AIR for TV обычно осуществляется с помощью пульта управления. Однако нажатие клавиш следует обрабатывать таким же образом, как нажатие клавиш на клавиатуре в настольных приложениях. В частности, обрабатывается событие `KeyboardEvent.KEY_DOWN`. Дополнительные сведения см. в разделе «Захват действий клавиатуры» в *Руководстве разработчика по ActionScript 3.0*.

Клавиши на пульте управления сопоставлены с константами ActionScript. Например, клавиши навигации на пульте управления сопоставлены следующим образом:

Клавиша навигации на пульте управления	Константа ActionScript 3.0
Вверх	<code>Keyboard.UP</code>
Вниз	<code>Keyboard.DOWN</code>
Влево	<code>Keyboard.LEFT</code>
Вправо	<code>Keyboard.RIGHT</code>
ОК или выбор	<code>Keyboard.ENTER</code>

В AIR 2.5 было добавлено много других констант `Keyboard` для поддержки ввода с пульта управления. Полный список констант см. в разделе «Класс `Keyboard`» в *Справочнике ActionScript 3.0 для платформы Adobe Flash Platform*.

Для того чтобы обеспечить работу приложения на как можно большем количестве устройств, Adobe рекомендует следующее:

- По возможности используйте только клавиши навигации.
Разные пульты управления имеют разный набор клавиш. Однако обычно на всех пультах имеются клавиши навигации.
Например, на пульте управления плеера Blu-ray обычно нет клавиш переключения каналов. Даже клавиши для воспроизведения, паузы и остановки присутствуют не на всех пультах управления.
- Используйте клавиши меню и информации, если приложению требуются другие клавиши помимо клавиш навигации.
Клавиши меню и информации являются наиболее распространенными клавишами на пульте управления.
- Рассмотрите возможность использования универсальных пультов управления.
Даже когда создаются приложения для определенного устройства, следует учитывать, что во многих случаях не используются пульты управления, входящие в комплект устройства. Вместо этого используются универсальные пульты управления. Кроме того, при программировании универсальных пультов функции клавиш не всегда соответствуют их функциям на пульте управления устройством. Поэтому рекомендуется использовать только стандартные клавиши.
- Убедитесь, что пользователь всегда сможет использовать для выхода одну из клавиш навигации.
Иногда в приложении требуется использовать клавиши, которые могут встречаться не на всех пультах управления. Если одна из клавиш навигации будет использоваться в качестве запасного выхода, приложение будет элегантно работать на всех устройствах.
- Не требуйте ввода с помощью указывающего устройства, если нет уверенности в том, что целевое устройство AIR for TV имеет возможность ввода с помощью указывающего устройства.

Тогда как многие компьютерные приложения ожидают ввода с помощью мыши, большинство телевизоров не поддерживают ввод с помощью указывающего устройства. Поэтому если вы преобразуете компьютерные приложения, чтобы их можно было запускать на ТВ-устройствах, не забудьте изменить приложение, чтобы исключить ввод с помощью мыши. В этом случае необходимо изменить обработку событий и инструкции для пользователя. Например, в окне запуска приложения не должно быть текста «Нажмите, чтобы начать».

Управление фокусом

Когда в настольных приложениях фокус имеет элемент пользовательского интерфейса, он является целевым объектом событий ввода пользователя, таких как события клавиатуры и мыши. Кроме того, приложение выделяет находящийся в фокусе элемент интерфейса. Управление фокусом в приложениях AIR for TV отличается от управления фокусом в настольных приложениях. Это связано со следующими причинами:

- В настольных приложениях для перемещения фокуса в следующий элемент пользовательского интерфейса часто используется клавиша табуляции. В приложениях AIR for TV клавиша табуляции не используется. На пультах управления клавиши табуляции обычно нет. Поэтому использовать свойство `tabEnabled` объекта `DisplayObject` для управления фокусом, как в настольных приложениях, не получится.
- В настольных приложениях часто предполагается, что для перемещения фокуса между элементами интерфейса, используется мышь.

Поэтому в приложении необходимо выполнить следующее:

- Добавьте прослушиватель события для объекта `Stage`, который будет прослушивать события `Keyboard`, такие как `KeyboardEvent.KEY_DOWN`.
- Разработайте логику приложения, чтобы определить, какой интерфейс элемента следует выделять для конечного пользователя. Не забудьте выделить элемент интерфейса при запуске приложения.
- В соответствии с логикой приложения отправьте событие `Keyboard`, которое получил объект `Stage`, в соответствующий объект элемента интерфейса.

Кроме того, для назначения фокуса для элементов интерфейса можно использовать методы `Stage.focus` и `Stage.assignFocus()`. Затем можно добавить прослушиватель событий для этого объекта `DisplayObject`, чтобы он получал события клавиатуры.

Разработка пользовательского интерфейса

Чтобы пользовательский интерфейс приложений AIR for TV хорошо работал на телевизионных устройствах, следует соблюдать рекомендации в отношении следующих критериев:

- способность приложения к реагированию;
- удобство приложения в использовании;
- личность и ожидания пользователя.

Способность к реагированию

Воспользуйтесь следующими советами, чтобы обеспечить максимальную скорость реагирования приложений AIR for TV.

- Размер исходного SWF-файла приложения должен быть как можно меньше.

В исходном SWF-файле приложения загружайте только ресурсы, которые требуются для запуска приложения. Например, загружайте только изображение окна запуска приложения.

Данные рекомендации относятся также и к настольным приложениям AIR, однако намного важнее их соблюдать на устройствах AIR for TV. Например, вычислительная мощность устройств AIR for TV существенно отличается от производительности настольных компьютеров. Кроме того, приложения хранятся во flash-памяти, скорость доступа к которой ниже, чем для жестких дисков настольных компьютеров.

- Добейтесь, чтобы приложение работало со скоростью, по крайней мере, 20 кадров в секунду.

Разработайте графику таким образом, чтобы достичь этой цели. Сложность графических операций может привести к снижению частоты кадров. Советы по повышению производительности см. в разделе [«Оптимизация производительности для платформы Adobe Flash Platform»](#).

***Примечание.** Графическое оборудование устройств AIR for TV, обычно обновляет экран с частотой 60 Гц или 120 Гц (60 или 120 раз в секунду). Оборудование выполняет сканирование обновлений рабочей области, например со скоростью 30 кадров в секунду или 60 кадров в секунду при частоте обновления 60 Гц и 120 Гц. Однако возможность достижения более высокой частоты кадров зависит от сложности графических операций приложения.*

- Обновляйте экран в течение 100-200 миллисекунд после ввода пользователя.

Пользователи теряют терпение, если обновление выполняется долго, что может приводить к многократному повторному нажатию клавиш.

Удобство в использовании

Пользователи приложений AIR for TV находятся в среде «гостиной». Они сидят примерно в 3 метрах от телевизора. Иногда свет в комнате может быть темно. Для ввода они обычно используют пульт управления. Приложением может пользоваться несколько человек, иногда они работают вместе, иногда — поочередно.

Поэтому чтобы пользовательский интерфейс было удобно использовать на ТВ-устройствах, необходимо учесть следующие моменты:

- Элементы пользовательского интерфейса должны иметь достаточно большой размер.

При разработке текста, кнопок и других элементов интерфейса учитывайте, что пользователи находятся в другом конце комнаты. Все элементы должны быть видимыми и легко читаемыми на расстоянии, например, 3 метра. Не загромождайте экран, рассчитывая на то, что он имеет большой размер.

- Используйте хороший контраст, чтобы содержимое было хорошо видно и легко читалось в другом конце комнаты.
- Элементы интерфейса, находящиеся в фокусе, следует выделить, чтобы фокус был легко заметен.
- Используйте движение только в случаях необходимости. Например, хорошо может работать переход со скольжением с одного экрана на другой. Однако движение может отвлекать, если оно не помогает пользователю при навигации или если оно не требуется для работы приложения.
- Всегда предусматривайте очевидный способ возврата с помощью элементов интерфейса.

Дополнительные сведения об использовании пульта управления см. в разделе [«Обработка ввода с пульта управления»](#) на странице 140».

Личность и ожидания пользователя.

Помните, что пользователи приложений AIR for TV обычно ожидают получить развлекательную и расслабляющую среду, свойственную телевизионным устройствам. Они не обязательно будут знакомы с компьютерами и технологиями.

Поэтому при разработке приложений AIR for TV необходимо учитывать следующее:

- Не используйте технические термины.
- Избегайте модульных диалогов.
- Используйте неформальные инструкции, подходящие для условий гостиной, а не для рабочей или технической среды.
- Используйте графику, которая обеспечит высокое качество изображения на ТВ, соответствующее ожиданиям зрителей.
- Реализуйте пользовательский интерфейс, который легко работает с пультом дистанционного управления. Не применяйте пользовательский интерфейс или элементы дизайна, которые больше подходят для компьютерного или мобильного приложения. Например, пользовательские интерфейсы на компьютерах или мобильных устройствах часто содержат кнопки для нажатия с помощью мыши или пальца.

Шрифты и текст

В приложении AIR for TV можно использовать шрифты устройства или встроенные шрифты.

Шрифты устройства — это шрифты, которые установлены на устройстве. На всех устройствах AIR for TV имеются следующие шрифты устройств:

Имя шрифта	Описание
_sans	Шрифт устройства _sans — это шрифт типа sans-serif. Шрифт устройства _sans , установленный на всех устройствах AIR for TV, — это Myriad Pro. Обычно шрифт sans-serif выглядит на телевизорах лучше, чем шрифты serif, вследствие большого расстояния от экрана.
_serif	Шрифт устройства _serif — это шрифт типа serif. Шрифт устройства _serif , установленный на всех устройствах AIR for TV, — это Minion Pro.
_typewriter	Шрифт устройства _typewriter — это моноширинный шрифт. Шрифт устройства _typewriter , установленный на всех устройствах AIR for TV, — это Courier Std.

На всех устройствах AIR for TV имеются следующие азиатские шрифты устройств:

Имя шрифта	Язык	Категория типа шрифта	Код языка
RyoGothicPlusN-Regular	Японский	sans	ja
RyoTextPlusN-Regular	Японский	serif	ja
AdobeGothicStd-Light	Корейский	sans	ko
AdobeHeitiStd-Regular	Китайский (упрощенный)	sans	zh_CN
AdobeSongStd-Light	Китайский (упрощенный)	serif	zh_CN
AdobeMingStd-Light	Китайский (традиционный)	serif	zh_TW и zh_HK

Это шрифты устройств AIR for TV:

- доступны в библиотеке Adobe® Type Library;
- хорошо смотрятся на ТВ-устройствах;
- предназначены для титров в видео;
- являются контурами шрифтов, а не растровыми шрифтами.

Примечание. Производители устройств часто устанавливают на устройствах другие шрифты. Такие шрифты производителей дополняют шрифты устройств AIR for TV.

Adobe предоставляет приложение FontMaster Deluxe, которое позволяет отобразить все шрифты, доступные на устройстве. Приложение доступно на странице [«Платформа Flash Platform для ТВ-устройств»](#).

В приложения AIR for TV также можно встраивать шрифты. Дополнительные сведения о встроенных шрифтах см. в разделе [«Расширенная визуализация текста»](#) в *Руководстве разработчика по ActionScript 3.0*.

Рекомендации Adobe при использовании текстовых полей TLF:

- Используйте текстовые поля TLF для текста на азиатских языках, чтобы воспользоваться преимуществами локали, в которой запускается приложение. Задайте свойство `locale` объекта `TextLayoutFormat`, связанного с объектом `TLFTextField`. Инструкции по определению текущей локали см. в разделе [«Выбор локали»](#) в *Руководстве разработчика по ActionScript 3.0*.
- Укажите имя шрифта с помощью свойства `fontFamily` объекта `TextLayoutFormat`, если используется шрифт, не являющийся шрифтом устройств AIR for TV. Среда AIR for TV использует шрифт, если он доступен на устройстве. Если запрошенного шрифта на устройстве нет, среда AIR for TV заменяет его на подходящий шрифт устройства в соответствии с настройками параметра `locale`.
- Установите значение `_sans`, `_serif`, или `_typewriter` для свойства `fontFamily`, а также настройте свойство `locale`, чтобы среда AIR for TV могла выбрать правильный шрифт устройства. В зависимости от локали среда AIR for TV выбирает шрифт из списка азиатских или неазиатских шрифтов устройства. Данные настройки позволяют легко автоматизировать выбор правильного шрифта для четырех основных азиатских локалей и английского языка.

Примечание. Если для текста на азиатском языке используется классическое текстовое поле, для корректной визуализации текста необходимо указать имя шрифта устройства AIR for TV. Если вам известно, что на целевом устройстве установлен другой шрифт, можно также указать имя этого шрифта.

Моменты, которые следует учитывать в отношении производительности приложения:

- Классические текстовые поля обеспечивают более высокую производительность, чем текстовые поля TLF.
- Для классических текстовых полей используются растровые шрифты, которые обеспечивают максимальную производительность.

Растровые шрифты предоставляют растровое изображение для каждого символа, тогда как контурные шрифты предоставляют только данные о контурах вокруг каждого символа. Как шрифты устройства, так и встроенные шрифты могут быть растровыми.

- Если вы указали шрифт устройства, убедитесь, что он установлен на целевом устройстве. Если шрифт не установлен на устройстве, среда AIR for TV выберет другой шрифт из списка установленных. Однако это поведение снижает производительность приложения.

- Как и в случае любых других отображаемых объектов, если объект `TextField` изменяется редко, установите для свойства объекта `cacheAsBitmap` значение `true`. Это позволит повысить производительность при таких преобразованиях, как затухание, скольжение и альфа-наложение. Для масштабирования и преобразования используйте `cacheAsBitmapMatrix`. Дополнительные сведения см. в разделе «Аппаратное ускорение графики» на странице 133.

Безопасность файловой системы

Приложения AIR for TV являются приложениями AIR, поэтому они имеют доступ к файловой системе устройства. Однако для устройств «в гостиной» крайне важно, чтобы приложения не имели доступа к системным файлам устройства и файлам других приложений. Пользователи телевизионных и соответствующих устройств не ожидают и не допускают возможности сбоев в работе устройства: в конце концов они смотрят телевизор.

Поэтому доступ приложений AIR for TV к файловой системе устройства ограничен. При использовании ActionScript 3.0 приложение может обращаться только к определенным каталогам (и их подкаталогам). Кроме того, имена каталогов, которые используются в ActionScript, не являются реальными именами каталогов на устройстве. Этот дополнительный уровень позволяет защитить приложения AIR for TV от злоумышленного и случайного обращения к локальным файлам, которые не принадлежат приложениям.

Дополнительные сведения см. в разделе «Вид каталога приложений AIR for TV».

Изолированная программная среда приложения AIR

Запуск приложений AIR for TV в изолированной программной среде приложений AIR описан в разделе «Изолированная программная среда приложений AIR».

Единственное отличие приложений AIR for TV заключается в том, что они имеют ограниченный доступ к файловой системе, как описано в разделе «Безопасность файловой системы» на странице 145».

Цикл разработки приложений

В отличие от настольных приложений пользователь не может закрыть окно, в котором выполняется приложение AIR for TV. Поэтому в пользовательском интерфейсе необходимо предусмотреть механизм выхода из приложения.

Обычно устройство разрешает пользователю безусловный выход из приложения по нажатию клавиши выхода на пульте управления. Однако среда AIR for TV не отправляет в приложение событие `flash.events.Event.EXITING`. Поэтому необходимо регулярно сохранять состояние приложения, чтобы при последующем запуске его можно было восстановить.

Файлы cookie HTTP

AIR for TV поддерживает постоянные файлы cookie и файлы cookie сеанса HTTP. AIR for TV сохраняет файлы cookie для каждого приложения AIR в специальной папке:

```
/app-storage/<app id>/Local Store
```

Именем файла cookie является `cookies`.

Примечание. Среда AIR на других устройствах, таких как настольные устройства, не сохраняет файлы cookie отдельно для каждого приложения. Хранилище файлов cookie приложения поддерживает модель безопасности приложения и системы AIR for TV.

Используйте свойство `ActionScript URLRequest.manageCookies` следующим образом:

- Задайте для свойства `manageCookies` значение `true`. Данное значение установлено по умолчанию. Это означает, что среда AIR for TV автоматически добавляет файлы cookie для запросов HTTP и запоминает файлы cookie, полученные в ответе HTTP.

Примечание. Даже когда для свойства `manageCookies` установлено значение `true`, приложение может вручную добавлять файлы cookie для запросов HTTP с помощью `URLRequest.requestHeaders`. Если эти файлы cookie должны соответствовать файлам cookie, которыми управляет среда AIR for TV, запрос содержит два файла cookie с одним именем. Значения двух файлов cookie могут отличаться.

- Задайте для свойства `manageCookies` значение `false`. Это значение означает, что приложение выполняет отправку файлов cookie по HTTP-запросам и запоминает файлы cookie, полученные в ответе HTTP.

Дополнительные сведения см. в документе [URLRequest](#).

Процедура разработки приложений AIR for TV

Приложения AIR for TV можно создавать с помощью следующих инструментов платформы Adobe Flash Platform:

- Adobe Flash Professional

Adobe Flash Professional CS5.5 поддерживает AIR 2.5 для ТВ-устройств, первую версию среды AIR, поддерживающую приложения AIR for TV.

- Adobe Flash® Builder®

Flash Builder 4.5 поддерживает AIR 2.5 для ТВ-устройств.

- AIR SDK

Начиная с версии AIR 2.5, разработку приложений можно выполнять с помощью инструментов командной строки, включенных в пакет AIR SDK. Загрузить AIR SDK можно на странице <http://www.adobe.com/ru/products/air/sdk/>.

Использование Flash Professional

Использование Flash Professional для разработки, тестирования и публикации приложений AIR for TV аналогично использованию инструмента для настольных приложений AIR.

Однако при написании кода ActionScript 3.0 следует использовать только классы и методы, которые поддерживают профили AIR `tv` и `extendedtv`. Дополнительные сведения см. в разделе «[Профили устройств](#)» на странице 263».

Настройки проекта

Выполните следующие действия, чтобы настроить проект для приложений AIR for TV.

- На вкладке Flash в диалоговом окне «Параметры публикации» установите для параметра Player значение AIR 2.5 (или более позднюю версию).
- На вкладке «Общие» в диалоговом окне «Параметры AIR» («Настройки приложения и установщика») установите профиль `tv` или `extended TV`.

Отладка

Приложение можно запустить с помощью инструмента AIR Debug Launcher, который включен в Flash Professional. Выполните следующие действия.

- Чтобы запустить приложение в режиме отладки, выберите:
«Отладка» > «Отладка ролика» > «В AIR Debug Launcher (стандартная версия)»
После этого при последующем запуске отладки можно будет выбирать:
«Отладка» > «Отладка ролика» > «Отладка»
- Чтобы запустить приложение без возможностей отладки, выберите:
«Управление» > «Тестировать ролик» > «В AIR Debug Launcher (стандартная версия)»
После этого при последующих запусках можно будет выбирать «Управление» > «Тестировать ролик» > «Проверка».

Поскольку установлен профиль TV или extended TV, AIR Debug Launcher предоставляет меню «Кнопки пульта управления». Это меню можно использовать для моделирования нажатия клавиш на пульте управления.

Дополнительные сведения см. в разделе «[Удаленная отладка с помощью Flash Professional](#)» на странице 156.

Использование собственных расширений

Если в вашем приложении используется собственное расширение, следуйте инструкциям в разделе «[Список задач для использования собственного расширения](#)» на странице 161.

Однако при использовании собственных расширений:

- Flash Professional нельзя использовать для публикации приложения. Приложения следует публиковать с помощью ADT. См. раздел «[Упаковка с помощью ADT](#)» на странице 152».
- Flash Professional нельзя использовать для выполнения или отладки приложения. Для выполнения отладки приложения на рабочем компьютере используйте ADL. См. раздел «[Моделирование устройства с помощью ADL](#)» на странице 154».

Использование Flash Builder

Начиная с версии Flash Builder 4.5, Flash Builder поддерживает разработку с использованием AIR for TV. Использование Flash Builder для разработки, тестирования и публикации приложений AIR for TV аналогично использованию инструмента для настольных приложений AIR.

Настройка приложения

Убедитесь в соблюдении следующих условий:

- Приложение использует элемент `Application` в качестве класса контейнера в файле MXML, если используется файл 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>
```

Важная информация. Приложения AIR for TV не поддерживают элемент `WindowedApplication`.

Примечание. Использовать файл MXML не обязательно. Вместо этого можно создать проект ActionScript 3.0.

- Используйте только классы и методы ActionScript 3.0, которые поддерживают профили AIR tv и extendedTV. Дополнительные сведения см. в разделе «[Профили устройств](#)» на странице 263.

Кроме того, в файле приложения XML необходимо выполнить следующее:

- Для атрибута xmlns элемента application необходимо установить значение «AIR 2.5»:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```
- Для элемента supportedProfiles установлено значение tv или extendedTV:

```
<supportedProfiles>tv</supportedProfiles>
```

Отладка приложения

Приложение можно запустить с помощью инструмента AIR Debug Launcher, который включен в Flash Builder. Выполните следующие действия.

- 1 Выберите «Выполнить» > «Конфигурации отладки».
- 2 Убедитесь, что в поле «Профиль» установлено значение «Настольный».
- 3 Выберите «Выполнить» > «Отладка», чтобы запустить приложение в режиме отладки, или «Выполнить» > «Выполнить», чтобы запустить приложение без возможностей режима отладки.

Поскольку для элемента supportedProfiles установлено значение «TV» или «extended TV», AIR Debug Launcher предоставляет меню «Кнопки пульта управления». Это меню можно использовать для моделирования нажатия клавиш на пульте управления.

Дополнительные сведения см. в разделе «[Удаленная отладка с помощью Flash Builder](#)» на странице 156.

Использование собственных расширений

Если в вашем приложении используется собственное расширение, следуйте инструкциям в разделе «[Список задач для использования собственного расширения](#)» на странице 161.

Однако при использовании собственных расширений:

- Flash Builder нельзя использовать для публикации приложения. Приложения следует публиковать с помощью ADT. См. раздел «[Упаковка с помощью ADT](#)» на странице 152.
- Flash Builder нельзя использовать для выполнения или отладки приложения. Для выполнения отладки приложения на рабочем компьютере используйте ADL. См. раздел «[Моделирование устройства с помощью ADL](#)» на странице 154.

Свойства дескриптора приложений AIR for TV

Как и при разработке других приложений AIR в файле дескриптора приложения задаются базовые свойства приложения. Приложения с профилем ТВ-устройств игнорируют некоторые настройки, относящиеся к настольным системам, например размер окна и прозрачность. В приложениях для целевых устройств с профилем extendedTV можно использовать собственные расширения. Используемые собственные расширения идентифицируются в элементе extensions.

Общие настройки

Некоторые настройки в дескрипторе приложения важны для всех приложений для ТВ-устройств.

Требуемая версия среды выполнения AIR

Укажите версию среды выполнения AIR, которая требуется для приложения, использующего пространство имен файла дескриптора приложения.

Пространство имен, назначенное элементом `application`,

Например, рассмотрите приложение, в котором используется пространство имен AIR 2.5, однако у пользователя установлена более поздняя версия. В этом случае приложение по-прежнему будет видеть поведение AIR 2.5, даже если поведение будущей версии AIR отличается. Приложение получит доступ к новому поведению и функциям только после изменения пространства имен и публикации обновления. Исправления безопасности являются важным исключением из этого правила.

Укажите пространство имен с помощью атрибута `xmlns` корневого элемента `application`:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5 — это первая версия AIR, поддерживающая приложения для ТВ-устройств.

Идентификация приложения

Некоторые настройки публикуемых приложений могут быть уникальными. К этим настройкам относятся элементы `id`, `name` и `filename`.

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

Версия приложения

Укажите версию приложения с помощью элемента `versionNumber`. При указании значения для элемента `versionNumber` можно использовать последовательность из трех цифр, разделенных точками, например «0.1.2». Каждый сегмент номера версии может содержать до трех цифр (то есть максимальным номером версии может быть «999.999.999»). Номер не обязательно должен включать все три сегмента. Также допускается устанавливать версии «1» и «1.0».

Кроме того, можно задать метку для версии с помощью элемента `versionLabel`. Если добавлена метка версии, она будет показана вместо номера версии.

```
<versionNumber>1.23.7</versionNumber>  
<versionLabel>1.23 Beta 7</versionLabel>
```

Основной SWF-файл приложения

Укажите основной SWF-файл приложения в дочернем элементе `versionLabel` элемента `initialWindow`. Если целевое устройство имеет профиль ТВ-устройства, необходимо использовать SWF-файл (приложения на основе HTML не поддерживаются).

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

Файл должен быть включен в пакет AIR (с помощью ADT или ИСР). Если только указать имя в дескрипторе приложения, это не позволит автоматически включить файл в пакет.

Свойства главного окна

Исходный вид и поведение главного окна приложения определяют несколько дочерних элементов элемента `initialWindow`. Несмотря на то что большинство этих свойств игнорируются на устройствах с ТВ-профилем, можно использовать элемент `fullScreen`:

- `fullScreen` — указывает, заполняет ли приложение всю область отображения на устройстве или на экране также отображаются системные элементы, например обычный хром операционной системы.

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

Элемент `visible`

Элемент `visible` является дочерним элементом для элемента `initialWindow`. Элемент `initialWindow` AIR for TV игнорирует элемент `visible`, так как содержимое приложения всегда является видимым на устройствах AIR for TV.

Однако можно задать элементу `visible` значение `true`, если приложение также предназначено для компьютеров.

На компьютерах этот элемент по умолчанию имеет значение `false`. Поэтому если не включить элемент `visible`, содержимое приложения не будет отображаться на компьютерах. Хотя можно использовать класс `ActionScript NativeWindow`, чтобы сделать содержимое видимым на компьютерах, профили для ТВ-устройств не поддерживают класс `NativeWindow`. Если попытаться использовать класс `NativeWindow` в приложении, выполняемом на устройстве AIR for TV, приложение не загрузится. Независимо от того, вызывается ли метод класса `NativeWindow`, приложение, в котором используется этот класс, не загружается на устройстве AIR for TV.

Поддерживаемые профили

Если приложение предназначено только для телевизионных устройств, можно запретить его установку на другие типы устройств. Исключите другие профили из списка поддерживаемых в элементе `supportedProfiles`:

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Если в приложении используется собственное расширение, включите только профиль `extendedTV` в список поддерживаемых профилей:

```
<supportedProfiles>extendedTV</supportedProfiles>
```

Если элемент `supportedProfiles` не указан, предполагается, что приложение поддерживает все профили.

Не включайте *только* профиль `tv` в список `supportedProfiles`. Некоторые ТВ-устройства всегда выполняют AIR for TV в режиме, который соответствует профилю `extendedTV`. Это позволяет AIR for TV выполнять приложения, в которых используются собственные расширения. Если в элементе `supportedProfiles` указан только профиль `tv`, этим объявляется, что содержимое несовместимо с режимом AIR for TV для профиля `extendedTV`. По этой причине некоторые ТВ-устройства не загружают приложения, в которых указан только профиль `tv`.

Список классов `ActionScript`, поддерживаемых в профилях `tv` и `extendedTV`, представлен в разделе «[Возможности различных профилей](#)» на странице 265.

Необходимые собственные расширения

Приложения, поддерживающие профиль `extendedTV`, могут использовать собственные расширения.

Объявите все собственные расширения, которые использует приложение AIR, в дескрипторе приложения, используя элементы `extensions` и `extensionID`. В следующем примере показан синтаксис, применяемый для указания требуемых собственных расширений:

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

Если расширение не указано в списке, приложение не может его использовать.

Значение элемента `extensionID` совпадает со значением элемента `id` в файле дескриптора расширения. Файл дескриптора расширения — это файл XML с именем `extension.xml`. Он упакован в ANE-файл, получаемый от производителя устройства.

Если расширение перечислено в элементе `extensions`, но не установлено на устройстве AIR for TV, приложение не сможет работать. Исключением из этого правила является случай, когда ANE-файл, упакованный с приложением AIR for TV имеет фиктивный модуль расширения. В таком случае приложение может выполняться и использовать фиктивный модуль расширения. Фиктивная версия содержит код ActionScript без собственного кода.

Значки приложения

Требования к значкам приложений для телевизионных устройств зависят от конкретных устройств. Например, производитель устройств указывает следующую информацию:

- Требуемые значки и размеры значков.
- Требуемые типы файлов и правила именования.
- Порядок предоставления значков для приложения, например, требуется ли включать значки в пакет приложения.
- Необходимость указания значков в элементе `icon` в файле дескриптора приложения.
- Поведение в случае, когда приложение не предоставляет значков.

Для получения дополнительной информации свяжитесь с производителем устройства.

Игнорируемые настройки

Приложения на телевизионных устройствах игнорируют настройки приложения, которые относятся к мобильным устройствам, собственным окнам или функциям операционной системы настольного компьютера. Игнорируются следующие настройки:

- `allowBrowserInvocation`
- `aspectRatio`
- `autoOrients`
- `customUpdateUI`
- `fileTypes`
- `height`
- `installFolder`
- `maximizable`
- `maxSize`

- minimizable
- minSize
- programMenuFolder
- renderMode
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

Упаковка приложений AIR for TV

Упаковка с помощью ADT

Упаковку приложения AIR for TV можно выполнить с помощью инструмента командной строки AIR ADT. Начиная с версии AIR SDK 2.5, инструмент ADT поддерживает упаковку приложений для ТВ-устройств. Перед упаковкой скомпилируйте все коды ActionScript и MXML. Кроме того, требуется сертификат для подписи кода. Сертификат можно создать с помощью команды ADT -certificate.

Подробная информация о командах и параметрах ADT представлена в разделе «[AIR Developer Tool \(ADT\)](#)» на странице 176».

Создание пакета AIR

Для создания пакета AIR используйте команду ADT package:

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

Например, предположим следующее:

- Путь к инструменту ADT указан в определении пути оболочки командной строки. (См. раздел «[Переменные среды Path](#)» на странице 327».)
- Сертификат codesign.p12 расположен в родительском каталоге, из которого запускается команда ADT.

Запустите команду из родительского каталога, содержащего файлы приложения. В данном примере представлены файлы приложений myApp-app.xml (файл дескриптора приложения), myApp.swf и каталог значков.

Если приложение запускается, как показано в примере, ADT предложит ввести пароль для хранилища ключей. Вводимые символы паролей отображаются не во всех оболочках. После завершения ввода просто нажмите клавишу Enter. Также можно использовать параметр storepass, чтобы включить пароль в команду ADT.

Создание пакета AIRN

Если в приложении AIR for TV используется собственное расширение, вместо пакета AIR создайте пакет AIRN. Чтобы создать пакет AIRN, используйте команду ADT package, задав для аргумента target значение airn.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml  
myApp.swf icons -extdir C:\extensions
```

Например, предположим следующее:

- Путь к инструменту ADT указан в определении пути оболочки командной строки. (См. раздел «[Переменные среды Path](#)» на странице 327».)
- Сертификат codesign.p12 расположен в родительском каталоге, из которого запускается команда ADT.
- Параметр -extdir задает имя каталога, в котором содержатся ANE-файлы, используемые приложением. ANE-файлы содержат только версию с фрагментом или моделью для расширения ActionScript. На устройстве AIR for TV установлена версия расширения, содержащая исходный код.

Запустите команду из родительского каталога, содержащего файлы приложения. В данном примере представлены файлы приложений myApp-app.xml (файл дескриптора приложения), myApp.swf и каталог значков.

Если приложение запускается, как показано в примере, ADT предложит ввести пароль для хранилища ключей. Вводимые символы паролей отображаются не во всех оболочках. После завершения ввода просто нажмите клавишу Enter. Также можно использовать параметр storepass, чтобы включить в команду пароль.

Кроме того, для приложения AIR for TV, в котором используются собственные расширения, можно создать файл AIRI. Файл AIRI отличается от файла AIRN только тем, что не имеет подписи. Например:

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Файл AIRN можно будет создать на основе файла, когда приложение будет готово для подписания:

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Дополнительные сведения см. в разделе [Разработка собственных расширений для Adobe AIR](#).

Создание пакетов приложений с помощью Flash Builder или Flash Professional

С помощью Flash Professional и Flash Builder пакеты AIR можно публиковать и экспортировать без запуска ADT. Процедура создания пакетов AIR для приложений AIR описана в документации для этих программ.

В настоящее время только инструмент ADT позволяет создавать пакеты AIRN, то есть пакеты приложений AIR for TV, в которых используются собственные расширения.

Дополнительные сведения см. в следующих разделах:

- [Упаковка приложений AIR с использованием Flash Builder](#)
- [Публикация для Adobe AIR с использованием Flash Professional](#)

Отладка приложений AIR for TV

Моделирование устройства с помощью ADL

Самым простым и быстрым способом тестирования и отладки функций приложений является запуск приложения на рабочем компьютере с использованием утилиты Adobe Debug Launcher (ADL).

ADL определяет используемый профиль на основании элемента `supportedProfiles`, который указан в дескрипторе приложения. В частности:

- Если указано несколько профилей, ADL использует первый профиль из списка.
- Используя параметр `ADL -profile`, можно выбрать другой профиль из списка `supportedProfiles`.
- Если элемент `supportedProfiles` не включен в дескриптор приложения, для аргумента `-profile` может быть указан любой профиль.

Например, следующая команда позволяет запустить приложение для моделирования профиля `tv`:

```
adl -profile tv myApp-app.xml
```

При моделировании профиля `tv` или `extendedTV` на настольном компьютере с помощью ADL приложение запускается в среде, которая наиболее точно соответствует среде целевого устройства. Например:

- API-интерфейсы `ActionScript`, которые не являются частью профиля в аргументе `-profile`, будут недоступны.
- ADL разрешает ввод из элементов управления вводом устройства, таких как пульты управления, через меню команд.
- Укажите значение `tv` или `extendedTV` для аргумента `-profile`, чтобы смоделировать с помощью ADL класс `StageVideo` на настольном компьютере.
- Укажите значение `extendedTV` для аргумента `-profile`, чтобы разрешить приложению использовать фрагмент или модель собственного расширения, включенные в файл `AIRN` приложения.

Однако в связи с тем, что приложение ADL запускается на настольном компьютере, при тестировании приложений AIR for TV имеются некоторые ограничения.

- Производительность приложения на устройстве не отражается. Тесты производительности следует выполнять на целевом устройстве.
- Моделирование ограничений объекта `StageVideo` не выполняется. Обычно для воспроизведения видео в приложениях AIR for TV вместо класса `Video` используется класс `StageVideo`. Класс `StageVideo` использует преимущества производительности оборудования устройства, но имеет ограничения при воспроизведении. На настольном компьютере ADL воспроизводит видео без этих ограничений. Поэтому тесты воспроизведения видео следует выполнять на целевом устройстве.
- Исходный код собственного расширения не моделируется. Однако можно указать профиль `extendedTV`, поддерживающий собственные расширения, с помощью аргумента `ADL -profile`. ADL разрешает тестировать версию с фрагментом или моделью только расширения `ActionScript`, включенную в пакет `ANE`. Однако обычно соответствующее расширение, установленное на устройстве, также включает исходный код. Чтобы выполнить тестирование расширения с исходным кодом, запустите приложение на целевом устройстве.

Дополнительные сведения см. в разделе «[AIR Debug Launcher \(ADL\)](#)» на странице 170».

Использование собственных расширений

Если в приложении используются собственные расширения, команда ADL имеет следующий вид:

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

Например, предположим следующее:

- Путь к инструменту ADL указан в определении пути оболочки командной строки. (См. раздел «[Переменные среды Path](#)» на странице 327».)
- Файлы приложения находятся в текущем каталоге. В число этих файлов входят SWF-файлы и файл дескриптора приложения, который в данном примере имеет имя myApp-app.xml.
- Параметр `-extdir` задает имя каталога, содержащего каталог для всех собственных расширений, используемых в приложении. В каждом из этих каталогов содержится *распакованный* ANE-файл собственного расширения. Например:

```
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
```

Эти распакованные ANE-файлы содержат только версию с фрагментом или моделью для расширения ActionScript. На устройстве AIR for TV установлена версия расширения, содержащая исходный код.

Дополнительные сведения см. в разделе [Разработка собственных расширений для Adobe AIR](#).

Управление вводом

ADL моделирует нажатие кнопок на пульте управления ТВ-устройства. Отправлять события нажатия этих кнопок в смоделированную среду устройства можно с помощью меню, которое отображается при запуске ADL с профилем ТВ-устройства.

Размер экрана

Выполните тестирование приложения на экранах разных размеров, изменяя значение параметра ADL-`screenize`. Можно указать строку, содержащую четыре значения, которые представляют ширину и высоту стандартного и развернутого окна.

Всегда указывайте для портретной ориентации размеры в пикселях, задавая для ширины меньшее значение, чем для высоты. Например:

```
adl -screenize 728x1024:768x1024 myApp-app.xml
```


Трассировочные инструкции

При запуске приложения для ТВ-устройства на настольном компьютере данные трассировки выводятся в окне отладчика или терминала, из которого был запущен инструмент ADL.

Удаленная отладка с помощью Flash Professional

Flash Professional можно использовать для выполнения удаленной отладки приложения AIR for TV, запущенного на целевом устройстве. Процедура настройки удаленной отладки зависит от конкретного устройства. Например, MAX 2010, комплект инструментов разработчика Adobe® AIR® для ТВ-устройств, содержит документацию с подробным описанием процедуры для данного устройства.

Следующие операции по подготовке к удаленной отладке требуется выполнять на любом целевом устройстве.

- 1 В диалоговом окне «Параметры публикации» на вкладке «Flash» выберите «Разрешить отладку».

При установке данного параметра Flash Professional включает отладочную информацию во все SWF-файлы, создаваемые на основе файла FLA.

- 2 На вкладке «Подпись» в диалоговом окне «Параметры AIR» («Настройки приложения и установщика») установите параметры для подготовки файла AIR Intermediate (AIRI).

Когда разработка приложения еще не завершена, достаточно использовать AIRI-файл, не требующий цифровой подписи.

- 3 Опубликуйте приложение, создав файл AIRI.

Последним шагом является установка и запуск приложения на целевом устройстве. Однако эти шаги зависят от конкретного устройства.

Удаленная отладка с помощью Flash Builder

Flash Builder можно использовать для выполнения удаленной отладки приложения AIR for TV, запущенного на целевом устройстве. Процедура удаленной отладки зависит от конкретного устройства.

Следующие операции по подготовке к удаленной отладке требуется выполнять на любом целевом устройстве.

- 1 Выберите «Проект» > «Экспорт сборки выпуска». Установите параметры для подготовки файла AIR Intermediate (AIRI).

Когда разработка приложения еще не завершена, достаточно использовать AIRI-файл, не требующий цифровой подписи.

- 2 Опубликуйте приложение, создав файл AIRI.

- 3 Измените пакет приложения AIRI, чтобы входящие в него SWF-файлы содержали отладочную информацию.

SWF-файлы, содержащие отладочную информацию, расположены в каталоге проектов приложения Flash Builder с именем bin-debug. Замените файлы SWF в пакете AIRI SWF-файлами из каталога bin-debug.

На рабочем компьютере с Windows эту замену можно выполнить следующим образом.

- 1 Переименуйте файл пакета AIRI, установив для него расширение .zip вместо .airi.
- 2 Извлеките содержимое ZIP-файла.
- 3 Замените SWF-файлы в извлеченной структуре каталогов файлами из каталога bin-debug.
- 4 Повторно упакуйте файлы в извлеченном каталоге.
- 5 Измените имя ZIP-файла, вернув для него расширение .airi.

На рабочем компьютере Mac процедура замены зависит от конкретного устройства. Однако в общем случае требуется выполнить следующие действия.

- 1 Установите пакет AIRI на целевое устройство.
- 2 Замените SWF-файлы в установочном каталоге приложения на целевом устройстве SWF-файлами из каталога bin-debug.

Например, рассмотрите устройство, включенное в MAX 2010, комплект инструментов разработчика Adobe AIR for TV. Установите пакет AIRI, как описано в документации. Затем с помощью утилиты telnet в режиме командной строки подключитесь с рабочего компьютера Mac к целевому устройству. Замените SWF-файлы в каталоге установки приложения /opt/adobe/stagecraft/apps/<имя приложения>/ на SWF-файлы из каталога bin-debug.

Ниже представлена процедура удаленной отладки с помощью Flash Builder на устройстве, включенном в MAX 2010, комплект инструментов разработчика Adobe AIR for TV.

- 1 На рабочем компьютере с Flash Builder запустите AIR for TV Device Connector, включенный в комплект инструментов разработчика MAX 2010. Инструмент отобразит IP-адрес рабочего компьютера.
- 2 На устройстве из комплекта разработчика запустите приложение DevMaster, также включенное в комплект.
- 3 В приложении DevMaster введите IP-адрес рабочего компьютера, полученный с помощью AIR for TV Device Connector.
- 4 В приложении DevMaster убедитесь, что включен режим удаленной отладки.
- 5 Закройте приложение DevMaster.
- 6 На рабочем компьютере выберите «Запустить в AIR for TV Connector»
- 7 Запустите другое приложение на устройстве из комплекта. Убедитесь, что информация о трассировке отображается в AIR for TV Connector.

Если операция о трассировке не отображается, рабочий компьютер не подключен к устройству. Убедитесь, что порт рабочего компьютера, используемый для передачи информации о трассировке, доступен. В AIR for TV Device Connector можно выбрать другой порт. Также убедитесь, что в брандмауэре разрешен доступ к выбранному порту.

Затем запустите отладчик в среде Flash Builder. Выполните следующие действия.

- 1 В среде Flash Builder выберите «Выполнить» > «Конфигурации отладки».
- 2 Скопируйте имя проекта из существующей конфигурации отладки, предназначенной для локальной отладки.
- 3 В диалоговом окне «Конфигурации отладки» выберите «Веб-приложение». Затем нажмите на значок «Новая конфигурация запуска».
- 4 Вставьте имя проекта в поле «Проект».
- 5 В разделе «URL или путь для запуска» снимите флажок возле поля «Использовать по умолчанию». В текстовом поле введите `about:blank`.
- 6 Нажмите «Применить», чтобы сохранить изменения.
- 7 Выберите «Отладка», чтобы запустить отладчик Flash Builder.
- 8 Запустите приложение на устройстве из комплекта.

Теперь с помощью отладчика Flash Builder можно, например, задавать точки прерывания и анализировать переменные.

Глава 9. Использование собственных расширений для Adobe AIR

Собственные расширения для Adobe AIR предоставляют API-интерфейсы ActionScript, с помощью которых можно получать доступ к функциям устройства, запрограммированным с использованием собственного кода. Иногда разработчики собственных расширений сотрудничают с производителями устройств, а иногда являются сторонними разработчиками.

Сведения о разработке собственных расширений см. в документе [«Разработка собственных расширений для Adobe AIR»](#).

Собственное расширение включает следующие компоненты:

- классы ActionScript;
- собственный код.

Однако, поскольку разработчик приложения AIR использует собственное расширение, вам необходимо работать только с классами ActionScript.

Собственные расширения используются в следующих случаях:

- Реализация собственного кода предоставляет доступ к характерным для платформы функциям. Эти характерные для платформы функции недоступны во встроенных классах ActionScript и не могут быть реализованы в классах ActionScript приложения. Реализация собственного кода позволяет обеспечить подобные функции, поскольку с ее помощью можно получать доступ к аппаратному и программному обеспечению устройства.
- Реализация собственного кода иногда может обладать более высоким быстродействием, чем реализация, в которой используется только ActionScript.
- Реализация собственного кода может предоставлять доступ ActionScript к существующему собственному коду.

Примеры собственных расширений доступны в Центре разработчика Adobe. Например, одно собственное расширение предоставляет доступ приложениям AIR к функции вибрации Android. См. документ [«Собственные расширения для Adobe AIR»](#).

Файлы собственных расширений AIR (ANE)

Разработчики собственных расширений упаковывают собственное расширение в ANE-файл. ANE-файл представляет собой архивный файл, содержащий необходимые библиотеки и ресурсы для собственного расширения.

Обратите внимание, что для некоторых устройств в ANE-файле содержится библиотека собственного кода, используемая собственным расширением. Но для других устройств библиотека собственного кода установлена на устройстве. В некоторых случаях для определенного устройства собственное расширение может вовсе не иметь собственного кода; он реализован только с помощью ActionScript.

Разработчик приложений AIR использует ANE-файл следующим образом.

- Включение ANE-файла в путь к библиотеке приложения (этот процесс аналогичен включению SWC-файла в путь к библиотеке). Благодаря этому приложение сможет обращаться к классам ActionScript расширения.

***Примечание.** При компиляции приложения убедитесь, что для файла ANE выбрано динамическое связывание. При использовании Flash Builder на панели Path Properties (Параметры пути) средства ActionScript Builder укажите External. Если используется командная строка, укажите external-library-path.*

- Создание файла пакета ANE с приложением AIR.

Сравнение собственных расширений с классами ActionScript NativeProcess

ActionScript 3.0 предоставляет класс NativeProcess. Класс позволяет приложению AIR выполнять собственные процессы в операционной системе хоста. Эта возможность аналогична собственным расширениям, которые предоставляют доступ к характерным для платформы функциям и библиотекам. При принятии решения об использовании класса NativeProcess или собственного расширения учитывайте следующее.

- Только профиль AIR extendedDesktop поддерживает класс NativeProcess. Поэтому для приложений с профилями AIR mobileDevice и extendedMobileDevice возможен выбор только собственных расширений.
- Разработчики собственных расширений часто предоставляют собственные реализации для различных платформ, но предоставляемый ими API-интерфейс ActionScript, как правило, одинаков на всех платформах. При использовании класса NativeProcess код ActionScript, предназначенный для запуска собственного процесса, может различаться на разных платформах.
- Класс NativeProcess запускает отдельный процесс, в то время как собственное расширение выполняется в том же процессе, что и приложение AIR. Поэтому, если существуют опасения по поводу сбоя кода, безопаснее использовать класс NativeProcess. Однако отдельный процесс в некоторых случаях подразумевает необходимость реализации обработки межпроцессного взаимодействия.

Сравнение собственных расширений с библиотеками классов ActionScript (SWC-файлы)

SWC-файл является библиотекой классов ActionScript в архивном формате. SWC-файл содержит SWF-файл и другие файлы ресурсов. С помощью SWC-файла можно удобно получать общий доступ к классам ActionScript, не используя специальный код ActionScript и файлы ресурсов.

Пакет собственного расширения является ANE-файлом. Как и SWC-файл, ANE-файл также является библиотекой классов ActionScript, содержащей SWF-файл и другие файлы ресурсов в архивном формате. Однако наиболее важное различие ANE- и SWC-файлов заключается в том, что только ANE-файл может содержать библиотеку собственного кода.

***Примечание.** При компиляции приложения убедитесь, что для файла ANE выбрано динамическое связывание. При использовании Flash Builder на панели Path Properties (Параметры пути) средства ActionScript Builder укажите External. Если используется командная строка, укажите external-library-path.*

Дополнительные разделы справки

[Сведения о SWC-файлах](#)

Поддерживаемые устройства

Начиная с версии AIR 3, можно использовать собственные расширения в приложениях для следующих устройств:

- устройства Android с Android 2.2 и более поздних версий;
- устройства iOS с ОС iOS 4.0 и более поздних версий;
- iOS Simulator, начиная с версии AIR 3.3
- BlackBerry PlayBook
- настольные устройства Windows, поддерживающие AIR 3.0;
- настольные устройства Mac OS X, поддерживающие AIR 3.0;

Часто одно и то же собственное расширение предназначено для нескольких платформ. В ANE-файле расширения содержатся ActionScript и собственные библиотеки для каждой поддерживаемой платформы. Как правило, библиотеки ActionScript имеют одинаковые общедоступные интерфейсы для всех платформ. Собственные библиотеки обязательно различаются.

Иногда собственное расширение поддерживает платформу по умолчанию. Реализация для платформы по умолчанию содержит только код ActionScript, но не содержит собственного кода. Если пакет приложения создается для платформы, которая не поддерживается расширением, при выполнении в приложении используется реализация по умолчанию. Например, рассмотрим расширение, предоставляющее доступ к функции, которая применяется только на мобильных устройствах. Расширение может также обеспечивать реализацию по умолчанию, используемую в приложениях для настольных систем для имитации этой функции.

Поддерживаемые профили устройств

Следующие профили AIR поддерживают собственные расширения:

- `extendedDesktop`, начиная с AIR 3.0;
- `mobileDevice`, начиная с AIR 3.0;
- `extendedMobileDevice`, начиная с AIR 3.0.

Дополнительные разделы справки

[Поддержка профилей AIR](#)

Список задач для использования собственного расширения

Для использования собственного расширения в приложении выполните следующие задачи.

- 1 Объявите расширение в файле дескриптора приложения.
- 2 Включите ANE-файл в путь к библиотеке приложения.
- 3 Упакуйте приложение.

Объявление расширения в файле дескриптора приложения

Для всех приложений AIR предусмотрен файл дескриптора приложения. Когда в приложении используется собственное расширение, файл дескриптора приложения включает элемент `<extensions>`. Например:

```
<extensions>
  <extensionID>com.example.Extension1</extensionID>
  <extensionID>com.example.Extension2</extensionID>
</extensions>
```

Значение элемента `extensionID` совпадает со значением элемента `id` в файле дескриптора расширения. Файл дескриптора расширения — это файл XML с именем `extension.xml`. Он упаковывается в ANE-файл. Для просмотра файла `extension.xml` можно использовать инструмент извлечения архива.

Включение ANE-файла в путь к библиотеке приложения

Чтобы скомпилировать приложение, в котором используется собственное расширение, включите ANE-файл в путь к библиотеке.

Использование ANE-файла с Flash Builder

Если в приложении используется собственное расширение, включите в путь к библиотеке ANE-файл для собственного расширения. Затем с помощью Flash Builder можно выполнить компиляцию кода ActionScript.

Выполните следующие действия, используя Flash Builder 4.5.1.

- 1 Измените расширения ANE-файла с `.ane` на `.swc`. Это действие необходимо для того, чтобы среда Flash Builder могла найти файл.
- 2 В проекте Flash Builder выберите «Проект» > «Свойства».
- 3 В диалоговом окне «Свойства» выберите путь сборки Flex.
- 4 На вкладке «Путь к библиотеке» выберите «Добавить SWC-файл».
- 5 Выберите SWC-файл и нажмите «Открыть».
- 6 В диалоговом окне «Добавить SWC-файл» нажмите «ОК».

Теперь ANE-файл отображается на вкладке «Путь к библиотеке» в диалоговом окне «Свойства».

- 7 Разверните запись SWC-файла. Дважды нажмите на тип ссылки, чтобы открыть диалоговое окно «Параметры элемента пути библиотеки».
- 8 В диалоговом окне «Параметры элемента пути библиотеки» для параметра «Тип ссылки» установите значение «Внешняя».

Теперь можно скомпилировать приложение, например «Проект» > «Сборка проекта».

Использование ANE-файла с Flash Professional

Если в приложении используется собственное расширение, включите в путь к библиотеке ANE-файл для собственного расширения. Затем с помощью Flash Professional CS5.5 можно выполнить компиляцию кода ActionScript. Выполните следующие действия.

- 1 Измените расширения ANE-файла с .ane на .swc. Это действие необходимо для того, чтобы среда Flash Professional могла найти файл.
- 2 Выберите «Файл» > «Параметры ActionScript» для FLA-файла.
- 3 Перейдите на вкладку «Путь к библиотеке» в диалоговом окне «Дополнительные параметры ActionScript 3.0».
- 4 Нажмите кнопку «Перейти к SWC-файлу».
- 5 Выберите SWC-файл и нажмите «Открыть».

Теперь SWC-файл отображается на вкладке «Путь к библиотеке» в диалоговом окне «Дополнительные параметры ActionScript 3.0».

- 6 После выбора SWC-файла нажмите кнопку «Выберите параметры связывания для библиотеки».
- 7 В диалоговом окне «Параметры элемента пути библиотеки» для параметра «Тип ссылки» установите значение «Внешняя».

Создание пакета приложения, в котором используются собственные расширения

Используйте инструмент ADT для упаковки приложения, в котором используются собственные расширения. Создание пакета приложения с помощью Flash Professional CS5.5 или Flash Builder 4.5.1 невозможно.

Сведения об использовании инструмента ADT приводятся в документе [«AIR Developer Tool \(ADT\)»](#).

Например, следующая команда ADT создает DMG-файл (собственный файл установщика для ОС Mac OS X) для приложения, в котором используются собственные расширения:

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

Следующая команда создает пакет APK для устройства Android:

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

Следующая команда создает пакет iOS для приложения 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
```

Обратите внимание на следующие моменты.

- Используйте тип пакета собственного установщика.
- Укажите каталог расширения.
- Убедитесь, что ANE-файл поддерживает целевое устройство приложения.

Использование типа пакета собственного установщика

Пакет приложения должен быть собственным установщиком. Нельзя создавать кроссплатформенный пакет AIR (пакет .air) для приложения, в котором используется собственное расширение, поскольку собственные расширения, как правило, содержат собственный код. Однако обычно собственное расширение поддерживает несколько платформ с одинаковыми API-интерфейсами ActionScript. В таких случаях можно использовать один и тот же ANE-файл в различных пакетах собственных установщиков.

В следующей таблице приводится значение, которое необходимо использовать для параметра `-target` команды ADT.

Целевая платформа приложения	-target
Настольные устройства Mac OS X или Windows	-target native -target bundle
Android	-target apk или другие целевые устройства пакета Android.
iOS	-target ipa-ad-hoc или другие целевые устройства пакета iOS
iOS Simulator	-target ipa-test-interpretor-simulator -target ipa-debug-interpretor-simulator

Указание каталога расширения

Используйте параметр ADT `-extdir` для указания в ADT каталога, в котором содержатся собственные расширения (ANE-файлы).

Сведения об этом параметре см. в разделе «[Параметры файлов и путей](#)» на странице 194».

Убедитесь, что ANE-файл поддерживает целевое устройство приложения

Во время предоставления ANE-файла разработчик собственного расширения сообщает о том, какие платформы поддерживает расширение. Можно также воспользоваться инструментом извлечения архива для просмотра содержимого ANE-файла. Извлеченные файлы содержат каталог для каждой поддерживаемой платформы.

Знание поддерживаемых расширением платформ играет важную роль при создании пакета приложения, использующего ANE-файл. Примите во внимание следующие правила:

- Для создания пакета приложения Android ANE-файл должен включать платформу `Android-ARM`. Или ANE-файл должен включать платформу по умолчанию и хотя бы еще одну другую платформу.
- Для создания пакета приложения iOS ANE-файл должен включать платформу `iPhone-ARM`. Или ANE-файл должен включать платформу по умолчанию и хотя бы еще одну другую платформу.
- Для создания пакета приложения iOS Simulator ANE-файл должен содержать платформу `iPhone-x86`.
- Для создания пакета приложения Mac OS X ANE-файл должен включать платформу `macOS-x86`. Или ANE-файл должен включать платформу по умолчанию и хотя бы еще одну другую платформу.
- Для создания пакета приложения Windows ANE-файл должен включать платформу `Windows-x86`. Или ANE-файл должен включать платформу по умолчанию и хотя бы еще одну другую платформу.

Глава 10. Компиляторы ActionScript

Перед включением в приложение AIR код ActionScript и MXML требуется скомпилировать. Если используется интегрированная среда разработки (ИСР), например Adobe Flash Builder или Adobe Flash Professional, она выполняет компиляцию в фоновом режиме. Если ИСР не используется или используется сценарий сборки, компиляторы ActionScript для создания SWF-файлов можно вызвать из командной строки.

Сведения об инструментах командной строки AIR в пакете Flex SDK

Используемые для создания приложения Adobe AIR инструменты командной строки вызывают соответствующие инструменты, используемые для создания приложений:

- `amxmlc` вызывает `mxmlc` для компиляции классов приложения
- `acomps` вызывает `comps` для компиляции классов библиотек и компонентов
- `aasdoc` вызывает `asdoc` для создания файлов документации из комментариев в исходном коде

Единственное отличие между версиями служебных программ Flex и AIR заключается в том, что версии AIR загружают параметры конфигурации из файла `air-config.xml`, а не `flex-config.xml`.

Полное описание инструментов библиотеки Flex SDK и их параметров командной строки см. в [документации Flex](#). Здесь содержится лишь краткое описание инструментов Flex SDK, необходимое для понимания разницы между созданием приложений Flex и приложений AIR.

Дополнительные разделы справки

«Создание первого настольного приложения AIR с использованием пакета Flex SDK» на странице 39

Настройка компилятора

Обычно параметры компиляции вводятся в командной строке и одном из файлов конфигурации. В глобальном файле конфигурации Flex SDK обычно хранятся значения по умолчанию, используемые каждый раз при запуске компиляторов. Чтобы настроить Flex SDK под собственную среду разработки, внесите изменения в этот файл. Существует два глобальных файла конфигурации Flex, расположенные в папке `frameworks` в каталоге установки Flex SDK. `air-config.xml` — используется при запуске компилятора `amxmlc`. Этот файл настраивает компилятор для AIR путем включения библиотек AIR. `flex-config.xml` — используется при запуске компилятора `mxmlc`.

Значения по умолчанию можно оставить на время ознакомления с Flex и AIR, однако при работе над полномасштабным проектом рекомендуется подробно изучить различные варианты их настройки. В локальном файле конфигурации можно указать значения параметров компилятора для данного проекта, который имеет приоритет над глобальными значениями для данного проекта.

Примечание. Для приложений AIR не существует специальных параметров компиляции, однако при компиляции приложения AIR необходимы библиотеки AIR. Обычно ссылки на эти библиотеки размещаются в файле конфигурации проекта, в файле инструмента для создания, например Ant, или непосредственно в командной строке.

Компиляция исходных файлов MXML и ActionScript для AIR

Для компиляции файлов Adobe® ActionScript® 3.0 и MXML для приложения AIR можно использовать компилятор командной строки MXML (amxmlc). (Компиляция для HTML-приложений не требуется. Для компиляции SWF-файла во Flash Professional просто опубликуйте видеоролик в SWF-файл.)

Базовый шаблон командной строки для использования инструмента amxmlc имеет следующий вид:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

где *[compiler options]* задает параметры командной строки, используемые для компиляции приложения AIR.

Команда amxmlc вызывает стандартный компилятор Flex mxmlc с дополнительным параметром `+configname=air`. Этот параметр сообщает компилятору, что нужно использовать файл `air-config.xml`, а не `flex-config.xml`. В остальном amxmlc идентичен mxmlc.

Компилятор загружает файл конфигурации `air-config.xml`, в котором задаются библиотеки AIR и Flex, в большинстве случаев необходимые для компиляции приложения AIR. Также для переопределения или дополнения глобальных параметров конфигурации можно воспользоваться локальным файлом конфигурации на уровне проекта. Как правило, проще всего создать файл локальной конфигурации, скопировав и отредактировав его глобальную версию. Локальный файл можно загрузить с помощью `-load-config`:

`-load-config=project-config.xml` переопределяет глобальные параметры.

`-load-config+=project-config.xml` добавляет значения к тем глобальным параметрам, у которых их более одного, таким как `-library-path`. Глобальные параметры, принимающие только одно значение, переопределяются.

Если для файла локальной конфигурации используется специальное соглашение об именах, компилятор amxmlc автоматически загружает локальный файл. Например, если основной MXML-файл — `RunningMan.mxml`, то файл локальной конфигурации будет называться `RunningMan-config.xml`. Теперь для компиляции приложения нужно ввести всего лишь следующее:

```
amxmlc RunningMan.mxml
```

`RunningMan-config.xml` загружается автоматически, потому что его имя соответствует имени скомпилированного MXML-файла.

Примеры amxmlc

Ниже приводятся примеры использования компилятора amxmlc. (В приложении компилируются только ресурсы ActionScript и MXML).

Компиляция MXML-файла AIR:

```
amxmlc myApp.mxml
```

Компиляция и задание имени вывода:

```
amxmlc -output anApp.swf -- myApp.mxml
```

Компиляция MXML-файла ActionScript:

```
amxmlc myApp.as
```

Указание файла конфигурации компилятора:

```
amxmlc -load-config config.xml -- myApp.mxml
```

Добавление параметров из другого файла конфигурации:

```
amxmlc -load-config+=moreConfig.xml -- myApp.mxml
```

Добавление библиотек в командную строку (помимо библиотек, уже содержащихся в файле конфигурации):

```
amxmlc -library-path+=/libs/libOne.swc,/libs/libTwo.swc -- myApp.mxml
```

Компиляция MXML-файла AIR без файла конфигурации (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
```

Компиляция MXML-файла AIR без файла конфигурации (Mac OS X или 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
```

Компиляция MXML-файла AIR для использования библиотеки в общем доступе с средой выполнения:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --  
myApp.mxml
```

Чтобы использовать ANE, скомпилируйте файл MXML AIR (для ANE необходимо использовать
-external-library-path):

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

Компиляция из Java (путь к классу содержит mxmlc.jar):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional  
compiler options] -- myApp.mxml
```

Параметр flexlib определяет расположение каталога frameworks комплекта Flex SDK, благодаря чему
компилятор находит файл flex_config.xml.

Компиляция из Java (без задания пути к классу):

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air  
[additional compiler options] -- myApp.mxml
```

Порядок вызова компилятора, используя Apache Ant (пример использует задачу Java для запуска 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>
```

Компиляция компонента AIR или библиотеки кодов (Flex)

Используйте компилятор `asompc` для компиляции библиотек AIR и независимых компонентов. Компилятор компонента `asompc` во многом похож на `amxmlc`, но есть и ряд отличий:

- Необходимо указать, какие классы в коде необходимо включить в библиотеку или компонент.
- Команда `asompc` не ищет файл локальной конфигурации автоматически. Для использования файла конфигурации проекта необходимо задать параметр `-load-config`.

Команда `asompc` вызывает стандартный для Flex компилятор компонента `compc`, но загружает его параметры конфигурации из файла `air-config.xml` (не из `flex-config.xml`).

Файл конфигурации компилятора компонента

Используйте файл локальной конфигурации, чтобы не вводить вручную (и, возможно, с ошибками) в командную строку исходный путь и имена классов. Добавьте параметр `-load-config` к командной строке `asompc`, чтобы загрузить файл локальной конфигурации.

В примере ниже показана конфигурация для создания библиотеки с помощью двух классов: `ParticleManager` и `Particle`, оба включены в пакет `com.adobe.samples.particles`. Файлы классов расположены в папке `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>
```

Для компиляции библиотеки с помощью файла конфигурации `ParticleLib-config.xml` введите следующее:

```
asompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Для выполнения того же действия целиком из командной строки введите следующее:

```
asompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Введите всю команду в одну строку или воспользуйтесь символом продолжения строки).

Примеры асотрс

Предполагается, что используется файл конфигурации `myLib-config.xml`.

Компиляция компонента или библиотеки AIR:

```
асотрс -load-config myLib-config.xml -output lib/myLib.swc
```

Компиляция библиотеки в совместном доступе с средой выполнения:

```
асотрс -load-config myLib-config.xml -directory -output lib
```

(Перед тем как вызывать команду, убедитесь, что папка `lib` существует и пуста).

Глава 11. AIR Debug Launcher (ADL)

Средство AIR Debug Launcher (ADL) рекомендуется использовать для запуска SWF- и HTML-приложений во время разработки. С помощью ADL можно запустить приложение без предварительной упаковки и установки. По умолчанию, ADL использует среду выполнения, включенную в комплект SDK, т. е. для работы с ADL не потребуется отдельно устанавливать среду выполнения.

ADL печатает трассировочные инструкции и ошибки рабочей среды в виде стандартного вывода, но не поддерживает контрольные точки и иные функции отладки. Можно использовать программу отладки Flash Debugger (или интегрированные среды разработки, такие как Flash Builder) для решения сложных проблем с отладкой.

Примечание. Если инструкции `trace()` не отображаются на консоли, убедитесь, что в файле `mt.cfg` не задано `ErrorReportingEnable` или `TraceOutputFileEnable`. Дополнительные сведения о том, где находится этот файл в зависимости от платформы, см. в документе [Редактирование файла mt.cfg](#).

AIR поддерживает непосредственную отладку, что исключает необходимость в использовании отладочной версии среды выполнения (как, например, в ситуации с проигрывателем Adobe® Flash® Player). Для отладки с использованием командной строки воспользуйтесь отладчиком Flash Debugger или AIR Debug Launcher (ADL).

Отладчик Flash Debugger содержится в папке Flex SDK. Собственные версии, например `fdb.exe` для ОС Windows, находятся во вложенной папке `bin`. Java-версия находится во вложенной папке `lib`. AIR Debug Launcher, `adl.exe`, находится в папке `bin` в каталоге установки Flex SDK. (Отдельной Java-версии нет).

Примечание. Запустить приложение AIR непосредственно с помощью `fdb` нельзя, так как `fdb` пытается осуществить его запуск с помощью проигрывателя Flash Player. Вместо этого позвольте приложению AIR подключиться к активному сеансу `fdb`.

Использование ADL

Для выполнения приложения с помощью ADL, используйте следующий шаблон:

```
adl application.xml
```

где `application.xml` — это файл дескриптора приложения.

Полный синтаксис для ADL имеет следующий вид:

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(элементы в квадратных скобках «[]» являются необязательными)

-runtime runtime-directory задает каталог, содержащий используемую среду выполнения. Если каталог среды выполнения не указан отдельно, используется каталог из того же комплекта SDK, что и для ADL. Если переместить утилиту ADL из папки ее SDK, необходимо будет задать каталог среды выполнения. В Windows и Linux необходимо задать каталог, содержащий каталог `adobe AIR`. В Mac OS X задайте каталог, содержащий `Adobe AIR.framework`.

-pubid publisher-id присваивает определенное значение в качестве ID издателя приложения AIR для данного запуска. Указание временного ID издателя позволяет протестировать функции приложения AIR (такие как связь по локальной сети), использующие ID издателя для уникального определения приложения. Начиная с версии AIR 1.5.3, можно также указать идентификатор издателя в файле дескриптора приложения (и не использовать этот параметр).

Примечание. Начиная с версии AIR 1.5.3, идентификатор издателя больше автоматически не вычисляется и не назначается приложению AIR. Идентификатор издателя можно указать при создании обновления существующего приложения AIR, однако для новых приложений идентификатор издателя не требуется и не нужно указывать его.

-nodebug отключает поддержку отладки. В этом случае процесс приложения не сможет подключиться к программе отладки Flash, а диалоги, связанные с необработанными исключениями, будут заблокированы. (Несмотря на это трассировочные инструкции по-прежнему печатаются в окне консоли.) Отключение отладки ускоряет работу приложения и более точно эмулирует режим работы установленного приложения.

-atlogin симулирует запуск приложения при входе в систему. Этот флаг позволяет проверять логику приложения, которая выполняется, только когда приложение запускается при входе пользователя в систему. Когда используется `-atlogin`, свойство `reason` объекта `InvokeEvent`, отправляемое приложению, будет иметь значение `login` вместо `standard` (если приложение еще не запущено).

-profile profileName — ADL выполняет отладку приложения с помощью указанного профиля. `profileName` может принимать одно из следующих значений:

- `desktop`
- `extendedDesktop`
- `mobileDevice`

Если в дескриптор приложения включен элемент `supportedProfiles`, профиль, заданный с помощью аргумента `-profile`, должен входить в список поддерживаемых элементов. Если аргумент `-profile` не используется, в качестве активного профиля используется первый профиль из дескриптора приложения. Если элемент `supportedProfiles` не включен в дескриптор приложения и аргумент `-profile` не задан, будет использоваться профиль `desktop`.

Дополнительные сведения см. в разделах «[supportedProfiles](#)» на странице 256 и «[Профили устройств](#)» на странице 263».

-screensize value — размер симулированного экрана, используемый при запуске приложений с профилем `mobileDevice` в настольных системах. Задайте размер экрана как предустановленный тип или укажите обычную ширину и высоту в пикселях для портретной ориентации, а также ширину и высоту для полноэкранного режима. Чтобы указать в виде типа, используйте один из следующих предопределенных типов экрана:

Тип экрана	Обычная ширина x высота	Полноэкранная ширина x высота
480	720 x 480	720 x 480
720	1280 x 720	1280 x 720

Тип экрана	Обычная ширина x высота	Полноэкранная ширина x высота
1080	1920 x 1080	1920 x 1080
Droid	480 x 816	480 x 854
FWQVGA	240x432	240x432
FWVGA	480 x 854	480 x 854
HVGA	320 x 480	320 x 480
iPad	768 x 1004	768 x 1024
iPadRetina	1536 x 2008	1536x2048
iPhone	320 x 460	320 x 480
iPhoneRetina	640 x 920	640 x 960
iPhone5Retina	640 x 1096	640 x 1136
iPhone6	750 x 1294	750 x 1334
iPhone6Plus	1242 x 2148	1242 x 2208
iPod	320 x 460	320 x 480
iPodRetina	640 x 920	640 x 960
iPod5Retina	640 x 1096	640 x 1136
NexusOne	480 x 762	480 x 800
QVGA	240x320	240x320
SamsungGalaxyS	480 x 762	480 x 800
SamsungGalaxyTab	600 x 986	600 x 1024
WQVGA	240x400	240x400
WVGA	480 x 800	480 x 800

Чтобы задать размер экрана в пикселях, используйте следующий формат:

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Всегда указывайте для портретной ориентации размеры в пикселях, задавая для ширины меньшее значение, чем для высоты. Например, размеры экрана для NexusOne можно задать следующим образом:

```
-screensize 480x762:480x800
```

-extdir extension-directory Каталог, в котором среда выполнения осуществляет поиск собственных расширений. В каталоге содержится подкаталог для каждого собственного расширения, используемого в приложении. В каждом из этих подкаталогов содержится *распакованный* ANE-файл расширения. Например:

```
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
```

При использовании параметра `-extdir` учитывайте следующее:

- Для команды ADL требуется, чтобы все эти указанные каталоги имели расширение `.ane`. Но часть имени файла, предшествующая суффиксу `.ane`, может быть любым допустимым именем файла. Она может *не* совпадать со значением элемента `extensionID` файла дескриптора приложения.
- Параметр `-extdir` можно указать несколько раз.
- Использование параметра `-extdir` для инструмента ADT и инструмента ADL отличается. В инструменте ADT параметр указывает каталог, в котором содержатся файлы ANE.
- Для указания каталогов расширения можно также использовать переменную среды `AIR_EXTENSION_PATH`. См. раздел «[Переменные среды ADT](#)» на странице 201».

`application.xml` — это файл дескриптора приложения. См. раздел «[Файлы дескриптора приложения AIR](#)» на странице 220». Дескриптор приложения — это единственный параметр, необходимый ADL и, в большинстве случаев, единственный требуемый параметр.

`root-directory` задает корневой каталог запускаемого приложения. Если он не задан, используется каталог, в котором находится файл дескриптора.

`--arguments` Любые строки символов, стоящие после `--`, передаются приложению в качестве аргументов командной строки.

Примечание. При запуске уже работающего приложения AIR новый экземпляр приложения не запускается. Вместо этого работающему экземпляру отправляется событие `invoke`.

Примеры ADL

Запуск приложения в текущем каталоге:

```
adl myApp-app.xml
```

Запуск приложения в подкаталоге текущего каталога:

```
adl source/myApp-app.xml release
```

Запуск приложения и передача двух аргументов командной строки: «tick» и «tock»:

```
adl myApp-app.xml -- tick tock
```

Запуск приложения в определенной среде выполнения:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Запуск приложения без поддержки отладки:

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

Запустите приложение с использованием профиля мобильного устройства и симулируйте размер экрана Nexus One:

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Запустите приложение с помощью Apache Ant (в примере приведены пути для 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>
```

Завершение работы ADL и коды ошибок

Ниже перечислены коды выхода, выводимые ADL.

Код выхода	Описание
0	Успешный запуск. ADL завершает работу после выхода из приложения AIR.
1	Успешный вызов уже работающего приложения AIR. ADL завершает работу немедленно.
2	Ошибка использования. ADL переданы неверные аргументы.
3	Не удается найти среду выполнения.
4	Не удается запустить среду выполнения. Это часто происходит из-за того, что версия, указанная в приложении, не совпадает с версией среды выполнения.
5	Произошла ошибка, причины которой неизвестны.
6	Не удается найти файл дескриптора приложения.
7	Недопустимое содержимое дескриптора приложения. Обычно эта ошибка говорит о неправильной сборке XML-файла.
8	Не удается найти основной файл содержимого приложения (заданный элементом <content> файла дескриптора).

Код выхода	Описание
9	Основной файл содержимого приложения не является допустимым SWF- или HTML-файлом.
10	Приложение не поддерживает профиль, указанный в параметре -profile.
11	В текущем профиле аргумент -screensize не поддерживается.

Глава 12. AIR Developer Tool (ADT)

AIR Developer Tool (ADT) — это многофункциональный инструмент с интерфейсом командной строки, предназначенный для разработки приложений AIR. ADT позволяет выполнять следующие задачи:

- Упаковка приложений AIR в файл установочный файл .air.
- Упаковка приложений AIR в формат файла исходного установщика, например в файл .exe для Windows, файл .ipa для iOS или файл .apk для Android.
- Упаковка собственного расширения в формат файла AIR Native Extension (ANE)
- Подписание приложений AIR с помощью цифрового сертификата.
- Изменение (перенос) цифровой подписи, используемой для обновлений приложения.
- Определение устройств, подключенных к компьютеру
- Создание самозаверяющего сертификата для подписания кода.
- Удаленная установка, запуск и удаление приложений с мобильных устройств.
- Удаленная установка и удаление среды выполнения AIR с мобильных устройств.

ADT — это программа Java, которая включена в [AIR SDK](#). Для использования этого инструмента требуется среда Java 1.5 или более поздней версии. В SDK включен файл сценария для вызова ADT. Для использования сценария необходимо установить путь к программе Java с помощью переменной среды path. Если в переменной среды path также указан путь к каталогу AIR SDK bin, вызвать ADT можно путем выполнения команды `adt` в командной строке с указанием соответствующих аргументов. (Инструкции по настройке переменной среды path можно найти в документации к операционной системе. Дополнительные сведения об определении путей для большинства компьютерных систем также описаны в разделе «[Переменные среды Path](#)» на странице 327.)

Для использования ADT требуется не менее 2 ГБ свободной памяти на компьютере. В противном случае при работе ADT могут возникать проблемы, связанные с недостатком памяти, особенно при упаковке приложений для iOS.

Если в переменной path определены пути к программе Java и каталогу bin AIR SDK, для запуска ADT можно использовать следующую команду:

```
adt -command options
```

Примечание. Большинство интегрированных сред разработки, включая *Adobe Flash Builder* и *Adobe Flash Professional*, могут создавать пакеты и подписывать приложения AIR. Если применяется подобная среда разработки, то использование ADT для выполнения этих общих задач как правило не требуется. Но, возможно, все равно потребуется использовать ADT в качестве инструмента командной строки для реализации функций, не поддерживаемых интегрированной средой разработки. Кроме того, ADT можно использовать в качестве инструмента командной строки для процессов автоматической сборки.

Команды ADT

В первом аргументе, который передается в ADT, указывается одна из следующих команд.

- `-package` — упаковывает приложения AIR или AIR Native Extension (ANE).

- `-prepare` — упаковывает приложения AIR в виде промежуточного файла (AIRI), при этом подписание не выполняется. Файлы AIRI не могут быть установлены.
- `-sign` — подписывает пакет AIRI, созданный с помощью команды `-prepare`. При использовании команд `-prepare` и `-sign` упаковка и подписание могут выполняться в разное время. Команду `-sign` можно использовать для подписания и переподписания пакета ANE.
- `-migrate` — применяет подпись переноса к подписанному пакету AIR, который позволяет использовать новый или обновленный сертификат для подписания кода.
- `-certificate` — создает самозаверяющий сертификат для подписания цифрового кода.
- `-checkstore` — проверяет наличие доступа к цифровому сертификату в хранилище ключей.
- `-installApp` — устанавливает приложение AIR на устройство или эмулятор устройств.
- `-launchApp` — запускает приложение AIR на устройстве или эмуляторе устройств.
- `-appVersion` — возвращает версию приложения AIR, установленного на устройстве или эмуляторе устройств.
- `-uninstallApp` — удаляет приложение AIR с устройства или эмулятора устройств.
- `-installRuntime` — устанавливает среду выполнения AIR на устройстве или эмуляторе устройств.
- `-runtimeVersion` — возвращает версию среды выполнения AIR, установленную на устройстве или эмуляторе устройств.
- `-uninstallRuntime` — удаляет среду выполнения AIR, установленную на устройстве или эмуляторе устройств.
- `-version` — возвращает номер версии ADT.
- `-devices` — передает информацию о подключенных мобильных устройствах или эмуляторах.
- `-help` — отображает список команд и параметров.

Многие команды ADT используют одинаковые наборы флагов и параметров. Подробное описание наборов параметров представлено ниже.

- [«Параметры подписания кода ADT»](#) на странице 192
- [«Параметры файлов и путей»](#) на странице 194
- [«Параметры подключения к отладчику»](#) на странице 195
- [«Параметры собственных расширений»](#) на странице 196

Команда ADT package

Команда `-package` всегда должна запускаться из основного каталога приложения. Команда имеет следующий синтаксис:

Создание пакета AIR на основе файлов компонентов приложения:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Создание собственного пакета платформы на основе файлов компонентов приложения:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Создание собственного пакета платформы, включающего собственное расширение, на основе файлов компонентов приложения:

```
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
```

Создание исходного пакета на основе файла AIR или AIRI:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Создайте пакет собственного расширения из файлов собственного расширения компонента:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Примечание. ANE-файл можно не назначать, поэтому параметры `AIR_SIGNING_OPTIONS` являются необязательными в этом примере.

AIR_SIGNING_OPTIONS Параметры подписи AIR задают сертификат, который используется для подписания файла установки AIR. Параметры подписи описаны в разделе «[Параметры подписания кода ADT](#)» на странице 192».

-migrate. Этот флаг указывает, что приложение подписано сертификатом переноса в дополнение к сертификату, указанному в параметрах `AIR_SIGNING_OPTIONS`. Этот флаг допустим, только если упаковывается приложение для настольных компьютеров в качестве собственного установщика и в приложении используется собственное расширение. В других случаях возникнет ошибка. Параметры подписи для сертификата переноса указываются в параметрах `MIGRATION_SIGNING_OPTIONS`. Эти параметры

подписи описаны в разделе «[«Параметры подписания кода ADT»](#) на странице 192». Флаг `-migrate` позволяет создавать обновления для приложений с собственным файлом установки настольной платформы, использующих собственное расширение платформы, а также изменять сертификат подписи кода приложения в тех случаях, например, когда срок оригинального сертификата истек. Дополнительные сведения см. в разделе «[«Подписание обновленной версии приложения AIR»](#) на странице 213».

Флаг `-migrate` команды `-package` доступен в версиях AIR 3.6 и более поздних.

-target Тип создаваемого пакета. Поддерживаются следующие типы пакетов:

- `air` — пакет AIR. «`air`» — это значение по умолчанию. При создании файлов AIR и AIRI указывать флаг `-target` не требуется.
- `airn` — исходный пакет приложения для устройств с расширенным профилем телевизионных устройств.
- `ane` — пакет собственных расширений AIR
- Целевые объекты пакета Android:
 - `apk` — пакет Android. Пакет такого типа может быть установлен только на устройство Android, не на эмулятор.
 - `apk-captive-runtime` — пакет Android, включающий приложение и встроенную версию среды выполнения AIR. Пакет такого типа может быть установлен только на устройство Android, не на эмулятор.
 - `apk-debug` — пакет Android с дополнительной информацией для отладки. (SWF-файлы приложения также должны быть скомпилированы с поддержкой отладки.)
 - `apk-emulator` — пакет Android, предназначенный для использования в эмуляторе без поддержки отладки. (Чтобы разрешить отладку одновременно на эмуляторе и устройстве, используйте флаг `apk-debug`.)
 - `apk-profile` — пакет Android, поддерживающий профили производительности и памяти приложения.
- Целевые объекты пакета iOS:
 - `ipa-ad-hoc` — пакет iOS для специального развертывания.
 - `ipa-app-store` — пакет iOS для распространения через Apple App Store.
 - `ipa-debug` — пакет iOS с дополнительной информацией для отладки. (SWF-файлы приложения также должны быть скомпилированы с поддержкой отладки.)
 - `ipa-test` — пакет iOS, скомпилированный без информации для оптимизации и отладки.
 - `ipa-debug-interpreter` — функциональные возможности, как у пакета отладки, но компиляция выполняется быстрее. Однако байт-код ActionScript интерпретируется и не преобразуется в машинный код. Поэтому в пакете интерпретатора код выполняется медленнее.
 - `ipa-debug-interpreter-simulator` — функциональность эквивалентна `ipa-debug-interpreter`, но пакет предназначен для приложения iOS Simulator. Только для компьютеров Macintosh. Если используется этот параметр, необходимо также добавить параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.
 - `ipa-test-interpreter` — функциональные возможности, как у пакета тестирования, но компиляция выполняется быстрее. Однако байт-код ActionScript интерпретируется и не преобразуется в машинный код. Поэтому в пакете интерпретатора код выполняется медленнее.

- `ipa-test-interpreter-simulator` — функциональность эквивалентна `ipa-test-interpreter`, но пакет предназначен для приложения iOS Simulator. Только для компьютеров Macintosh. Если используется этот параметр, необходимо также добавить параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.
- `native` — исходный установщик для настольных систем. Тип создаваемого файла соответствует исходному формату установщика операционной системы, в которой выполняется команда:
 - EXE — Windows
 - DMG — Mac
 - DEB — Ubuntu Linux (AIR 2.6 или более ранней версии)
 - RPM — Fedora или OpenSuse Linux (AIR 2.6 или более ранней версии)

Дополнительные сведения см. в разделе «[Упаковка собственного установщика для настольной системы](#)» на странице 59».

-sampler (только iOS, AIR 3.4 и более поздней версии) Включает сэмплер ActionScript, основанный на телеметрии, в приложениях для iOS. Использование этого флага позволяет произвести профилирование приложения с помощью инструмента Adobe Scout. Несмотря на то, что инструмент [Scout](#) позволяет выполнить профилирование любого содержимого платформы Flash, включение подробной телеметрии дает более полное представление о времени вызова функций ActionScript, списке DisplayList, визуализации Stage3D и т. д. Включение этого флага приводит к незначительному ухудшению производительности, поэтому не используйте его в окончательных сборках приложений.

-hideAneLibSymbols (только iOS, AIR 3.4 и более поздних версий) Разработчики приложений могут использовать несколько расширений в машинном коде из различных источников, и если файлы ANE содержат одинаковые имена символов, ADT сгенерирует ошибку «duplicate symbol in object file» (дублирование символов в объектном файле). В некоторых случаях эта ошибка может проявиться в виде непредвиденного завершения во время выполнения. Параметр `hideAneLibSymbols` позволяет указать, сделать ли символы библиотеки ANE видимыми только для исходных кодов только этой библиотеки (`yes`) или видимыми глобально (`no`):

- **yes** — скрывает символы ANE, за счет чего устраняются любые непреднамеренные конфликты имен.
- **no** — (по умолчанию) символы ANE не скрываются. Это поведение соответствует версиям AIR ниже 3.4

-embedBitcode (только для ОС iOS, AIR 25 или более поздних версий) Разработчики могут использовать параметр `embedBitcode` для указания того, требуется ли встраивать двоичный код в приложения для iOS («Да» или «Нет»). Если не указано иное, для этого параметра используется значение по умолчанию «Нет».

DEBUGGER_CONNECTION_OPTIONS Параметры подключения к отладчику указывают, должен ли пакет отладки подключаться к удаленному отладчику, запущенному на другом компьютере, или прослушивать соединение удаленного отладчика. Этот набор параметров поддерживают только мобильные пакеты отладки (с типом `ark-debug` и `ipa-debug`). Эти параметры описаны в разделе «[Параметры подключения к отладчику](#)» на странице 195».

-airDownloadURL — указывает альтернативный URL для загрузки и установки среды выполнения AIR на устройствах Android. Если параметр не указан и среда выполнения не установлена, приложение AIR будет перенаправлять пользователя к среде выполнения AIR на Android Маркет.

Если приложение распространяется через альтернативные рынки (отличные от Android Маркета, управляемого Google), возможно потребуются указать URL для загрузки среды выполнения AIR с этого рынка. Некоторые альтернативные рынки не разрешают приложениям запрашивать загрузку из внешних источников. Этот параметр поддерживается только пакетами Android.

NATIVE_SIGNING_OPTIONS Параметры исходной подписи указывают сертификат, который используется для подписания файла исходного пакета. Данный параметр предназначен для применения подписи, используемой исходной операционной системой, не средой выполнения AIR. В остальном параметры идентичны параметрам AIR_SIGNING_OPTIONS. Подробное описание этих параметров приведено в разделе «[Параметры подписания кода ADT](#)» на странице 192.

Исходные подписи поддерживаются в Windows и Android. В Windows необходимо задать параметры подписи AIR и параметры исходной подписи. В Android можно задать только параметры исходной подписи.

Во многих случаях для применения подписи AIR и исходной подписи можно использовать сертификат подписи кода. Однако это верно не для всех ситуаций. Например, в соответствии с политикой Google для приложений, размещаемых в Android Маркете, необходимо, чтобы все приложения были подписаны сертификатом, действительным по крайней мере до 2033 года. Это означает, что для подписания приложений Android не следует использовать сертификат, выданный известным сертифицирующим органом, который рекомендуется для применения подписи AIR (ни один из сертифицирующих органов не выдает сертификаты для подписания кода с таким сроком действия).

output — имя создаваемого файла пакета. Указание расширения файла не является обязательным. Если расширение не указано, добавляется расширение, определяемое значением параметра `-target` и текущей операционной системой.

app_descriptor — путь к файлу дескриптора приложения. Можно задать абсолютный путь или путь относительно текущей директории. (В файле AIR имя файла дескриптора приложения изменяется на `application.xml`.)

-platformsdk — путь к SDK платформы для целевого устройства:

- Android — в состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды AIR_ANDROID_SDK_HOME (если заданы оба значения, используется путь, указанный в командной строке).
- iOS — в комплект поставки набора средств SDK для AIR входит связанный набор SDK для iOS. Параметр `-platformsdk` позволяет создавать пакеты приложений с внешним набором средств SDK, чтобы не ограничиваться использованием только набора SDK для iOS. Например, если с помощью самой последней версии набора SDK для iOS создано расширение, можно указать этот набор SDK при формировании пакета приложения. Кроме того, если вместе с приложением iOS Simulator используется ADT, необходимо всегда добавлять параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.

-arch Разработчики приложений могут использовать этот аргумент, чтобы создавать APK для платформ x86, принимает следующие значения:

- `armv7` — ADT упаковывает APK для платформы Android armv7.
- `x86` — ADT упаковывает APK для платформы Android x86.

`armv7` используется по умолчанию, если значение не определено

FILE_OPTIONS — указывает, какие файлы приложения включаются в пакет. Подробное описание параметров файлов приведено в разделе «[Параметры файлов и путей](#)» на странице 194». Не указывайте параметры файлов при создании исходного пакета на основе файла AIR или AIRI.

input_airi — указывается при создании исходного пакета из файла AIRI. Параметр AIR_SIGNING_OPTIONS требуется указать, если для пакета задан тип `air` (или тип пакета не указан).

input_air Указывается при создании исходного пакета на основе файла AIR. Не указывайте параметр AIR_SIGNING_OPTIONS.

ANE_OPTIONS — указываются параметры и файлы для создания пакета собственных расширений. Полное описание параметров пакета расширений приводится в разделе «[Параметры собственных расширений](#)» на странице 196.

Примеры использования команды ADT -package

Упаковка файлов приложения в текущей папке для приложения AIR на основе SWF:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Упаковка файлов приложения в текущей папке для приложения AIR на основе HTML:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js image.gif
```

Упакуйте все файлы и подкаталоги в текущем рабочем каталоге:

```
adt -package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Примечание. *Файл-хранилище ключей содержит закрытый ключ, которым подписано приложение. Ни в коем случае не упаковывайте сертификат подписи в пакет AIR! Если в командах ADT используются знаки подстановки, поместите файл-хранилище ключей в другую папку, чтобы он не попал в пакет. В этом примере файл-хранилище ключей cert.p12 находится в родительском каталоге.*

Включайте в пакет только основные файлы и подкаталог images:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Включайте в пакет HTML-приложение и все файлы в подкаталогах HTML, scripts и images:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRAliases.js html scripts images
```

Включайте в пакет файл application.xml и основной SWF-файл в рабочем каталоге (release/bin):

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml -C release/bin myApp.swf
```

Включайте в пакет ресурсы более чем из одного места в файловой системе сборки. В этом примере ресурсы приложения до упаковки находятся в следующих папках:

```
/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
```

Выполнение команды ADT из каталога /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
```

создает следующую структуру пакета:

```
/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
```

Запустите ADT в качестве программы Java для простого приложения на основе SWF (без указания пути к классам):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Запустите ADT в качестве программы Java для простого приложения на основе HTML (без указания пути к классам):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Запустите ADT как программу Java (путь к классу Java должен включать пакет ADT.jar):

```
java -com.adobe.air.ADT -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml  
myApp.swf
```

Запускает ADT в виде задачи Java в Apache Ant (несмотря на то, что рекомендуемым вариантом является запуск команды ADT непосредственно из скрипта Ant). Пути в данном примере указаны для ОС 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>
```

Примечание. В некоторых системах двухбайтные символы, входящие в путь файла, могут быть интерпретированы неверно. Если это произойдет, попробуйте настроить для среды JRE, с помощью которой запускается ADT, использование набора символов UTF-8. Это выполняется по умолчанию в сценарии, который используется для запуска ADT на Mac и Linux. В файле `adt.bat` для Windows или при запуске ADT непосредственно из Java укажите параметр `-Dfile.encoding=UTF-8` в командной строке Java.

Команда ADT prepare

Команда `-rprepare` создает неподписанный пакет AIRI. Пакет AIRI не может использоваться самостоятельно. Используйте команду `-sign` для преобразования файла AIRI в подписанный пакет AIR или команду `package` для преобразования файла AIRI в исходный пакет.

Команда `-rprepare` имеет следующий синтаксис:

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output Имя создаваемого файла AIRI.

app_descriptor — путь к файлу дескриптора приложения. Можно задать абсолютный путь или путь относительно текущей директории. (В файле AIR имя файла дескриптора приложения изменяется на `application.xml`.)

FILE_OPTIONS — указывает, какие файлы приложения включаются в пакет. Подробное описание параметров файлов приведено в разделе «[Параметры файлов и путей](#)» на странице 194».

Команда ADT sign

Команда `-sign` подписывает файлы AIRI и ANE.

Команда `-sign` имеет следующий синтаксис:

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS — параметры подписи AIR указывают, какой сертификат используется для подписания файла пакета. Параметры подписи описаны в разделе «[«Параметры подписания кода ADT»](#)» на странице 192».

input — имя подписываемого файла AIRI или ANE.

output — имя создаваемого подписанного пакета.

Если файл ANE уже подписан, старая подпись удаляется. (Файлы AIR не могут быть переподписаны. Чтобы использовать новую подпись для обновления приложения, используйте команду `-migrate`.)

Команда ADT migrate

Команда `-migrate` применяет подпись переноса к файлу AIR. Подпись переноса следует использовать при обновлении или изменении цифрового сертификата, а также когда требуется обновить приложения, подписанные старым сертификатом.

Дополнительные сведения об упаковке приложений AIR с подписью переноса см. в разделе «[«Подписание обновленной версии приложения AIR»](#)» на странице 213».

***Примечание.** Сертификат переноса должен быть применен в течение 365 дней после окончания срока действия сертификата. Если льготный период истек, подпись переноса больше нельзя использовать для подписания обновлений приложения. Пользователи могут сначала выполнить обновление до версии приложения, которое было подписано с помощью подписи переноса, а затем установить последнее обновление. Также они могут удалить исходное приложение и установить новый пакет AIR.*

Для использования подписи переноса необходимо сначала подписать приложение AIR новым или обновленным сертификатом (с помощью команд `-package` или `-sign`), а затем применить подпись переноса, используя старый сертификат и команду `-migrate`.

Команда `-migrate` имеет следующий синтаксис:

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS — параметры подписи AIR указывают исходный сертификат, который использовался для подписания существующих версий приложения AIR. Параметры подписи описаны в разделе «[«Параметры подписания кода ADT»](#)» на странице 192».

input — файл AIR, уже подписанный НОВЫМ сертификатом приложения.

output — имя файла итогового пакета, содержащий подписи, примененные с использованием нового и старого сертификатов.

Имена входного и выходного файлов AIR должны отличаться.

***Примечание.** Команда ADT migrate не может использоваться в приложениях AIR для настольных платформ, включающих расширения на собственном языке платформы, поскольку такие приложения упаковываются в виде собственного файла установки платформы, а не файла air. Для смены сертификата приложения AIR для настольных платформ, включающего собственные расширения, следует упаковать такое приложение при помощи [«Команда ADT package»](#) на странице 177 с флагом `-migrate`.*

Команда ADT checkstore

С помощью команды `-checkstore` можно выполнить проверку хранилища ключей. Команда имеет следующий синтаксис:

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS — параметры подписания указывают, проверку какого хранилища ключей требуется выполнить. Параметры подписи описаны в разделе «[Параметры подписания кода ADT](#)» на странице 192».

Команда ADT certificate

С помощью команды `-certificate` можно создавать самозаверяющие сертификаты для подписания цифрового кода. Команда имеет следующий синтаксис:

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type  
output password
```

-cn — это строка, представляющая общее имя нового сертификата.

-ou — это строка, представляющая организационную единицу, выдавшую сертификат. (необязательно.)

-o — это строка, представляющая орган, выдавший сертификат. (необязательно.)

-c — двухбуквенный код страны в формате ISO-3166. Если код недопустимый, сертификат не генерируется. (необязательно.)

-validityPeriod — количество лет, в течение которых сертификат будет действовать. Если параметр не указан, будет установлен срок действия в течение пяти лет (необязательно.)

key_type Для сертификата следует использовать тип ключа `2048-RSA`.

output — путь и имя файла генерируемого сертификата.

password Пароль для доступа к новому сертификату. Пароль необходим для подписи файлов AIR этим сертификатом.

Команда ADT installApp

С помощью команды `-installApp` выполняется установка приложения на устройство или эмулятор.

Прежде чем переустанавливать приложение с помощью этой команды, существующее приложение необходимо удалить.

Команда имеет следующий синтаксис:

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package  
fileName
```

-platform Название платформы устройства. Укажите `ios` или `android`.

-platformsdk — путь к SDK платформы для целевого устройства (необязательный параметр):

- **Android** — в состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды `AIR_ANDROID_SDK_HOME` (если заданы оба значения, используется путь, указанный в командной строке).
- **iOS** — в комплект поставки набора средств SDK для AIR входит связанный набор SDK для iOS. Параметр `-platformsdk` позволяет создавать пакеты приложений с внешним набором средств SDK, чтобы не ограничиваться использованием только набора SDK для iOS. Например, если с помощью самой последней версии набора SDK для iOS создано расширение, можно указать этот набор SDK при формировании пакета приложения. Кроме того, если вместе с приложением iOS Simulator используется ADT, необходимо всегда добавлять параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.

-device Укажите *ios_simulator*, серийный номер (Android) или дескриптор (iOS) подключенного устройства. В ОС iOS этот параметр является обязательным; в ОС Android этот параметр необходимо указывать только в том случае, если к компьютеру подключены несколько работающих устройств или эмуляторов Android. Если указанное устройство не подключено, ADT возвращает код выхода 14: ошибка устройства (Android) или указано недопустимо устройство (iOS). Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

Примечание. Установка файла IPA непосредственно на устройство iOS доступна в AIR 3.4 и более поздних версий (требуется iTunes 10.5.0 или более поздней версии).

Используйте команду `adt -devices` (доступна в AIR 3.4 и более поздних версий) для определения дескриптора или серийного номера подключенных устройств. Обратите внимание, что в ОС iOS необходимо использовать дескриптор, а не UUID устройства. Дополнительные сведения см. в разделе «[Команда ADT devices](#)» на странице 191».

Кроме того, в Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:

```
adb devices
```

-package — имя файла устанавливаемого пакета. В ОС iOS это должен быть файл IPA. В Android это должен быть пакет APK. Если указанный пакет уже установлен, ADT возвращает код ошибки 14: «ошибка устройства».

Команда ADT `appVersion`

Команда `-appVersion` возвращает версию приложения, установленного на устройстве или в эмуляторе. Команда имеет следующий синтаксис:

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Название платформы устройства. Укажите *ios* или *android*.

-platformsdk — путь к SDK платформы для целевого устройства:

- Android — в состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды `AIR_ANDROID_SDK_HOME` (если заданы оба значения, используется путь, указанный в командной строке).
- iOS — в комплект поставки набора средств SDK для AIR входит связанный набор SDK для iOS. Параметр `-platformsdk` позволяет создавать пакеты приложений с внешним набором средств SDK, чтобы не ограничиваться использованием только набора SDK для iOS. Например, если с помощью самой последней версии набора SDK для iOS создано расширение, можно указать этот набор SDK при формировании пакета приложения. Кроме того, если вместе с приложением iOS Simulator используется ADT, необходимо всегда добавлять параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.

-device Укажите *ios_simulator* или серийный номер устройства. Устройство требуется указывать, только когда к компьютеру подключено несколько рабочих устройств или эмуляторов Android. Если указанное устройство не подключено, ADT возвращает код выхода 14: «ошибка устройства». Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

В Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:


```
adb devices
```

-appid — идентификатор установленного приложения AIR. Если приложение с указанным идентификатором не установлено на устройстве, ADT возвращает код выхода 14: «ошибка устройства».

Команда ADT launchApp

Команда `-launchApp` запускает установленное приложение на устройстве или эмуляторе. Команда имеет следующий синтаксис:

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Название платформы устройства. Укажите *ios* или *android*.

-platformsdk — путь к SDK платформы для целевого устройства:

- **Android** — в состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды `AIR_ANDROID_SDK_HOME` (если заданы оба значения, используется путь, указанный в командной строке).
- **iOS** — в комплект поставки набора средств SDK для AIR входит связанный набор SDK для iOS. Параметр `-platformsdk` позволяет создавать пакеты приложений с внешним набором средств SDK, чтобы не ограничиваться использованием только набора SDK для iOS. Например, если с помощью самой последней версии набора SDK для iOS создано расширение, можно указать этот набор SDK при формировании пакета приложения. Кроме того, если вместе с приложением iOS Simulator используется ADT, необходимо всегда добавлять параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.

-device Укажите *ios_simulator* или серийный номер устройства. Устройство требуется указывать, только когда к компьютеру подключено несколько рабочих устройств или эмуляторов Android. Если указанное устройство не подключено, ADT возвращает код выхода 14: «ошибка устройства». Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

В Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:

```
adb devices
```

-appid — идентификатор установленного приложения AIR. Если приложение с указанным идентификатором не установлено на устройстве, ADT возвращает код выхода 14: «ошибка устройства».

Команда ADT uninstallApp

Команда `-uninstallApp` полностью удаляет установленное приложение с устройства или эмулятора. Команда имеет следующий синтаксис:

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform Название платформы устройства. Укажите *ios* или *android*.

-platformsdk — путь к SDK платформы для целевого устройства:

- **Android** — в состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды AIR_ANDROID_SDK_HOME (если заданы оба значения, используется путь, указанный в командной строке).
- **iOS** — в комплект поставки набора средств SDK для AIR входит связанный набор SDK для iOS. Параметр `-platformsdk` позволяет создавать пакеты приложений с внешним набором средств SDK, чтобы не ограничиваться использованием только набора SDK для iOS. Например, если с помощью самой последней версии набора SDK для iOS создано расширение, можно указать этот набор SDK при формировании пакета приложения. Кроме того, если вместе с приложением iOS Simulator используется ADT, необходимо всегда добавлять параметр `-platformsdk`, указав путь к набору средств SDK приложения iOS Simulator.

-device Укажите `ios_simulator` или серийный номер устройства. Устройство требуется указывать, только когда к компьютеру подключено несколько рабочих устройств или эмуляторов Android. Если указанное устройство не подключено, ADT возвращает код выхода 14: «ошибка устройства». Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

В Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:

```
adb devices
```

-appid — идентификатор установленного приложения AIR. Если приложение с указанным идентификатором не установлено на устройстве, ADT возвращает код выхода 14: «ошибка устройства».

Команда ADT `installRuntime`

Команда `-installRuntime` устанавливает среду выполнения AIR на устройство.

Прежде чем выполнять переустановку с помощью этой команды, необходимо удалить существующую версию среды выполнения AIR.

Команда имеет следующий синтаксис:

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package fileName
```

-platform Название платформы устройства. В настоящее время эта команда поддерживается только на платформе Android. Используйте имя *android*.

-platformsdk — путь к SDK платформы для целевого устройства. В настоящее время SDK поддерживает только платформу Android. В состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды AIR_ANDROID_SDK_HOME (если заданы оба значения, используется путь, указанный в командной строке).

-device — серийный номер устройства. Устройство требуется указывать, только когда к компьютеру подключено несколько рабочих устройств или эмуляторов. Если указанное устройство не подключено, ADT возвращает код выхода 14: «ошибка устройства». Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

В Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:

```
adb devices
```

-package — имя файла среды выполнения, которую требуется установить. В Android это должен быть пакет APK. Если пакет не указан, среда выполнения для устройства или эмулятора выбирается из списка доступных сред в AIR SDK. Если среда выполнения уже установлена, ADT возвращает код ошибки 14: «ошибка устройства».

Команда ADT runtimeVersion

Команда `-runtimeVersion` возвращает версию среды выполнения AIR, установленную на устройстве или эмуляторе. Команда имеет следующий синтаксис:

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Название платформы устройства. В настоящее время эта команда поддерживается только на платформе Android. Используйте имя *android*.

-platformsdk — путь к SDK платформы для целевого устройства. В настоящее время SDK поддерживает только платформу Android. В состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды AIR_ANDROID_SDK_HOME (если заданы оба значения, используется путь, указанный в командной строке).

-device — серийный номер устройства. Устройство требуется указывать, только когда к компьютеру подключено несколько рабочих устройств или эмуляторов. Если среда выполнения не установлена или указанное устройство не подключено, ADT возвращает код выхода 14: «ошибка устройства». Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

В Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:

```
adb devices
```

Команда ADT uninstallRuntime

Команда `-uninstallRuntime` полностью удаляет среду выполнения AIR с устройства или эмулятора. Команда имеет следующий синтаксис:

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Название платформы устройства. В настоящее время эта команда поддерживается только на платформе Android. Используйте имя *android*.

-platformsdk — путь к SDK платформы для целевого устройства. В настоящее время SDK поддерживает только платформу Android. В состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Кроме того, путь к платформе SDK не требуется указывать в командной строке, если уже установлена переменная среды AIR_ANDROID_SDK_HOME (если заданы оба значения, используется путь, указанный в командной строке).

-device — серийный номер устройства. Устройство требуется указывать, только когда к компьютеру подключено несколько рабочих устройств или эмуляторов. Если указанное устройство не подключено, ADT возвращает код выхода 14: «ошибка устройства». Если подключено несколько устройств или эмуляторов, а устройство не указано, ADT возвращает код выхода 2: «ошибка использования».

В Android для получения списка серийных номеров подключенных устройств и запущенных эмуляторов используйте инструмент Android ADB:

```
adb devices
```

Команда ADT devices

Команда ADT `-help` выводит идентификаторы подключенных в данный момент мобильных устройств и эмуляторов:

```
adt -devices -platform ios|android
```

-platform — имя проверяемой платформы. Укажите `android` или `ios`.

Примечание. В ОС iOS данная команда требует iTunes 10.5.0 или более поздней версии.

Команда ADT help

Команда ADT `-help` выводит краткий список параметров командной строки:

```
adt -help
```

В данной справке используются следующие условные обозначения:

- `<>` — в угловых скобках указана информация, которую требуется предоставить.
- `()` — в круглых скобках указаны параметры в результатах выполнения команды `help`, которые считаются группой.
- `ALL_CAPS` — элементы, указанные заглавными буквами, представляют собой набор параметров, который описан отдельно.
- `|` — ИЛИ. Например, `(A | B)` означает элемент A или элемент B.
- `?` — 0 или 1. Знак вопроса после элемента указывает, что элемент является необязательным и может существовать только один экземпляр элемента.
- `*` — 0 или больше. Знак звездочки после элемента указывает, что элемент является необязательным, и может существовать любое количество экземпляров.
- `+` — 1 или больше. Знак плюса после элемента указывает, что элемент является обязательным, и может существовать несколько экземпляров.
- Без символа: если элемент не имеет суффикса, он является обязательным, и может существовать только один экземпляр.

Наборы параметров ADT

Некоторые команды ADT имеют общие наборы параметров.

Параметры подписания кода ADT

ADT использует криптографическую архитектуру Java (JCA) для доступа к закрытым ключам и сертификатам подписи приложений AIR. Параметры подписи позволяют определить хранилище ключей, а также закрытый ключ и сертификат внутри него.

В хранилище ключей должны быть закрытый ключ и связанная с ним цепочка сертификатов. Если сертификат подписи ведет к надежному сертификату на компьютере, содержимое поля общего имени сертификата выводится в качестве имени издателя в диалоговом окне установки AIR.

ADT требует, чтобы сертификат отвечал стандарту x509v3 (RFC3280) и включал расширение Extended Key Usage с соответствующими значениями для подписания кода. Ограничения, определенные в сертификате, соблюдаются, поэтому некоторые сертификаты могут не подойти для подписи приложений AIR.

***Примечание.** Когда возможно, ADT использует параметры прокси среды выполнения Java для подключения к интернет-ресурсам, что необходимо для проверки отозванных сертификатов и получения временных отметок. Если при использовании ADT у вас возникают проблемы с подключением к этим интернет-ресурсам, а сетевые параметры должны включать особые настройки прокси, возможно, следует изменить конфигурацию прокси рабочей среды Java.*

Синтаксис параметров подписи AIR

Для параметров подписи используется следующий синтаксис (в одной командной строке):

```
-alias aliasName  
-storetype type  
-keystore path  
-storepass password1  
-keypass password2  
-providerName className  
-tsa url
```

-alias — псевдоним ключа в хранилище. Псевдоним необязателен, если в хранилище всего один сертификат. Если псевдоним не задан, ADT просто использует первый ключ.

Не все программы управления хранилищами ключей позволяют давать сертификатам псевдонимы. Например, в системном хранилище ключей Windows в качестве псевдонима нужно использовать отличительное имя сертификата. Для вывода списка доступных сертификатов можно использовать утилиту Java Keytool, чтобы легче было определить псевдоним. Например, выполнение команды:

```
keytool -list -storetype Windows-MY
```

выводит следующую информацию о сертификатах:

```
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
```

Для ссылки на этот сертификат из командной строки ADT задайте следующий псевдоним:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

В Mac OS X псевдонимом сертификата в цепочке ключей (Keychain) является имя, отображаемое в приложении Keychain Access.

-storetype — тип хранилища, заданный его реализацией. Реализация хранилища по умолчанию, включаемая в среды Java, поддерживает типы JKS и PKCS12 types. Java 5.0 поддерживает и тип PKCS11 для доступа к аппаратным маркерам, и тип Keychain для доступа к цепочке ключей Mac OS X. Java 6.0 поддерживает тип MSCAPI (в Windows). Если установлены и настроены другие поставщики JCA, могут быть доступны и другие типы хранилищ ключей. Если тип хранилища ключей не задан, то по умолчанию используется тип поставщика JCA.

Тип хранилища	Формат хранилища	Минимальная версия Java
JKS	Файл Java keystore (.keystore)	1.2
PKCS12	Файл PKCS12 (.p12 или .pfx)	1.4
PKCS11	Аппаратный маркер	1.5
KeychainStore	Цепочка ключей Mac OS X	1.5
Windows-MY или Windows-ROOT	MSCAPI	1.6

-keystore — путь к файлу-хранилищу ключей, если хранилище является файлом.

-storepass — пароль для доступа к хранилищу ключей. Если не задан, ADT запросит пароль.

-keypass — пароль для доступа к закрытому ключу, которым подписано приложение AIR. Если не задан, ADT запросит пароль.

***Примечание.** Если в числе параметров команды ADT указан пароль, он сохраняется в истории командной строки. Поэтому не рекомендуется использовать параметры `-keypass` и `-storepass`, если крайне важно обеспечить безопасность сертификата. Также следует заметить, что если пароль не указан, во время ввода пароля по запросу символы не отображаются (в целях безопасности). Просто введите пароль и нажмите клавишу Enter.*

-providerName — поставщик JCA для определенного типа хранилища ключей. Если не задан, ADT использует поставщика по умолчанию для этого типа хранилища.

-tsa — задает URL-адрес сервера меток времени, соответствующего стандарту [RFC3161](#) для добавления метки времени к цифровой подписи. Если URL не задан, используется временная отметка Geotrust по умолчанию. Когда подписанное приложение AIR снабжено временной отметкой, приложение можно будет установить и после истечения срока действия сертификата, потому что отметка подтверждает, что на момент подписания сертификат был действителен.

Если ADT не может подключиться к серверу выдачи временных отметок, подпись не выполняется и пакет не формируется. Для отмены временных отметок укажите `-tsa none`. Однако приложение AIR, упакованное без временной отметки, нельзя будет установить после истечения срока действия сертификата.

***Примечание.** Большинство параметров подписи соответствуют параметрам утилиты Java Keytool. Утилиту Keytool можно использовать для изучения и управления хранилищами ключей в Windows. В Mac OS X для этой цели используется утилита безопасности Apple®.*

-provisioning-profile Файл поставки Apple iOS (требуется только для упаковки приложений iOS).

Примеры параметров подписи

Подпись с помощью файла .p12:

```
-storetype pkcs12 -keystore cert.p12
```

Подпись с помощью файла Java keystore по умолчанию:

```
-alias AIRcert -storetype jks
```

Подпись с помощью заданного файла Java keystore:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Подпись с помощью файла цепочки ключей Mac OS X:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Подпись с помощью файла системного хранилища ключей Windows:

```
-alias cn=AIRCert -storetype Windows-MY
```

Подпись с помощью аппаратного маркера (см. инструкции производителя маркера, чтобы настроить конфигурацию Java для использования маркера и получения верного значения `providerName`):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Подпись без включения временной отметки:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Параметры файлов и путей

Параметры файлов и путей позволяют указать файлы, которые требуется включить в пакет. Параметры файлов и путей имеют следующий синтаксис:

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs — файлы и каталоги, которые требуется упаковать в файл AIR. Можно задать любое количество файлов и каталогов, разделив их пробелом. Если указать каталог, то все файлы и подкаталоги в нем, кроме скрытых, будут добавлены в пакет. (Кроме того, если задан файл дескриптора приложения — напрямую, с помощью знака подстановки или подстановки каталогов, — он игнорируется и не добавляется в пакет повторно.) Заданные файлы и каталоги должны быть в текущем каталоге или одном из его подкаталогов. Используйте параметр `-C` для изменения текущего каталога.

Важная информация. Знаки подстановки нельзя использовать в аргументах `file_or_dir`, следующих за `-C`. (Перед передачей аргументов ADT командные оболочки подставляют значения вместо всех знаков подстановки, поэтому ADT будет искать файлы не в том месте.) Тем не менее для обозначения текущего каталога по-прежнему можно использовать знак точки «.». Например, `-C assets.` копирует все из каталога `assets`, включая подкаталоги, в корневой каталог пакета приложения.

`-C dir files_and_dirs` — изменяет рабочий каталог на значение `dir` перед обработкой последующих файлов и каталогов, добавленных в пакет приложения (определяется значением `files_and_dirs`). Файлы или каталоги добавляются в корневой каталог пакета приложения. Параметр `-C` можно использовать многократно для добавления файлов из разных точек файловой системы. Если для `dir` задан относительный путь, за основу всегда берется рабочий каталог.

В процессе обработки файлов и каталогов пакета средством ADT сохраняются относительные пути от текущего каталога к целевым файлам. При установке пакета эти пути подставляются в структуру каталогов приложения. Таким образом, параметр `-C release/bin lib/feature.swf` поместит файл `release/bin/lib/feature.swf` в подкаталог `lib` корневой папки приложения.

`-e file_or_dir dir` — помещает файл или каталог в указанный каталог пакета. Этот параметр нельзя использовать при упаковке ANE-файла.

Примечание. Элемент `<content>` файла дескриптора приложения должен указывать итоговое расположение основного файла приложения в дереве каталогов пакета приложения.

`-extdir dir` Значением `dir` является имя каталога, в котором должен выполняться поиск собственных расширений (ANE-файлов). Укажите абсолютный путь или путь относительно текущего каталога. Параметр `-extdir` можно указать несколько раз.

Указанный каталог содержит ANE-файлы для собственных расширений, используемых в приложении. Каждый ANE-файл в этом каталоге имеет расширение .ane. Однако имя файла перед расширением .ane не обязательно должно соответствовать значению элемента `extensionID` в файле дескриптора приложения.

Например, при использовании `-extdir ./extensions` каталог `extensions` может иметь следующий вид:

```
extensions/  
  extension1.ane  
  extension2.ane
```

Примечание. Использование параметра `-extdir` для инструмента ADT и инструмента ADL отличается. В инструменте ADL параметр указывает каталог с подкаталогами, каждый из которых содержит неупакованный ANE-файл. В инструменте ADT параметр указывает каталог, в котором содержатся ANE-файлы.

Параметры подключения к отладчику

Когда целью пакета является `ark-debug`, `ipa-debug` или `ipa-debug-interpreter`, параметры подключения можно использовать, чтобы указать, будет ли приложение пытаться подключиться к удаленному средству Debugger (как правило, используемому для отладки соединений WiFi) или прослушивать входящее подключение удаленного средства Debugger (как правило, используется для отладки соединений USB). Аргумент `-connect` используется для подключения к отладчику. Аргумент `-listen` используется для принятия подключения к отладчику через USB-соединение. Эти параметры являются взаимоисключающими (их нельзя использовать вместе).

Аргумент `-connect` имеет следующий синтаксис:

```
-connect hostString
```

-connect — указывает, пытается ли приложение подключиться к удаленному отладчику.

hostString — строка, определяющая компьютер, на котором запущен инструмент отладки Flash (FDB). Если аргумент не указан, приложение попытается подключиться к отладчику на компьютере, на котором пакет был создан. Хост можно указать в виде полностью определенного доменного имени:

machinename.subgroup.example.com или как IP-адрес: *192.168.4.122*. Если указанный хост (или хост по умолчанию) не удастся найти, среда выполнения выдаст запрос имени хоста.

Аргумент `-listen` имеет следующий синтаксис:

```
-listen port
```

-listen — если аргумент задан, среда выполнения ожидает подключения от удаленного отладчика.

port (необязательно) — порт для прослушивания. По умолчанию среда выполнения прослушивает порт 7936. Дополнительные сведения по использованию аргумента `-listen` см. в разделе «Удаленная отладка с помощью FDB через USB» на странице 112.

Параметры профилирования приложения Android

Если установлен тип целевого пакета `ark-profile`, можно задать параметры профилировщика, чтобы указать, какой SWF-файл предварительной загрузки следует использовать для профилирования производительности и памяти. Параметры профилировщика имеют следующий синтаксис:

```
-preloadSWFPath directory
```


-preloadSWFPath — если аргумент задан, приложение попытается найти SWF-файл предварительной загрузки в указанном каталоге. Если аргумент не задан, ADT включает SWF-файл предварительной загрузки из AIR SDK.

directory — каталог, содержащий SWF-файл предварительной загрузки профилировщика.

Параметры собственных расширений

Параметры собственных расширений задают параметры и файлы для упаковки ANE-файла для собственного расширения. Используйте эти параметры вместе с командой создания пакета ADT, в которой значением параметра `-target` является `ane`.

```
extension-descriptor -swc swcPath  
                    -platform platformName  
                    -platformoptions path/platform.xml  
                    FILE_OPTIONS
```

extension-descriptor — файл дескриптора для собственного расширения.

-swc — SWC-файл, содержащий код ActionScript и ресурсы для собственного расширения.

-platform — название платформы, которую поддерживает данный ANE-файл. Можно включить несколько параметров `-platform`, каждый из которых имеет собственное значение `FILE_OPTIONS`.

-platformoptions Путь к файлу с параметрами платформы (`platform.xml`). Этот файл предназначен для указания нестандартных параметров компоновщика, общих библиотек и статических библиотек сторонних разработчиков, используемых данным расширением. Дополнительные сведения и примеры приведены в разделе Встроенные библиотеки iOS.

FILE_OPTIONS Определяет, какие встроенные файлы платформы следует включить в пакет, например для включения определенных статических библиотек в пакет собственного расширения. Подробное описание параметров файлов приведено в разделе «[Параметры файлов и путей](#)» на странице 194». (Параметр `-e` нельзя использовать при упаковке ANE-файла.)

Дополнительные разделы справки

[Упаковка собственного расширения](#)

Сообщения об ошибках ADT

В следующих таблицах перечислены возможные ошибки, которые могут регистрироваться программой ADT, и их возможные причины:

Ошибки при проверке дескриптора приложения

Код ошибки	Описание	Примечания
100	Невозможно выполнить синтаксический анализ дескриптора приложения	Проверьте, нет ли в дескрипторе приложения синтаксических ошибок XML, например незакрытых тегов.
101	Отсутствует пространство имен	Добавьте недостающее пространство имен.
102	Недопустимое пространство имен	Проверьте написание пространства имен.

Код ошибки	Описание	Примечания
103	Непредвиденный элемент или атрибут	Удалите неподходящие элементы или атрибуты. Нестандартные значения в файле дескриптора не допустимы. Проверьте написание имен элементов или атрибутов. Убедитесь, что элементы размещены внутри соответствующих родительских элементов, а атрибуты используются с допустимыми элементами.
104	Отсутствует элемент или атрибут	Добавьте недостающий элемент или атрибут.
105	Элемент или атрибут содержит недопустимое значение	Исправьте неподходящее значение.
106	Недопустимая комбинация атрибутов окна	Некоторые параметры окна, например <code>transparency = true</code> и <code>systemChrome = standard</code> , не могут использоваться одновременно. Измените один из несовместимых параметров.
107	Минимальный размер окна больше, чем максимальный размер окна.	Измените или минимальное, или максимальное значение размера окна.
108	Атрибут уже используется в предыдущем элементе	
109	Повторяющийся элемент.	Удалите повторяющийся элемент.
110	Требуется указать по крайней мере один элемент заданного типа.	Добавьте отсутствующий элемент.
111	Ни один из профилей, указанных в дескрипторе приложения, не поддерживает исходные расширения.	Добавьте в список <code>supportedProfiles</code> профиль, поддерживающий собственные расширения.
112	Тип целевого пакета AIR не поддерживает исходные расширения.	Выберите целевое устройство, поддерживающее собственные расширения.
113	Аргументы <code><nativeLibrary></code> и <code><initializer></code> должны быть указаны одновременно.	Функция <code>initializer</code> должна быть указана для всех собственных библиотек в собственном расширении.
114	Найден аргумент <code><finalizer></code> без <code><nativeLibrary></code> .	Аргумент <code>finalizer</code> следует задать только в случае, если платформа использует исходную библиотеку.
115	Платформа по умолчанию не должна содержать исходной реализации.	Не указывайте исходную библиотеку для платформы по умолчанию.
116	Вызов браузера не поддерживается для этой цели.	Элемент <code><allowBrowserInvocation></code> не может иметь значение <code>true</code> для указанной цели создания пакета.
117	Для создания пакета собственных расширений эта цель должна иметь хотя бы пространство имен <code>n</code> .	Установите для пространства имен AIR в дескрипторе приложения поддерживаемое значение.

Сведения о пространствах имен, элементах, атрибутах и их допустимых значениях см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».

Ошибки значка приложения

Код ошибки	Описание	Примечания
200	Невозможно открыть файл значка	Проверьте существование файла по указанному пути. Попробуйте открыть файл другим приложением.
201	Неверный размер значка	Размер значка (в пикселях) должен соответствовать тегу XML. Для примера рассмотрим элемент дескриптора приложения: <pre><image32x32>icon.png</image32x32></pre> Размер изображения в значке <code>icon.png</code> должен быть ровно 32x32 пикселя.
202	Файл значка содержит неподдерживаемый формат изображения	Поддерживается только формат PNG. Преобразуйте изображения в другие форматы перед созданием пакета приложения.

Ошибки в файле приложения

Код ошибки	Описание	Примечания
300	Файл отсутствует, или его невозможно открыть	Невозможно найти или открыть файл, указанный в командной строке.
301	Файл дескриптора приложения отсутствует, или его невозможно открыть	Файл дескриптора приложения невозможно найти по указанному пути или его нельзя открыть.
302	В пакете отсутствует корневой файл содержимого	SWF- или HTML-файл, на который ссылается элемент <code><content></code> дескриптора приложения, должен быть добавлен к пакету путем включения в список файлов в командной строке ADT.
303	Файл значка отсутствует в пакете	Файлы значков, указанные в дескрипторе приложения, должны быть добавлены к пакету путем включения их в список файлов в командной строке ADT. Файлы значков не добавляются автоматически.
304	Недопустимое исходное содержимое окна	Файл, на который ссылается элемент <code><content></code> дескриптора приложения, не распознан как допустимый HTML- или SWF-файл.

Код ошибки	Описание	Примечания
305	SWF-версия содержимого исходного окна превышает версию пространства имен	SWF-версия файла, на который есть ссылка в элементе <code><content></code> дескриптора приложений, не поддерживается версией AIR, указанной в пространстве имен дескриптора. Например, при попытке создать пакет с файлом SWF10 (Adobe Flash Player 10) в качестве исходного содержимого приложения AIR 1.1 произойдет следующая ошибка.
306	Профиль не поддерживается.	Профиль, указываемый в файле дескриптора приложения, не поддерживается. См. раздел « supportedProfiles » на странице 256.
307	Пространство имен должно быть по крайней мере <i>млп</i> .	Используйте соответствующее пространство имен для функций в приложении (например, пространство имен 2.0).

Коды выхода для других ошибок

Код выхода	Описание	Примечания
2	Ошибка использования	Проверьте правильность аргументов командной строки
5	Неизвестная ошибка	Данная ошибка указывает на ситуацию, которая не может быть объяснена условиями общих ошибок. К числу возможных причин проблемы относятся несовместимость ADT и JRE, повреждения в установках ADT и JRE, а также ошибки программирования в ADT.
6	Запись в целевую папку невозможна	Убедитесь, что указанная (или подразумеваемая) целевая папка доступна, а на содержащем ее диске достаточно свободного места.
7	Доступ к сертификату невозможен	Убедитесь, что правильно указан путь к хранилищу ключей. Убедитесь, что возможен доступ к сертификату в хранилище ключей. Служебную программу Java 1.6 Keytool можно использовать для поиска и устранения проблем с доступом к сертификатам.
8	Недопустимый сертификат	Файл сертификата неправильно сформирован, изменен, просрочен или отозван.
9	Не удается подписать файл AIR	Проверьте параметры подписи, переданные ADT.

Код выхода	Описание	Примечания
10	Не удалось создать отметку времени	ADT не может установить соединение с сервером отметок времени. В случае подключения к Интернету через прокси-сервер, возможно, потребуется выполнить настройку параметров прокси для JRE.
11	Ошибка создания сертификата	Проверьте аргументы командной строки, использованной для создания подписей.
12	Недопустимый ввод	Проверьте пути к файлам и другие аргументы, переданные в ADT из командной строки.
13	Отсутствует SDK устройства	Проверьте конфигурацию SDK устройства. ADT не может найти SDK устройства, требуемую для выполнения указанной команды.
14	Ошибка устройства	ADT не может выполнить команду из-за ограничений устройства или проблемы с устройством. Данный код ошибки выдается, например, при попытке удаления приложения, которое не было установлено.
15	Нет устройств	Убедитесь, что устройство подключено и включено либо запущен эмулятор.
16	Отсутствуют компоненты GPL	Текущий SDK AIR не содержит всех компонентов, необходимых для выполнения запрошенной операции.
17	Сбой инструмента создания пакета устройства.	Создание пакета невозможно, поскольку необходимые компоненты операционной системы отсутствуют.

Ошибки Android

Код выхода	Описание	Примечания
400	Текущая версия SDK Android не поддерживает атрибут	Убедитесь, что аргумент указан корректно, и его значение соответствует элементу, для которого он задан. Если атрибут был добавлен после версии Android 2.2, в команде ADT может потребоваться указать флаг <code>-platformsdk</code> .
401	Текущая версия SDK Android не поддерживает значение атрибута	Убедитесь, что значение атрибута указано корректно и является действительным. Если значение атрибута было добавлено после версии Android 2.2, в команде ADT может потребоваться указать флаг <code>-platformsdk</code> .
402	Текущая версия SDK Android не поддерживает тег XML	Убедитесь, что имя тега XML указано корректно, и он является действительным для элемента манифеста документа Android. Если элемент был добавлен после версии Android 2.2, в команде ADT может потребоваться указать флаг <code>-platformsdk</code> .
403	Тег Android не может быть переопределен	Приложение пытается переопределить элемент манифеста Android, который зарезервирован для использования в AIR. См. раздел « Настройки Android » на странице 80».
404	Атрибут Android не может быть переопределен	Приложение пытается переопределить атрибут манифеста Android, который зарезервирован для использования в AIR. См. раздел « Настройки Android » на странице 80».
405	Тег android %1 должен быть первым элементом в теге manifestAdditions	Переместите указанный тег в необходимое расположение.
406	Атрибут %1 тега android %2 имеет недопустимое значение %3.	Укажите допустимое значение атрибута.

Переменные среды ADT

ADT считывает значения следующих переменных среды (если они установлены):

AIR_ANDROID_SDK_HOME — указывает путь к корневому каталогу SDK Android (каталог, содержащий папку инструментов). В состав пакета SDK AIR 2.6 и более поздних версий включены инструменты из пакета SDK для ОС Android, необходимые для реализации соответствующих команд ADT. Данный параметр следует задавать только для использования другой версии Android SDK. Если переменная определена, параметр `-platformsdk` не требуется указывать при выполнении команд ADT, требующих задания данного параметра. Если определена переменная и параметр в командной строке, будет использоваться путь, заданный в командной строке.

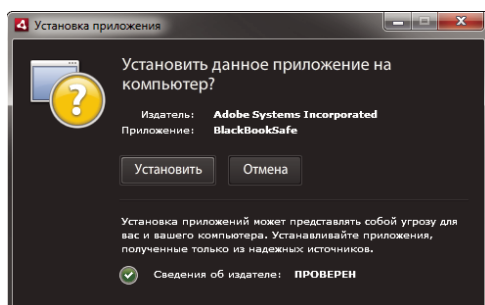
AIR_EXTENSION_PATH задает список каталогов, в которых должен осуществляться поиск собственных расширений, необходимых в приложении. Поиск указанных в списке каталогов осуществляется после поиска всех каталогов собственных расширений, указанных в командной строке ADT. В команде ADL также используется эта переменная среды.

***Примечание.** Некоторые системы могут некорректно интерпретировать двухбайтные символы в путях файловой системы, указанных с помощью этих переменных. Если это произойдет, попробуйте настроить для среды JRE, с помощью которой запускается ADT, использование набора символов UTF-8. Это выполняется по умолчанию в сценарии, который используется для запуска ADT на Mac и Linux. В файле `adt.bat` для Windows, а также при запуске ADT непосредственно из Java в командной строке Java укажите параметр - `Dfile.encoding=UTF-8`.*

Глава 13. Подписание приложений AIR

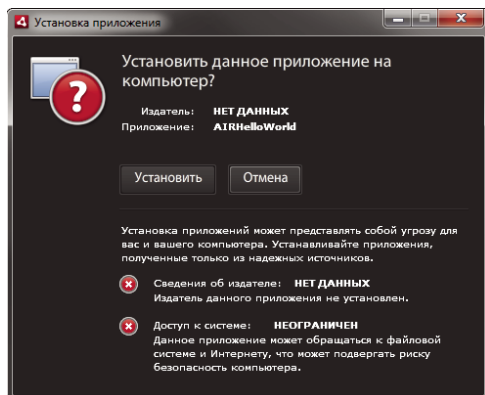
Цифровая подпись файлов AIR

Цифровая подпись установочных файлов AIR с помощью сертификатов, выданных сертифицирующими органами (СО), дает вашим пользователям гарантию того, что в устанавливаемый ими файл не был нечаянно или специально внедрен сторонний код, и обозначает вас в качестве автора подписи (издателя). Во время установки AIR отображает имя издателя, если приложение AIR заверено надежным сертификатом или сертификатом, *указывающим* на надежный сертификат на данном компьютере:



Диалоговое окно подтверждения для приложения, подписанного доверенным сертификатом

Если подписать приложение самоподписанным сертификатом (или если сертификат не связан с доверенным сертификатом), при установке приложения система безопасности пользователя будет подвергаться большей опасности. Дополнительная информация о риске отображается в следующем диалоговом окне:



Диалоговое окно подтверждения для приложения с самоподписанным сертификатом

Важная информация. Если злоумышленник каким-либо образом получит доступ к файлу с вашим ключом подписи или закрытым ключом, он сможет использовать ваши идентификационные данные в своих файлах AIR.

Сертификаты для подписывания кодов

Гарантии, ограничения безопасности и юридические обязательства, связанные с использованием сертификатов подписи кода, описаны в документе Certificate Practice Statements (CPS, «Положения об использовании сертификатов») и соглашениях с подписчиками, публикуемых сертифицирующими органами. Дополнительные сведения о соглашениях с сертифицирующими органами, которые выпускают сертификаты для подписания программных кодов AIR в данное время, см. по следующим адресам:

[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>)

[CPS Thawte](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/>)

[Соглашение с подписчиком VeriSign](https://www.verisign.com/repository/subscriber/SUBAGR.html) (<https://www.verisign.com/repository/subscriber/SUBAGR.html>)

О подписании кода AIR

При подписи файла AIR цифровая подпись включается в установочный файл. Подпись включает описание содержимого пакета, которая нужна, чтобы удостовериться, что файл AIR не изменялся с момента подписи, а также информацию о сертификатах подписи, которая нужна для идентификации издателя.

AIR использует инфраструктуру открытых ключей (PKI), поддерживаемую хранилищем ключей операционной системы, для проверки подлинности сертификата. Для проверки информации об издателе компьютер, на который устанавливается приложение AIR, должен либо подтвердить принятие сертификата, с помощью которого подписано приложение, либо подтвердить принятие цепочки сертификатов, ведущих к сертификату надежного СО.

Если файл AIR подписан сертификатом, не ведущим к одному из основных надежных сертификатов (как правило, это относится к самозаверяющим сертификатам), то информацию об издателе нельзя проверить. Хотя AIR может проверить, не был ли пакет AIR изменен с момента подписи, узнать, кто на самом деле создал и подписал файл, невозможно.

***Примечание.** Пользователь может принять самозаверяющий сертификат, после чего приложения AIR, подписанные с его помощью, будут выводить одно для всех имя издателя. В AIR сам пользователь не может отметить сертификат как надежный. Сертификат (без учета закрытого ключа) необходимо предоставлять пользователю отдельно, а он должен импортировать сертификат в нужное место в системном хранилище, пользуясь средствами, предлагаемыми операционной системой или соответствующим инструментом.*

Об идентификаторах издателя AIR

***Важная информация.** Начиная с версии AIR 1.5.3, идентификатор издателя больше не используется и не вычисляется на основе сертификата подписи кода. В новых приложениях не требуется и не должен использоваться идентификатор издателя. При обновлении существующих приложений необходимо указать исходный идентификатор издателя в файле дескриптора приложения.*

В версиях, предшествующих AIR 1.5.3, программа установки приложений AIR создавала идентификатор издателя во время установки файла AIR. Это был идентификатор, уникальный для сертификата, используемого для подписи файла AIR. Если один сертификат использовался для нескольких приложений AIR, им назначался одинаковый идентификатор издателя. При подписи обновления приложения с использованием другого сертификата, а иногда даже с использованием возобновленного экземпляра исходного сертификата, идентификатор издателя изменялся.

В AIR 1.5.3 и более поздних версиях идентификатор издателя не назначается средой AIR. В приложении, опубликованном с использованием AIR 1.5.3, строка идентификатора издателя может быть указана в дескрипторе приложения. Идентификатор издателя следует указывать только при публикации обновлений для приложений, которые изначально были опубликованы для версий AIR, предшествующих 1.5.3. Если исходный идентификатор не указывается в дескрипторе приложения, новый пакет AIR не считается обновлением существующего приложения.

Чтобы определить исходный идентификатор издателя, найдите файл `publisherid` в подкаталоге META-INF/AIR, в котором установлено исходное приложение. Строка в этом файле является идентификатором издателя. Чтобы указать идентификатор издателя вручную, в дескрипторе приложения должна быть указана среда выполнения AIR 1.5.3 (или более поздней версии) в объявлении пространства имен файла дескриптора приложения.

При наличии идентификатора издателя он используется в следующих целях:

- как часть ключа шифрования для зашифрованного локального хранилища;
- как часть пути к каталогу хранения приложения;
- как часть строки для локальных соединений;
- как часть строки учетных данных, используемой для вызова приложения в API-интерфейсе обозревателя AIR;
- как часть идентификатора OSID (используется при создании пользовательских программ установки/удаления).

При изменении идентификатора издателя также изменяется поведение функций AIR, зависящих от идентификатора. Например, данные в существующем зашифрованном локальном хранилище становятся недоступными, и в любых экземплярах Flash или AIR, создающих локальное соединение с приложением, в строке подключения должен использоваться новый идентификатор. В среде AIR 1.5.3 или более поздней версии изменение идентификатора издателя установленного приложения невозможно. При использовании другого идентификатора издателя и публикации пакета AIR в программе установки новый пакет считается другим приложением, а не обновлением.

О форматах сертификатов

Средства подписи AIR принимают любые хранилища ключей, доступные в рамках криптографической архитектуры Java (JCA). Сюда включены файлы-хранилища, например файлы формата PKCS12 (обычно имеют расширение `.pfx` или `.p12`), файлы Java `.keystore`, аппаратные хранилища PKCS11 и системные хранилища ключей. Форматы хранилищ ключей, к которым может иметь доступ средство ADT, зависят от версии и конфигурации среды выполнения Java, в которой запускается ADT. Доступ к некоторым типам хранилищ, например аппаратным маркерам PKCS11, возможен только при установке и конфигурации дополнительных программных драйверов и подключаемых модулей JCA.

Чтобы подписать файлы AIR, можно использовать большинство существующих сертификатов для подписания программного кода, а также можно получить новый ускоренно выпущенный сертификат для подписания приложений AIR. Например, можно использовать любой из следующих типов сертификатов, выпущенных VeriSign, Thawte, GlobalSign или ChosenSecurity:

- [ChosenSecurity](#)
 - Идентификатор TC Publisher для Adobe AIR
- [GlobalSign](#)
 - Сертификат для подписания кода ObjectSign
- [Thawte](#):
 - сертификат разработчика AIR
 - сертификат разработчика Apple
 - сертификат разработчика JavaSoft
 - сертификат Microsoft Authenticode
- [VeriSign](#):
 - Цифровое удостоверение Adobe AIR
 - цифровое удостоверение Microsoft Authenticode
 - цифровое удостоверение Sun Java

***Примечание.** Этот сертификат должен создаваться для подписания кода. Нельзя использовать SSL или сертификаты другого типа для подписания файлов AIR.*

Временные отметки

При подписи файла AIR средство упаковки отправляет серверу запрос временной отметки, чтобы получить независимо проверяемую дату и время подписи. Полученная временная отметка включается в файл AIR. Если сертификат подписи действителен на время подписания, приложение AIR может быть установлено, даже если на момент установки срок действия сертификата уже истек. С другой стороны, если временная отметка не получена, файл AIR не может быть установлен после истечения срока действия или отзыва сертификата.

Средства упаковки AIR получают временные отметки по умолчанию. Однако, чтобы приложение можно было упаковать, когда временную отметку получить невозможно, можно отключить эту функцию. Adobe рекомендует включать временные отметки во все файлы AIR, подлежащие открытому распространению.

По умолчанию средства упаковки AIR используют временные отметки Geotrust.

Получение сертификата

Для получения сертификата обычно нужно посетить сайт сертифицирующего органа и пройти процесс заверения. Средства, используемые для создания файла-хранилища ключей для средств AIR, зависят от типа приобретенного сертификата, способа хранения сертификата на принимающем компьютере, а в некоторых случаях и от обозревателя, через который получается сертификат. Например, чтобы получить и экспортировать сертификат разработчика Adobe от Thawte, необходимо использовать Mozilla Firefox. Затем этот сертификат можно экспортировать как файл P12 или PFX непосредственно из интерфейса пользователя Firefox.

Примечание. Java 1.5 и более поздние версии не позволяют использовать в паролях для защиты файлов сертификатов PKCS12 нестандартные символы ASCII. Язык Java используется средствами разработки AIR для создания подписанного пакета AIR. При экспорте сертификата в виде файла с расширением P12 или PFX, в пароле используются только стандартные символы ASCII.

С помощью средства ADT, которое используется для упаковки установочных файлов AIR, можно создать собственный самозаверяющий сертификат. Также можно использовать и инструменты сторонних разработчиков.

Инструкции по созданию самозаверяющих сертификатов и по подписи файлов AIR см. в разделе «[AIR Developer Tool \(ADT\)](#)» на странице 176. Файлы AIR также можно экспортировать и подписывать с помощью Flash Builder, Dreamweaver и обновления AIR для Flash.

В примере ниже показано, как получить от Thawte сертификат разработчика AIR и подготовить его для использования с ADT.

Пример: получение сертификата разработчика AIR от Thawte

Примечание. Рассматривается лишь один из возможных способов получения и подготовки сертификата подписания кода. У каждого выпускающего сертификаты органа есть собственные политики и процедуры.

Для приобретения сертификата разработчика AIR Thawte требуется использовать обозреватель Mozilla Firefox. Закрытый ключ сертификата сохраняется в хранилище ключей обозревателя. Убедитесь, что хранилище ключей Firefox защищено основным паролем, а компьютер защищен физически. (После завершения процесса заверения можно экспортировать и удалить сертификат из хранилища ключей обозревателя.)

В ходе процесса регистрации сертификата генерируется связка закрытого и открытого ключей. Закрытый ключ автоматически заносится в хранилище ключей Firefox. Для запроса и получения сертификата с сайта Thawte необходимо использовать один и тот же компьютер и один и тот же обозреватель.

- 1 Зайдите на сайт Thawte и перейдите к странице [Product page for Code Signing Certificates](#) («Сертификаты подписания кода»).
- 2 Выберите из списка сертификатов подписания кода сертификат разработчика AIR (Adobe AIR Developer Certificate).
- 3 Выполните все три шага процесса регистрации сертификата. Необходимо предоставить информацию об организации и контактные данные. Затем Thawte проверяет подлинность личности заявителя и запрашивает дополнительную информацию. После завершения проверки Thawte отправляет электронное письмо с инструкциями по получению сертификата.

Примечание. Дополнительные сведения о типе требуемой документации см. здесь: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.

- 4 Получите выписанный сертификат на сайте Thawte. Сертификат автоматически заносится в хранилище ключей Firefox.
- 5 Экпортируйте файл-хранилище с закрытым ключом и сертификат из хранилища ключей Firefox, как описано ниже.

Примечание. При экспорте закрытого ключа и сертификата из хранилища Firefox файл получает формат .p12 (pfx), который распознают ADT, Flex, Flash и Dreamweaver.

- a Откройте диалоговое окно *Менеджер сертификатов* в Firefox:
- b В ОС Windows откройте Сервис -> Параметры -> Дополнительно -> Шифрование -> Просмотр сертификатов

- c В Mac OS откройте Firefox -> Установки -> Дополнительно -> Шифрование -> Просмотр сертификатов
 - d В Linux откройте «Правка -> Установки -> Дополнительно -> Шифрование -> Просмотр сертификатов» (Edit -> Preferences -> Advanced -> Encryption -> View Certificates)
 - e Выберите сертификат подписания кода Adobe AIR из списка и нажмите кнопку **Резервная копия**.
 - f Введите имя файла и место его экспорта и нажмите кнопку **Сохранить**.
 - g Если используется основной пароль Firefox, вам потребуется ввести пароль устройства защиты ПО для экспорта файла. (Этот пароль используется только в Firefox.)
 - h В диалоговом окне *Выберите пароль резервного копирования* создайте пароль для файла-хранилища.
Важная информация. Пароль нужен для защиты файла хранилища и требуется, когда файл используется для подписи приложений AIR. Необходимо выбрать надежный пароль.
 - i Нажмите кнопку «ОК». Вы должны увидеть сообщение об успешном принятии пароля резервного копирования. Файл-хранилище с закрытым ключом и сертификатом сохраняется с расширением .p12 (в формате PKCS12)
- 6 Экспортированный файл можно использовать в ADT, Flash Builder, Flash Professional или Dreamweaver. Пароль, созданный для файла, потребуется при подписи приложения AIR.

Важная информация. *Закрытый ключ и сертификат по-прежнему сохраняются в хранилище ключей Firefox. Это не только позволяет экспортировать дополнительную копию файла с сертификатом, но и открывает еще один канал доступа, который также необходимо обезопасить, чтобы не ставить под угрозу общую безопасность сертификата и закрытого ключа.*

Замена сертификатов

В некоторых случаях необходимо изменить сертификат, используемый для подписи обновлений приложения AIR. Вот некоторые из них:

- Возобновление исходного сертификата подписи.
- обновление самозаверяющего сертификата до сертификата, выданного сертифицирующим органом;
- замена истекающего самозаверяющего сертификата на новый;
- замена одного коммерческого сертификата другим, например при изменении корпоративных данных

Чтобы файл AIR распознавался в среде AIR как обновление, необходимо подписать исходный и обновленный файлы AIR с использованием одного сертификата или применить подпись переноса сертификата для обновления. Подпись переноса — это вторая подпись, применяемая для обновления пакета AIR, использующего исходный сертификат. Подпись переноса использует исходный сертификат для определения того, что подписавшее лицо является издателем приложения.

После установки файла AIR с подписью переноса новый сертификат становится основным. Для последующих обновлений подпись переноса не требуется. Однако подписи переноса следует применять как можно дольше, чтобы обеспечить работу пользователей, пропустивших обновления.

Важная информация. *Чтобы обновить исходный сертификат, прежде чем закончится его срок действия, необходимо изменить сертификат и применить подпись переноса. В противном случае перед установкой новой версии пользователям потребуется удалить существующую версию приложения. Для AIR 1.5.3 и более поздней версии подпись переноса можно применить, используя устаревший сертификат в течение 365-дневного льготного периода после окончания его срока действия. Однако нельзя использовать просроченный сертификат для применения основной подписи приложения.*

Для замены сертификата выполните следующие действия:

- 1 Создайте обновление приложения
- 2 Упакуйте и подпишите файл обновления AIR **новым** сертификатом
- 3 Подпишите файл AIR снова, на этот раз **оригинальным** сертификатом (с помощью команды ADT - migrate)

В остальных отношениях файл AIR с подписью переноса является обычным файлом AIR. Если в системе, куда устанавливается приложение, нет оригинальной версии, AIR установит новую версию обычным образом.

***Примечание.** В версиях, предшествующих AIR 1.5.3, для подписи приложения AIR с использованием возобновленного сертификата не всегда требовалась подпись переноса. Начиная с версии AIR 1.5.3, подпись переноса всегда требуется для возобновленных сертификатов.*

Применение подписи переноса производится «Команда ADT migrate» на странице 185, как описано в разделе ««Подписание обновленной версии приложения AIR» на странице 213».

***Примечание.** Команда ADT migrate не может использоваться в приложениях AIR для настольных платформ, включающих расширения на собственном языке платформы, поскольку такие приложения упаковываются в виде собственного файла установки платформы, а не файла air. Для смены сертификата приложения AIR для настольных платформ, включающего собственные расширения, следует упаковать такое приложение при помощи «Команда ADT package» на странице 177 с флагом -migrate.*

Изменение учетных данных приложения

В версиях, предшествующих AIR 1.5.3, учетные данные приложения AIR изменялись при установке обновления с подписью переноса. Изменение учетных данных приложения имеет несколько последствий, в том числе:

- Новая версия приложения не может получать доступ к данным в существующем зашифрованном локальном хранилище.
- Изменяется адрес каталога хранилища приложения. Данные из старого каталога не копируются в новый каталог. (Но новое приложение может найти старый каталог с помощью ID издателя.)
- Приложение больше не может устанавливать локальные соединения с помощью ID издателя.
- Изменяется строка учетных данных, используемая для доступа к приложению с веб-страницы.
- Изменяется идентификатор OSID приложения. (Идентификатор OSID используется при написании пользовательских программ установки/удаления.)

При публикации обновления с использованием AIR 1.5.3 или более поздней версии изменение учетных данных приложения невозможно. Идентификаторы исходного приложения и издателя должны быть указаны в дескрипторе приложения файла AIR обновления. В противном случае новый пакет не распознается как обновление.

***Примечание.** При публикации нового приложения AIR с использованием AIR 1.5.3 или более поздней версии не следует указывать идентификатор издателя.*

Терминология

В этом разделе приводятся определения основных терминов, которые необходимо понимать, чтобы принимать верные решения о том, как подписывать свое приложение для открытого распространения.

Термин	Описание
Сертифицирующий орган (CO)	Сеть инфраструктуры открытых ключей, выступающая в роли независимой стороны, которая выдает сертификаты, тем самым подтверждая личность владельца ключа. Обычно CO выдают цифровые сертификаты, подписанные собственным закрытым ключом, — это означает, что личность держателя сертификата подтверждена.
Certificate Practice Statement (CPS, «Положение об использовании сертификатов»)	Содержит рекомендации и правила по выдаче и проверке сертификатов для сертифицирующих органов. CPS является частью договора между CO и его подписчиками и связанными лицами. Также в нем описывается политика проверки подлинности и степени гарантий различных сертификатов.
Список отозванных сертификатов (CRL)	Список выданных сертификатов, которые были впоследствии отозваны и более недействительны. AIR проверяет список CRL во время подписи приложения AIR, а если временной отметки нет, то еще раз при установке.
Цепочка сертификатов	Цепочка сертификатов — это последовательность, в которой каждый сертификат подписывается следующим.
Цифровой сертификат	Цифровой документ, содержащий информацию об идентификационных данных владельца, открытом ключе владельца и идентификационных данных самого сертификата. Сертификат, выданный CO, сам подписан сертификатом, принадлежащим этому CO.
Цифровая подпись	Зашифрованное сообщение или его свертка, которые могут быть расшифрованы только открытым ключом, образующим пару с закрытым. В PKI цифровая подпись содержит один или несколько цифровых сертификатов, которые выводят на выдавших их орган. Цифровая подпись может использоваться для проверки сообщения (или компьютерного файла) на предмет изменений с момента подписи (в пределах, разрешенных криптографическим алгоритмом) и, если исходить из того, что сертифицирующему органу доверяют, проверки подписавшего лица.
Хранилище ключей	База данных с цифровыми сертификатами и (в некоторых случаях) связанными с ними закрытыми ключами.
Криптографическая архитектура Java (JCA)	Открытая архитектура для управления и доступа к хранилищам ключей. Дополнительные сведения см. в руководстве по криптографической архитектуре Java .
PKCS #11	Стандарт интерфейса криптографических маркеров, разработанный Лабораторией RSA. Аппаратное хранилище ключей на основе маркеров.
PKCS #12	Стандарт синтаксиса обмена личной информацией, разработанный Лабораторией RSA. Файл-хранилище ключей, содержащий закрытый ключ и связанный с ним цифровой сертификат.
Закрытый ключ	Закрытая часть асимметричной криптографической системы, состоящей из закрытого и открытого ключей. Закрытый ключ необходимо держать в секрете, и его нельзя передавать по сети. Сообщения, подписанные цифровой подписью, шифруются автором с помощью закрытого ключа.
Открытый ключ	Открытая часть асимметричной криптографической системы, состоящей из закрытого и открытого ключей. Открытый ключ находится в общем доступе и используется для расшифровки сообщений, зашифрованных закрытым ключом.
Инфраструктура открытых ключей (PKI)	Система доверительных отношений, в которой CO подтверждают личности владельцев открытых ключей. Клиенты сети доверяют цифровым сертификатам, выданным CO, при проверке личности подписавшего цифровое сообщение или файл.
Временная отметка	Цифровая метка, содержащая дату и время того или иного события. ADT может ставить временные отметки согласно серверу времени пакета AIR, соответствующего стандарту RFC 3161 . Если функция временных отметок включена, AIR использует ее для проверки подлинности сертификата во время подписания. Это позволяет устанавливать приложение AIR после истечения срока действия сертификата.
Орган выдачи временных отметок	Орган, выдающий временные отметки. Чтобы AIR распознавала временные отметки, они должны соответствовать стандарту RFC 3161, а подпись временной отметки должна вести к надежному основному сертификату на принимающем компьютере.

Сертификаты iOS

Сертификат подписания кода, выданный Apple, используется для подписания приложений для ОС iOS, включая приложения, разработанные с помощью Adobe AIR. Применение подписи с использованием сертификата разработки Apple требуется для установки приложения на тестовые устройства. Применение подписи с использованием сертификата распространения требуется для распространения готового приложения.

Для подписания приложения ADT требуется доступ к сертификату подписания кода и связанному с ним личному ключу. Файл сертификата не содержит личный ключ. Необходимо создать хранилище ключей в формате файла обмена личной информацией (.p12 или .pfx), который будет содержать сертификат и личный ключ. См. раздел ««Преобразование сертификата разработчика в файл хранилища P12» на странице 212».

Генерация запроса на подпись сертификата

Чтобы получить сертификат разработчика, необходимо сгенерировать запрос на подписание сертификата и отправить его в Apple iOS Provisioning Portal.

При создании запроса на подписание сертификата генерируется связка закрытого и открытого ключей. Закрытый ключ сохраняется на компьютере. Отправьте в Apple запрос на подписание, содержащий открытый ключ и идентификационную информацию. Компания Apple возьмет на себя роль сертифицирующего органа. Apple подписывает ваш сертификат, используя собственный сертификат World Wide Developer Relations.

Формирование запроса на подпись сертификата в ОС Mac OS

В ОС Mac OS для формирования запроса подписи воспользуйтесь программой «Связка ключей». Программа «Связка ключей» расположена в каталоге Программы/Службные программы. Инструкции по генерации запроса на подписание сертификата доступны на портале Apple iOS Provisioning Portal.

Формирование запроса на подпись сертификата в ОС Windows

Для тех, кто работает в ОС Windows, будет проще получить сертификат разработчика iPhone, используя компьютер с ОС Mac. Тем не менее, получить сертификат на компьютере с ОС Windows возможно. Сначала нужно создать запрос на подпись сертификата (CSR-файл) с помощью OpenSSL.

- 1 Установите OpenSSL на компьютере Windows. (Перейдите на страницу <http://www.openssl.org/related/binaries.html>.)

Возможно, потребуется также установить файлы Visual C++ 2008 Redistributable, указанные на странице загрузки Open SSL. (Устанавливать Visual C++ на компьютер *не* требуется.)

- 2 Откройте сеанс командной строки Windows и компакт-диск для каталога корзины OpenSSL (например, c:\OpenSSL\bin\).
- 3 Создайте личный ключ, введя следующее в командную строку:

```
openssl genrsa -out mykey.key 2048
```

Сохраните файл с личным ключом. Он понадобится вам в дальнейшем.

При работе с OpenSSL внимательно читайте сообщения об ошибках. OpenSSL может формировать файлы даже при наличии ошибок, Однако эти файлы могут быть непригодны для использования. Если появляется сообщение об ошибке, проверьте правильность синтаксиса и выполните команду снова.

- 4 Создайте CSR-файл, введя следующее в командную строку:

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest -subj  
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

Подставьте свой адрес электронной почты, имя сертификата (CN) и страну (C).

- 5 Отправьте CSR-файл на [веб-сайт разработчиков iPhone](#). (См. раздел «Запрос сертификата разработчика iPhone и создание профиля обеспечения».)

Преобразование сертификата разработчика в файл хранилища P12

Чтобы создать хранилище ключей P12, необходимо объединить в одном файле сертификат разработчика Apple и связанный с ним закрытый ключ. Процедура создания файла хранилища определяется методом, который был использован для генерации запроса на подписание исходного сертификата, а также местом хранения закрытого ключа.

Преобразование сертификата разработчика iPhone в файл P12 в ОС Mac OS

Загрузив сертификат iPhone с сайта Apple, экспортируйте его в формате файла хранилища P12. В ОС Mac OS выполните следующие действия.

- 1 Откройте программу «Связка ключей» (каталог Программы/Служебные программы).
- 2 Если сертификат еще не добавлен в связку ключей, выберите «Файл» > «Импорт». Найдите файл сертификата (CER-файл), полученный от компании Apple.
- 3 В программе «Связка ключей» выберите категорию «Ключи».
- 4 Выберите личный ключ, связанный с данным сертификатом на разработку iPhone.
Личный ключ идентифицируется связанным с ним открытым сертификатом «Разработчик iPhone: <имя> <фамилия>».
- 5 Нажмите на сертификат iPhone Developer и выберите *Экспорт «iPhone Developer: Имя...»*.
- 6 Сохраните хранилище ключей в формате файла обмена личными данными (.p12).
- 7 Вам будет предложено создать пароль, который будет применяться при использовании хранилища для подписания приложений, а также при переносе ключа и сертификата из этого хранилища в другое хранилище.

Преобразование сертификата разработчика Apple в файл P12 в ОС Windows

Для разработки приложений AIR for iOS необходимо использовать файл сертификата P12. Этот сертификат создается на основе файла сертификата разработчика iPhone, полученного от компании Apple.

- 1 Преобразуйте файл сертификата разработчика, полученный от компании Apple, в файл сертификата PEM. С помощью командной строки запустите следующую операцию из каталога корзины (bin) OpenSSL.

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```
- 2 Если используется личный ключ из связки ключей на компьютере с ОС Mac, преобразуйте его в ключ PEM:

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```
- 3 Теперь можно создать действительный файл P12 на основе ключа и версии PEM сертификата разработчика iPhone:

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

Если используется ключ из связки ключей в ОС Mac OS, используйте версию PEM, созданную при выполнении предыдущего шага. В противном случае используйте ключ OpenSSL, созданный ранее (в ОС Windows).

Создание неподписанного промежуточного файла AIR с помощью ADT

С помощью команды `-prepare` можно создать неподписанный промежуточный файл AIR. Промежуточный файл AIR должен быть подписан командой ADT `-sign`, чтобы получился действительный файл установки AIR.

Команда `-prepare` принимает те же настройки и параметры, что и команда `-package` (кроме параметров подписи). Единственная разница в том, что получаемый файл не подписывается. Промежуточный файл имеет расширение `airi`.

Подписать промежуточный файл AIR можно с помощью команды ADT `-sign`. (См. раздел «[Команда ADT prepare](#)» на странице 184».)

Пример команды ADT `-prepare`

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Подписание промежуточного файла AIR с помощью ADT

Подписать промежуточный файл AIR можно с помощью команды ADT `-sign`. Команда `sign` может использоваться только с промежуточными файлами AIR (с расширением `airi`). Файл AIR нельзя подписать повторно.

Создать промежуточный файл AIR можно с помощью команды ADT `-prepare`. (См. раздел «[Команда ADT prepare](#)» на странице 184».)

Подпись файла AIRI

❖ Используйте команду ADT `-sign` со следующим синтаксисом:

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS. Параметры подписи определяют закрытый ключ и сертификат подписи файла AIR. Эти параметры описаны в разделе «[Параметры подписания кода ADT](#)» на странице 192».

airi_file — путь к неподписанному промежуточному файлу AIR, который требуется подписать.

air_file — имя создаваемого файла AIR.

Пример команды ADT `-sign`

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Дополнительные сведения см. в разделе «[Команда ADT sign](#)» на странице 184».

Подписание обновленной версии приложения AIR

Каждый раз при создании обновленной версии существующего приложения AIR следует подписывать обновленное приложение. При оптимальном раскладе для подписи обновленной версии используется тот же сертификат, что и для предыдущей версии. В этом случае процесс подписи в точности схож с первоначальной подписью приложения.

Если срок сертификата, использованного для подписи предыдущей версии приложения, истек, и он был заменен или обновлен, для подписи обновленной версии можно использовать обновленный или новый (полученный на замену) сертификат. Для этого следует подписать приложение новым сертификатом и применить подпись переноса к оригинальному сертификату. Подпись переноса подтверждает, что владелец исходного сертификата опубликовал обновление.

Прежде чем применить подпись переноса, обратите внимание на следующие моменты.

- Чтобы применить подпись переноса, исходный сертификат должен быть действителен или быть просроченным не более чем на 365 дней. Этот период считается льготным, и его продолжительность может быть изменена в будущем.

***Примечание.** До выхода AIR 2.6 льготный период составлял 180 дней.*

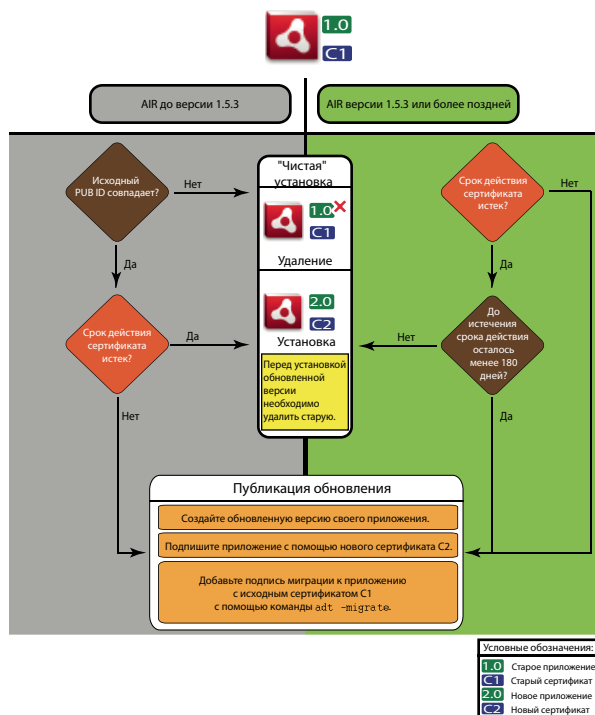
- После окончания 365-дневного льготного периода, который отсчитывается с момента истечения действия сертификата, подпись переноса не может быть применена. В этом случае пользователям потребуется перед установкой обновления удалить текущую версию приложения.
- 365-дневный льготный период применяется только к приложениям, для которых в пространстве имен дескриптора приложения указана версия AIR 1.5.3 или более поздняя версия.

Важная информация. Подписание обновлений с использованием подписи переноса просроченного сертификата является временным решением. Окончательное решение предполагает разработку стандартизированной процедуры подписания для управления развертыванием обновлений приложений. К примеру, каждое обновление подписывается последним имеющимся сертификатом, и применяется сертификат переноса — сертификат, использованный для подписи предыдущего обновления (если таковое было). Каждое обновление размещается на сервере с собственным URL, с помощью которого происходит загрузка приложения пользователями. Дополнительные сведения см. в разделе «[Процедура подписания для обновлений приложений](#)» на странице 279».

Следующая таблица и рисунок иллюстрируют процесс обращения с подписями переноса:

Сценарий	Состояние оригинального сертификата	Действие разработчика	Действие пользователя
Приложение на основе среды выполнения Adobe AIR 1.5.3 или более новой версии	Действительный	Опубликуйте последнюю версию приложения AIR	Действия не требуются Приложение обновляется автоматически
	Срок истек, но не более 365 дней назад	Подпишите приложение новым сертификатом. Примените подпись переноса с использованием устаревшего сертификата.	Действия не требуются Приложение обновляется автоматически
	Просрочен, льготный период закончился	Подпись переноса не может быть применена для обновления приложения AIR. Вместо этого опубликуйте другую версию приложения AIR, используя новый сертификат. Пользователи могут установить новую версию приложения AIR после удаления существующей версии.	Удалите текущую версию приложения AIR и установите последнюю версию

Сценарий	Состояние оригинального сертификата	Действие разработчика	Действие пользователя
<ul style="list-style-type: none"> Приложение на основе среды выполнения Adobe AIR 1.5.2 или более ранней версии Идентификатор издателя в дескрипторе обновленного приложения совпадает с идентификатором предыдущей версии 	Действительный	Опубликуйте последнюю версию приложения AIR	<p>Действия не требуются</p> <p>Приложение обновляется автоматически</p>
	Просрочен, льготный период закончился	<p>Подпись переноса не может быть применена для обновления приложения AIR.</p> <p>Вместо этого опубликуйте другую версию приложения AIR, используя новый сертификат. Пользователи могут установить новую версию приложения AIR после удаления существующей версии.</p>	Удалите текущую версию приложения AIR и установите последнюю версию
<ul style="list-style-type: none"> Приложение на основе среды выполнения Adobe AIR 1.5.2 или более ранней версии Идентификатор издателя в дескрипторе обновленного приложения не совпадает с идентификатором предыдущей версии 	Любой	Подпишите приложение AIR действующим сертификатом и опубликуйте его последнюю версию.	Удалите текущую версию приложения AIR и установите последнюю версию



Процедура подписания для обновлений

Перенос приложения AIR для использования нового сертификата

Процедура переноса приложения AIR на новый сертификат при обновлении приложения:

- 1 Создайте обновление приложения
 - 2 Упакуйте и подпишите файл обновления AIR **новым** сертификатом
 - 3 Подпишите файл AIR снова, на этот раз **оригинальным** сертификатом с помощью команды `ADT -migrate`
- Файл AIR, подписанный с помощью команды `-migrate`, может также использоваться для установки новой версии приложения — не только для обновления любой прошлой версии, подписанной устаревшим сертификатом.

Примечание. При обновлении приложения, опубликованного для версии AIR, предшествующей 1.5.3, в дескрипторе приложения необходимо указать исходный идентификатор издателя. В противном случае перед установкой обновления пользователям приложения потребуется удалить более раннюю версию.

Используйте команду ADT `-migrate` со следующим синтаксисом:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS.** Параметры подписи определяют закрытый ключ и сертификат подписи файла AIR. Эти параметры должны определять **оригинальный** сертификат подписи (их описание приводится в разделе «[Параметры подписания кода ADT](#)» на странице 192»).
- **air_file_in** — обновляемый файл AIR, подписанный **новым** сертификатом.
- **air_file_out** — создаваемый файл AIR.

Примечание. Имена входных и выходных файлов AIR должны отличаться.

Следующий пример иллюстрирует вызов функции ADT с флагом `-migrate` для применения подписи переноса к обновленной версии приложения AIR:

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Примечание. Команда `-migrate` была добавлена в ADT в AIR версии 1.1.

Перенос приложения AIR с собственным файлом установки на новый сертификат

Приложение AIR, опубликованное как собственный файл установки для конкретной платформы (например, приложение, использующее собственный API платформы), не может быть подписано командой ADT `-migrate`, поскольку не является файлом `air`. Для переноса приложения AIR, опубликованного в качестве собственного расширения платформы, на новый сертификат предусмотрена другая процедура:

- 1 Создайте обновление приложения.
- 2 Убедитесь, что в файле-дескрипторе приложения (`app.xml`) тег `<supportedProfiles>` содержит как профиль настольной системы (`desktop`), так и расширенный профиль настольной системы (`extendedDesktop`); либо совсем удалите тег `<supportedProfiles>` из дескриптора приложения.
- 3 Упакуйте и подпишите обновленное приложение как файл `air` при помощи команды ADT `-package` с новым сертификатом.
- 4 Примените сертификат переноса к файлу `.air`, используя команду ADT `-migrate` с оригинальным сертификатом (как описано ранее в разделе «[Перенос приложения AIR для использования нового сертификата](#)» на странице 216).
- 5 Упакуйте файл `.air` в виде собственного установочного файла платформы при помощи команды ADT `-package` с флагом `-target native`. Поскольку приложение уже подписано, на этом шаге нет необходимости указывать сертификат подписи.

В следующем примере показаны этапы 3–5 этой процедуры. Код вызывает программу ADT с командой `-package`, затем — с командой `-migrate`, затем снова с командой `-package` для упаковки обновленной версии приложения AIR в виде собственного файла установки данной платформы:

```
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
```

Перенос на новый сертификат приложения AIR, использующего собственное расширение платформы

Приложение AIR, использующее собственное расширение платформы, не может быть подписано при помощи команды ADT `-migrate`. Его перенос также не может быть выполнен по процедуре переноса приложения AIR с собственным файлом установки, поскольку оно не может быть опубликовано как промежуточный файл `.air`. Для переноса приложения AIR, использующего собственное расширение платформы, на новый сертификат предусмотрена другая процедура:

- 1 Создайте обновление приложения
- 2 Упакуйте и подпишите обновленный собственный файл установки при помощи команды ADT `-package`. Упакуйте приложение с указанием нового сертификата и включите в команду флаг `-migrate` с указанием оригинального сертификата.

Для вызова команды ADT `-package` с флагом `-migrate` используется следующий синтаксис:

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type  
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS.** Параметры подписи определяют закрытый ключ и сертификат подписи файла AIR. Эти параметры определяют **новый** сертификат подписи (их описание приводится в разделе «[Параметры подписания кода ADT](#)» на странице 192»).
- **MIGRATION_SIGNING_OPTIONS.** Параметры подписи определяют закрытый ключ и сертификат подписи файла AIR. Эти параметры определяют **оригинальный** сертификат подписи (их описание приводится в разделе «[Параметры подписания кода ADT](#)» на странице 192»).
- Прочие параметры те же, что применяются при упаковке приложения AIR с собственным файлом установки; они описаны в разделе «[Команда ADT package](#)» на странице 177».

В следующем примере иллюстрируется вызов программы ADT с командой `-package` и флагом `-migrate` для упаковки обновленной версии приложения AIR, использующего собственное расширение платформы, и для применения к нему подписи переноса:

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore  
original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Примечание. Флаг `-migrate` команды `-package` доступен в программе ADT в версиях AIR 3.6 и более поздних.

Создание самозаверяющего сертификата с помощью ADT

Для создания допустимых установочных файлов AIR можно использовать самозаверяющие сертификаты. Но самозаверяющие сертификаты обеспечивают пользователям ограниченную гарантию безопасности. Подлинность самозаверяющих сертификатов проверить нельзя. При установке файла AIR, подписанного самозаверяющим сертификатом, его издатель отображается как «Неизвестен». Сертификат, созданный ADT, действителен в течение 5 лет.

Если вы создаете обновление для приложения AIR, подписанного самозаверяющим сертификатом, следует использовать тот же сертификат для подписи оригинального файла и обновления AIR. Сертификаты, создаваемые ADT, всегда уникальны, даже если параметры повторяются. Поэтому, если вы планируете подписывать обновления самозаверяющим сертификатом, созданным ADT, храните оригинальный сертификат в надежном месте. Кроме того, вы не сможете создать обновленный файл AIR после истечения срока действия оригинального сертификата. (Другим сертификатом можно будет подписывать новые приложения, но не новые версии этого же приложения.)

Важная информация. Из-за ограничений самозаверяющих сертификатов корпорация Adobe настоятельно рекомендует использовать для подписи открыто распространяемых приложений AIR коммерческие сертификаты, выданные известными сертифицирующими органами.

Сертификат и связанный с ним закрытый ключ, созданный ADT, хранятся в файле-хранилище типа PKCS12. Заданный пароль относится только к ключу, но не к хранилищу.

Примеры генерации сертификата

```
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
```

Чтобы подписывать файлы AIR этим сертификатом, используйте команды ADT `-package` или `-prepare`:

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl  
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

Примечание. Java 1.5 и более поздние версии не позволяют использовать в паролях для защиты файлов сертификатов PKCS12 нестандартные символы ASCII. Используйте в паролях только стандартные символы ASCII.

Глава 14. Файлы дескриптора приложения AIR

Для каждого приложения AIR требуется создать файл дескриптора приложения. Файл дескриптора приложения представляет собой документ в формате XML, который определяет основные свойства приложения.

Во многих средах разработки, поддерживающих AIR, дескриптор приложения создается автоматически при создании проекта. В противном случае необходимо создать свой собственный файл дескриптора. Образец файла дескриптора `descriptor-sample.xml` можно найти в каталоге `samples` и в AIR, и во Flex SDK.

Файл дескриптора приложения может иметь любое имя. При упаковке приложения файл дескриптора приложения переименовывается в `application.xml` и помещается в специальном каталоге в пакете AIR.

Пример дескриптора приложения

Следующий дескриптор приложения настраивает базовые свойства, которые используются большинством приложений 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>
```

Если приложение использует файл HTML в качестве содержимого корневого каталога вместо SWF-файла, меняется только элемент `<content>`.

```
<content>
  HelloWorld.html
</content>
```

Изменения в дескрипторе приложения

Дескриптор приложения AIR был изменен в следующих выпусках AIR.

Изменения дескриптора в AIR 1.1

Разрешена локализация элементов `name` и `description` с помощью элемента `text`.

Изменения дескриптора в AIR 1.5

`contentType` стал обязательным дочерним элементом для `fileType`.

Изменения дескриптора в AIR 1.5.3

Добавлен элемент `publisherID`,

Изменения дескриптора в AIR 2.0

Добавлено:

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (см. `imageNxN`)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

Изменения дескриптора в AIR 2.5

Удалено: `version`

Добавлено:

- `android`
- `extensionID`
- `extensions`
- `image36x36` (см. `imageNxN`)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

Изменения дескриптора в AIR 2.6

Добавлено:

- `image114x114` (. `imageNxN`)
- `requestedDisplayResolution`
- `softKeyboardBehavior`

Изменения дескриптора в AIR 3.0

Добавлено:

- `colorDepth`
- `direct` как допустимое значение для `renderMode`
- `renderMode` больше не игнорируется для платформ рабочих столов
- Можно указать элемент Android `<uses-sdk>` (ранее не был разрешен)

Изменения дескриптора в AIR 3.1

Добавлено:

- «`Entitlements`» на странице 235

Изменения дескриптора в AIR 3.2

Добавлено:

- `depthAndStencil`
- `supportedLanguages`

Изменения дескриптора в AIR 3.3

Добавлено:

- `aspectRatio` теперь содержит параметр `ANY`.

Изменения дескриптора в AIR 3.4

Добавлено:

- `image50x50` (. «`imageNxN`» на странице 243)
- `image58x58` (. «`imageNxN`» на странице 243)
- `image100x100` (. «`imageNxN`» на странице 243)
- `image1024x1024` (. «`imageNxN`» на странице 243)

Изменения дескриптора в AIR 3.6

Добавлено:

- Элемент «[requestedDisplayResolution](#)» на странице 254 под «[iPhone](#)» на странице 247 теперь имеет атрибут `excludeDevices`, позволяющий задавать различное разрешение (стандартное или высокое) для различных устройств iOS.
- Новый элемент «[requestedDisplayResolution](#)» на странице 254 под «[initialWindow](#)» на странице 245 указывает, какое разрешение (стандартное или высокое) следует применять на настольных платформах, например на компьютерах Mac с дисплеями высокого разрешения.

Изменения дескриптора в AIR 3.7

Добавлено:

- Элемент «[iPhone](#)» на странице 247 теперь содержит элемент «[externalSwfs](#)» на странице 237, с помощью которого можно указать список SWF-файлов, которые необходимо загрузить во время выполнения.
- Элемент «[iPhone](#)» на странице 247 теперь содержит элемент «[forceCPURenderModeForDevices](#)» на странице 240, позволяющий принудительно включить режим визуализации с использованием ЦП для указанного списка устройств.

Структура файла дескриптора приложения

Файл дескриптора приложения представляет собой документ в формате XML со следующей структурой:

```
<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> 256>...</<supportedLanguages> 256>
<supportedProfiles>...</supportedProfiles>
<versionNumber>...</versionNumber>
<versionLabel>...</versionLabel>
</application>
```

Элементы дескриптора приложения AIR

В данном справочнике элементов представлено описание всех разрешенных элементов, которые можно использовать в файле дескриптора приложения AIR.

allowBrowserInvocation

Adobe AIR 1.0 и более поздние версии — необязательно

Разрешает API-интерфейсу AIR, встроенному в браузер, обнаруживать и запускать приложение.

Если это значение равно `true`, учитывайте ограничения системы защиты. Они описаны в разделах [Вызов приложения AIR из обозревателя](#) (для разработчиков ActionScript) и [Вызов приложения AIR из обозревателя](#) (для разработчиков HTML).

Дополнительные сведения см. в разделе «[Запуск установленных приложений AIR из обозревателя](#)» на странице 275.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

`true` или `false` (по умолчанию)

Пример

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 и более поздние версии — необязательно

Разрешает добавлять элементы в файл манифеста Android. AIR создает файл `AndroidManifest.xml` для каждого пакета APK. Для добавления дополнительных элементов в дескрипторе приложения можно использовать элемент `android`. Игнорируется на всех платформах, кроме Android.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы:

- [«colorDepth»](#) на странице 230
- [«manifestAdditions»](#) на странице 248

Содержимое

Элементы, определяющие свойства, относящиеся к Android, которые требуется добавить в манифест приложения Android.

Пример

```
<android>  
  <manifestAdditions>  
    ...  
  </manifestAdditions>  
</android>
```

Дополнительные разделы справки

[«Настройки Android»](#) на странице 80

[Файл AndroidManifest.xml](#)

application

Adobe AIR 1.0 и более поздние версии — обязательно

Корневой элемент дескриптора приложения AIR.

Родительский элемент: нет

Дочерние элементы:

- [«allowBrowserInvocation»](#) на странице 225
- [«android»](#) на странице 225
- [«copyright»](#) на странице 232
- [«customUpdateUI»](#) на странице 232
- [«description»](#) на странице 233
- [«extensions»](#) на странице 236
- [«filename»](#) на странице 237
- [«fileTypes»](#) на странице 239
- [«icon»](#) на странице 242
- [«id»](#) на странице 243
- [«initialWindow»](#) на странице 245
- [«installFolder»](#) на странице 246
- [«iPhone»](#) на странице 247
- [«name»](#) на странице 251
- [«programMenuFolder»](#) на странице 252
- [«publisherID»](#) на странице 252
- [«supportedLanguages»](#) на странице 256
- [«supportedProfiles»](#) на странице 256
- [«version»](#) на странице 259
- [«versionLabel»](#) на странице 260
- [«versionNumber»](#) на странице 260

Атрибуты

`minimumPatchLevel` — минимальный уровень исправлений среды выполнения AIR, который требуется для этого приложения.

`xmlns` — атрибут пространства имен XML определяет требуемую версию среды выполнения AIR для приложения.

Пространство имен изменяется с выпуском каждой новой версии AIR (это касается только старших версий, но не младших версий и исправлений). Последний фрагмент пространства имен, например «3.0», указывает на номер версии среды выполнения, которая требуется для работы приложения.

Атрибут `xmlns` имеет следующие значения для основных выпусков 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"
```

Для созданных на основе SWF-файлов приложений указанная в дескрипторе приложения версия среды выполнения AIR определяет максимальную версию SWF, которая может быть загружена в качестве исходного содержимого приложения. Приложения, для которых указано AIR 1.0 или AIR 1.1, могут использовать в качестве исходного содержимого только файлы SWF9 (Adobe Flash Player 9) — даже при воспроизведении в среде выполнения AIR 2. Приложения, для которых указано AIR 1.5 (или более ранние версии), могут использовать в качестве исходного содержимого файлы SWF9 или SWF10 (Adobe Flash Player 10).

SWF-версия определяет, какая версия прикладных интерфейсов программирования AIR и Adobe Flash Player доступна. Если файл SWF9 используется в качестве исходного содержимого приложения AIR 1.5, это приложение имеет доступ к прикладным интерфейсам программирования только для AIR 1.1 и Adobe Flash Player 9. Более того, изменения, внесенные в работу существующих API-интерфейсов, в AIR 2.0 или проигрывателе Flash Player 10.1 не будут действовать. (Важные, связанные с безопасностью изменения в прикладных интерфейсах программирования являются исключением из этого правила и могут применяться ретроспективно в текущих или будущих исправлениях для среды выполнения.)

Для HTML-приложений версия среды выполнения, указанная в дескрипторе приложения, определяет, какие версии прикладных интерфейсов программирования AIR и Flash Player доступны приложению. Выполнение HTML, CSS и JavaScript всегда определяется версией пакета Webkit, используемого в установленной среде выполнения AIR, но не дескриптором приложения.

Если приложение AIR загружает SWF-содержимое, версия доступных для этого содержимого прикладных интерфейсов программирования AIR и Adobe Flash Player зависит от способа загрузки содержимого. Иногда эффективная версия определяется пространством имен дескриптора приложения, иногда ее можно определить по версии загружаемого содержимого, а иногда она определяется версией загруженного содержимого. В следующей таблице показано, как определяется версия прикладных интерфейсов программирования на основе метода загрузки.

Метод загрузки содержимого	Метод определения версии прикладного интерфейса программирования
Исходное содержимое, приложение на основе SWF-файла	SWF-версия загруженного файла
Исходное содержимое, приложение на основе HTML-файла	Пространство имен дескриптора приложения
SWF, загруженный SWF-содержимым	Версия загружающего содержимого

Метод загрузки содержимого	Метод определения версии прикладного интерфейса программирования
SWF-библиотека, загруженная HTML-содержимым с помощью тега <script>	Пространство имен дескриптора приложения
SWF, загруженный HTML-содержимым с помощью прикладных интерфейсов программирования AIR или Adobe Flash Player (например, flash.display.Loader)	Пространство имен дескриптора приложения
SWF, загруженный HTML-содержимым с помощью тегов <object> или <embed> (либо эквивалентных прикладных интерфейсов программирования JavaScript)	SWF-версия загруженного файла

При загрузке SWF-файла другой версии, отличающейся от загружающего содержимого, могут возникнуть две следующие проблемы:

- Загрузка более новой версии SWF более старой версией SWF — ссылки на API-интерфейсы, добавленные в более новые версии AIR и Flash Player, в загруженном содержимом будут неразрешимыми.
- Загрузка более старой версии SWF более новой версией SWF — поведение API-интерфейсов, измененных в более новых версиях AIR и Flash Player, может быть непредсказуемым для загруженного содержимого.

Содержимое

Элемент приложения содержит дочерние элементы, которые определяют свойства приложения AIR.

Пример

```
<?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 и более поздних версий, ОС iOS и Android — необязательный элемент

Определяет соотношение сторон экрана приложения.

Если параметр не указан, приложение открывается с «естественным» соотношением сторон и ориентацией устройства. Естественная ориентация на разных устройствах может отличаться. Обычно на устройствах с небольшим экраном, например на телефонах, это портретное соотношение сторон. На некоторых устройствах, таких как планшеты iPad, приложение открывается с текущей ориентацией. В среде AIR 3.3 и более поздних версий это применяется ко всему приложению, а не только к начальному экрану.

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

portrait, landscape или any

Пример

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

autoOrients

Adobe AIR 2.0 и более поздних версий, ОС iOS и Android — необязательный элемент

Указывает, изменяется ли автоматически ориентация содержимого в приложении при изменении физического положения самого устройства. Дополнительные сведения см. на веб-странице [«Ориентация рабочей области»](#).

При использовании функции автоматической ориентации попробуйте установить следующие значения для свойств Stage align и scaleMode:

```
stage.align = StageAlign.TOP_LEFT;  
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Такие настройки разрешают вращение приложения относительно левого верхнего угла и предотвращают автоматическое масштабирование содержимого приложения. Другие режимы масштабирования могут изменять содержимое для заполнения повернутой рабочей области, однако они также могут обрезать, исказить и чрезмерно сжимать содержимое. Практически всегда лучших результатов вы можете достичь при самостоятельной перерисовке или повторном размещении содержимого.

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

true или false (по умолчанию)

Пример

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

colorDepth

Adobe AIR 3 и более поздние версии — необязательно

Указывает какую глубину цвета необходимо использовать: 16 бит или 32 бита.

При использовании глубины цвета 16 бит увеличивается скорость визуализации, но ухудшается точность цветопередачи. До AIR 3 на устройствах Android всегда используется 16-битный цвет. В AIR 3 по умолчанию используется 32-битный цвет.

***Примечание.** Если в приложении используется класс StageVideo, необходимо использовать 32-битный цвет.*

Родительский элемент: [«android»](#) на странице 225

Дочерние элементы: нет

Содержимое

Одно из следующих значений:

- 16-битный
- 32-битный

Пример

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

containsVideo

Определяет, будет приложение содержать видео или нет.

Родительский элемент: [«android»](#) на странице 225

Дочерние элементы: нет

Содержимое

Одно из следующих значений:

- true
- false

Пример

```
<android>  
  <containsVideo>true</containsVideo>  
  <manifestAdditions>...</manifestAdditions>  
</android>
```

content

Adobe AIR 1.0 и более поздние версии — обязательно

Значение элемента `content` — это URL-адрес главного файла с содержимым в приложении. Это может быть файл в формате SWF или HTML. URL-адрес задается относительно корневого расположения установочной папки. (Если приложение AIR работает с ADL, URL-адрес задается относительно папки, в которой расположен файл дескриптора приложения. Указать другой корневой каталог можно с помощью параметра `root-dir` в ADL).

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

URL-адрес, указанный относительно каталога приложения. Так как значение элемента `content` считается URL-адресом, символы в имени файла должны кодироваться согласно документу [RFC 1738](#). Например, знак пробела необходимо кодировать как `%20`.

Пример

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR, версии с 1.0 по 1.1 — необязательно; AIR 1.5 и более поздние версии — обязательно

Элемент `contentType` требуется в AIR 1.5 (он был необязательным в AIR 1.0 и 1.1). Это свойство помогает некоторым операционным системам выбирать лучшее приложение для открытия файла. Значение должно быть типом содержимого MIME. Обратите внимание, что это значение игнорируется в Linux, если тип файла уже зарегистрирован и имеет назначенный тип MIME.

Родительский элемент: [«fileType»](#) на странице 238

Дочерние элементы: нет

Содержимое

Тип и подтип MIME. Дополнительные сведения о типах MIME см. в разделе [«RFC2045»](#).

Пример

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 и более поздние версии — необязательно

Информация об авторских правах для приложения AIR. В Mac OS информация об авторских правах отображается в диалоговом окне «О программе» в меню установленного приложения. В Mac OS информация об авторских правах также используется в поле NSHumanReadableCopyright файла Info.plist.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

Строка, содержащая информацию об авторских правах для приложения.

Пример

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 и более поздние версии — необязательно

Указывает, включает ли приложение собственные диалоговые окна для обновления. Если установлено значение `false`, AIR предоставляет пользователю стандартные диалоговые окна для обновления. Встроенный механизм обновления AIR может использоваться только в приложениях, которые распространяются в виде файлов AIR.

Когда в установленной версии приложения элемент `customUpdateUI` имеет значение `true` и пользователь дважды щелкает на файле AIR для получения новой версии или устанавливает приложение с помощью функции непрерывной установки, в среде выполнения открывается установленная версия приложения. Среда выполнения не открывает заданный по умолчанию установщик приложения AIR. Ход процесса обновления может определяться логикой приложения. (Идентификатор приложения и идентификатор издателя в файле AIR должны совпадать с аналогичными значениями в установленном приложении, чтобы удалось выполнить обновление).

Примечание. Механизм `customUpdateUI` включается только тогда, когда приложение уже установлено и пользователь дважды щелкает на установочном файле AIR, содержащем обновление, или выполняет обновление с помощью функции непрерывной установки. Загрузить и начать обновление в рамках логики приложения с отображением заказного пользовательского интерфейса (при необходимости) можно вне зависимости от того, имеет ли свойство `customUpdateUI` значение `true`.

Дополнительные сведения см. в разделе [«Обновление приложений AIR»](#) на странице 277.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

`true` или `false` (по умолчанию)

Пример

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 и более поздние версии — необязательно

Указывает на необходимость использования приложением глубины или трафаретного буфера. Такой тип буферов обычно используется при работе с трехмерным содержимым. По умолчанию значение этого элемента равно `false` — глубина и трафаретные буферы отключены. Необходимость этого элемента определяется тем, что буферы должны быть выделены в момент запуска приложения, до того, как будет загружено любое содержимое.

Значение, установленное для данного элемента, должно совпадать со значением, переданным в аргументе `enableDepthAndStencil` методу `Context3D.configureBackBuffer()`. Если эти значения не совпадают, AIR сообщит об ошибке.

Данный элемент применим только в случае, если `renderMode = direct`. Если значение `renderMode` отличается от `direct`, ADT генерирует ошибку 118:

```
<depthAndStencil> element unexpected for render mode cpu. It requires "direct" render mode.
```

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

`true` или `false` (по умолчанию)

Пример

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

description

Adobe AIR 1.0 и более поздние версии — необязательно

Описание приложения, отображаемое в установщике приложения AIR.

Если задан единый текстовый узел (вместо нескольких элементов `text`), установщик приложения AIR использует это описание вне зависимости от языка системы. В противном случае установщик AIR использует описание, наиболее близкое к языку пользовательского интерфейса операционной системы. Например, рассмотрим процесс установки, где элемент `description` файла дескриптора приложения содержит значение для английской локали (`en`). Установщик приложения AIR использует описание `en`, если язык интерфейса операционной системы определяется как английский (`en`). Также описание `en` используется, если языком системы является английский США (`en-US`). Однако если в системе используется английский США (`en-US`), а в файле дескриптора приложения определяются имена как для американского (`en-US`), так и британского (`en-GB`) английского, то установщик приложения AIR использует значение `en-US`. Если в приложении не задано описание, соответствующее языкам пользовательского интерфейса, установщик приложения AIR использует первое значение `description` из списка в файле дескриптора приложения.

Дополнительные сведения о разработке многоязычных приложений см. в разделе [««Локализация приложений AIR»](#) на странице 316».

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: [«text»](#) на странице 258

Содержимое

Схема дескриптора приложения AIR 1.0 позволяет задать для имени только один простой текстовый узел (а не несколько элементов `text`).

Версия AIR 1.1 (или более поздняя) позволяет задать в элементе `description` несколько языков. Атрибут `xml:lang` для каждого языка задает код языка согласно документу [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Пример

Простой текстовый узел с описанием:

```
<description>This is a sample AIR application.</description>
```

Описание с локализованными элементами `text` для английского, французского и испанского языков (действует в AIR 1.1 и более поздних версиях):

```
<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 и более поздние версии — обязательно

Описание типа файла показывается пользователю операционной системой. Локализация описания типа файла не выполняется.

См. также описание элемента «[description](#)» на странице 233, являющегося дочерним элементом для `application`

Родительский элемент:«[fileType](#)» на странице 238

Дочерние элементы: нет

Содержимое

Строка, содержащая описание содержимого файла.

Пример

```
<description>PNG image</description>
```

embedFonts

Позволяет использовать заказные шрифты для экземпляра StageText в приложении AIR. Это необязательный элемент.

Родительский элемент:«[application](#)» на странице 226

Дочерние элементы:«[font](#)» на странице 239

Содержимое

Элемент `embedFonts` может содержать любое число элементов шрифтов.

Пример

```
<embedFonts>
  <font>
    <fontPath>ttf/space age.ttf</fontPath>
    <fontName>space age</fontName>
  </font>
  <font>
    <fontPath>ttf/xminus.ttf</fontPath>
    <fontName>xminus</fontName>
  </font>
</embedFonts>
```

Entitlements

Adobe AIR 3.1 и более поздних версий, Только ОС iOS — необязательный элемент

В ОС iOS используются свойства, называемые entitlements (разрешения), обеспечивающие приложению доступ к дополнительным ресурсам и возможностям. Эта информация для мобильных приложений ОС iOS задается в элементе Entitlements.

Родительский элемент: «iPhone» на странице 247

Дочерние элементы: элементы Entitlements.plist ОС iOS

Содержимое

Содержит дочерние элементы, которые определяют пары «ключ-значение», для использования настроек Entitlements.plist приложения. Содержимое элемента Entitlements должно быть заключено в блок CDATA. Дополнительные сведения см. в разделе [Руководство по разрешениям](#) в библиотеке разработчика iOS.

Пример

```
<iphone>
...
  <Entitlements>
    <![CDATA[
      <key>aps-environment</key>
      <string>development</string>
    ]]>
  </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 и более поздние версии — обязательно

Строка расширения типа файла.

Родительский элемент: «fileType» на странице 238

Дочерние элементы: нет

Содержимое

Строка, определяющая символы расширения файла (без точки «.»).

Пример

```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 и более новых версий

Указывает идентификатор расширения ActionScript, который используется в приложении. Идентификатор определен в дескрипторе расширения.

Родительский элемент: «[extensions](#)» на странице 236

Дочерние элементы: нет

Содержимое

Строка, определяющая идентификатор расширения ActionScript.

Пример

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

extensions

Adobe AIR 2.5 и более поздние версии — необязательно

Определяет расширения ActionScript, которые используются в приложении.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: «[extensionID](#)» на странице 236

Содержимое

Дочерние элементы `extensionID`, содержащие идентификаторы расширений ActionScript из файла дескриптора расширения.

Пример

```
<extensions>
  <extensionID>extension.first</extensionID>
  <extensionID>extension.next</extensionID>
  <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Только Adobe AIR 3.7 и более поздние версии, iOS — необязательно

Задаёт имя текстового файла со списком SWF-файлов, которые должны быть настроены ADT для размещения на удалённом сервере. Можно уменьшить первоначальный размер загружаемого приложения путём упаковки подмножества SWF-файлов, используемых приложением, и загрузки оставшихся SWF-файлов (содержащих только ресурсы) во время выполнения с помощью метода `Loader.load()`. Для использования этой функции необходимо упаковать приложение таким образом, чтобы ADT переместил весь код ActionScript ByteCode (ABC) из внешне загружаемых SWF-файлов в главный SWF-файл приложения, оставляя SWF-файл, содержащий только ресурсы. Этого требует правило Apple Store, запрещающее загрузку любого кода после установки приложения.

Дополнительные сведения см. в разделе «[Уменьшение загружаемого объёма путём загрузки внешних ресурсных SWF](#)» на странице 91.

Родительский элемент: «[iPhone](#)» на странице 247, «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

Имя текстового файла со списком SWF-файлов (по одному на строке), которые необходимо разместить на удалённом сервере.

Атрибуты:

Нет.

Примеры

iOS:

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

filename

Adobe AIR 1.0 и более поздние версии — обязательно

Строка, используемая в качестве имени файла приложения (без расширения) при установке. Файл приложения запускает приложение AIR в среде выполнения. Если значение `name` не задано, `filename` также используется в качестве имени установочной папки.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: нет

Содержимое

Свойство `filename` может содержать любые символы в кодировке Unicode (UTF-8), кроме перечисленных ниже (они недопустимы для использования в именах файлов в различных системах):

Символ	Шестнадцатеричный код
<i>разные</i>	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
	x7C

Значение filename не может заканчиваться точкой.

Пример

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет тип файла, который может быть зарегистрирован для приложения.

Родительский элемент: [«fileTypes»](#) на странице 239

Дочерние элементы:

- [«contentType»](#) на странице 231
- [«description»](#) на странице 234
- [«extension»](#) на странице 235
- [«icon»](#) на странице 242
- [«name»](#) на странице 252

Содержимое

Элементы, описывающие тип файла.

Пример

```
<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 и более поздние версии — необязательно

Элемент `fileTypes` позволяет объявлять типы файлов, с которыми могут сопоставляться приложения AIR.

После установки приложения AIR, любой объявляемый тип файла регистрируется в операционной системе и, если эти типы файлов еще не сопоставлены с другим приложением, они назначаются приложению AIR. Для переопределения существующих сопоставлений типов файлов с другими приложениями используйте метод `NativeApplication.setAsDefaultApplication()` во время выполнения (желательно с разрешения пользователя).

Примечание. Методы, используемые в среде выполнения, управляют только сопоставлениями типов файлов, указанных в дескрипторе приложения.

Элемент `fileTypes` является необязательным.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: «[fileType](#)» на странице 238

Содержимое

Элемент `fileTypes` может содержать любое число элементов `fileType`.

Пример

```
<fileTypes>
  <fileType>
    <name>adobe.VideoFile</name>
    <extension>avf</extension>
    <description>Adobe Video File</description>
    <contentType>application/vnd.adobe.video-file</contentType>
    <icon>
      <image16x16>icons/AIRApp_16.png</image16x16>
      <image32x32>icons/AIRApp_32.png</image32x32>
      <image48x48>icons/AIRApp_48.png</image48x48>
      <image128x128>icons/AIRApp_128.png</image128x128>
    </icon>
  </fileType>
</fileTypes>
```

font

Описывает один заказной шрифт, который можно использовать в приложении AIR.

Родительский элемент: «[embedFonts](#)» на странице 234

Дочерние элементы: «[fontName](#)» на странице 240, «[fontPath](#)» на странице 240

Содержимое

Элементы, определяющие имя заказного шрифта и путь к нему.

Пример

```
<font>  
  <fontPath>ttf/space age.ttf</fontPath>  
  <fontName>space age</fontName>  
</font>
```

fontName

Указывает имя заказного шрифта.

Родительский элемент: «font» на странице 239

Дочерние элементы: нет

Содержимое

Имя заказного шрифта, которое должно быть задано в StageText.fontFamily

Пример

```
<fontName>space age</fontName>
```

fontPath

Указывает местоположение файла заказного шрифта.

Родительский элемент: «font» на странице 239

Дочерние элементы: нет

Содержимое

Путь к файлу заказного шрифта (относительно источника).

Пример

```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Только Adobe AIR 3.7 и более поздние версии, iOS — необязательно

Принудительное включение режима визуализации с использованием ЦП для указанного списка устройств. Эта функция позволяет выборочно включить режим визуализации с использованием графического процессора для остальных устройств iOS.

Этот тег необходимо добавить к тегу iPhone в качестве дочернего тега и указать перечень моделей устройств, разделенный пробелами. Допустимые названия моделей устройств включают:

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		

Родительский элемент: «[iPhone](#)» на странице 247, «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

Список наименований моделей устройств, разделенных пробелами.

Атрибуты:

Нет.

Примеры

iOS:

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

fullScreen

Adobe AIR 2.0 и более поздних версий, ОС iOS и Android — необязательный элемент

Указывает, запускается ли приложение в полноэкранном режиме.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

true или false (по умолчанию)

Пример

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 и более поздние версии — обязательно

Исходная высота главного окна приложения.

Если высота не задана, она определяется настройками в корневом SWF-файле, а для приложений AIR на основе HTML высоту определяет операционная система.

В AIR 2 максимальная высота окна увеличена с 2048 до 4096 пикселей.

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

Положительное целое число. Максимально разрешенное значение — 4095.

Пример

```
<height>4095</height>
```

icon

Adobe AIR 1.0 и более поздние версии — необязательно

Свойство `icon` указывает один или несколько файлов значков, которые используются в приложении. Значки не являются обязательными. Если свойство `icon` не задано, операционная система отображает значок по умолчанию.

Путь задается относительно корневого каталога приложения. Файлы значков должны быть в формате PNG. Можно задать следующие размеры значков:

Если удастся обнаружить значок для заданного размера, изображение в файле должно точно ему соответствовать. Если значки какого-либо размера отсутствуют, выбирается значок ближайшего размера и масштабируется для конкретного случая.

***Примечание.** Значки не добавляются в пакет AIR автоматически. К файлам значков внутри пакета должны вести правильные относительные пути.*

Рекомендуется создать изображение для всех размеров значков. Кроме того, проверьте, хорошо ли смотрятся значки в 16- и 32-битном режиме.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: [«imageNxN»](#) на странице 243

Содержимое

Элемент `imageNxN` для каждого требуемого размера значка.

Пример

```
<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 и более поздние версии — обязательно

Строка идентификатора для приложения, также называемая идентификатором приложения. Часто используется идентификатор, представляющий собой обратное написание строки DNS, однако данный метод не является обязательным.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: нет

Содержимое

В значении идентификатора могут использоваться следующие символы:

- 0–9
- a–z
- A–Z
- . (точка)
- - (дефис).

Значение может содержать от 1 до 212 символов. Это обязательный элемент.

Пример

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет путь к значку относительно каталога приложения.

Возможно использование следующих изображений значков, каждое из которых указывает отдельный размер значка:

- image16x16
- image29x29 (AIR 2+)
- image32x32
- image36x36 (AIR 2.5+)
- image48x48
- image50x50 (AIR 3.4+)
- image57x57 (AIR 2+)
- image58x58 (AIR 3.4+)
- image72x72 (AIR 2+)
- image100x100 (AIR 3.4+)
- image114x114 (AIR 2.6+)
- image128x128

- image144x144 (AIR 3.4+)
- image512x512 (AIR 2+)
- image1024x1024 (AIR 3.4+)

Значки должны быть созданы в формате PNG и иметь размер, заданный с помощью элемента `image`. Файлы значков следует включить в пакет приложения. Значки, ссылки на которые содержатся в дескрипторе приложения, не включаются автоматически.

Родительский элемент: `application` на странице 226

Дочерние элементы: нет

Содержимое

Путь к файлу значка может содержать любые символы в кодировке Unicode (UTF-8), кроме перечисленных ниже (они недопустимы для использования в именах файлов в различных системах):

Символ	Шестнадцатеричный код
<i>разные</i>	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
	x7C

Пример

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 и более поздние версии — необязательно

Позволяет указывать дополнительные свойства приложения iOS.

Родительский элемент: `iPhone` на странице 247

Дочерние элементы: элементы Info.plist ОС iOS

Содержимое

Содержит дочерние элементы, которые определяют пары «ключ-значение», для использования настроек Info.plist приложения. Содержимое элемента InfoAdditions должно быть заключено в блок CDATA.

Сведения о парах «ключ-значение» и их представлении в документе XML см. в «Справочнике ключей списков свойств информации» в библиотеке Apple iPhone.

Пример

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

Дополнительные разделы справки

«[Настройки iOS](#)» на странице 86

initialWindow

Adobe AIR 1.0 и более поздние версии — обязательно

Определяет основной файл содержимого и исходный вид приложения.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы (Все перечисленные ниже элементы могут использоваться в качестве дочерних элементов для элемента initialWindow. Однако некоторые элементы игнорируются, если AIR не поддерживает окна на конкретной платформе.)

Элемент	Настольный компьютер	Мобильное устройство
« aspectRatio » на странице 229	игнорируется	используется
« autoOrients » на странице 229	игнорируется	используется
« content » на странице 231	используется	используется
« depthAndStencil » на странице 233	используется	используется
« fullScreen » на странице 241	игнорируется	используется
« height » на странице 241	используется	игнорируется
« maximizable » на странице 249	используется	игнорируется
« maxSize » на странице 250	используется	игнорируется
« minimizable » на странице 250	используется	игнорируется
« minSize » на странице 250	используется	игнорируется
« renderMode » на странице 253	используется (AIR 3.0 или более поздних версий)	используется
« requestedDisplayResolution » на странице 254	используется (AIR 3.6 или более поздних версий)	игнорируется
« resizable » на странице 255	используется	игнорируется

Элемент	Настольный компьютер	Мобильное устройство
«softKeyboardBehavior» на странице 255	игнорируется	используется
«systemChrome» на странице 257	используется	игнорируется
«title» на странице 258	используется	игнорируется
«transparent» на странице 259	используется	игнорируется
«visible» на странице 260	используется	игнорируется
«width» на странице 261	используется	игнорируется
«x» на странице 261	используется	игнорируется
«y» на странице 262	используется	игнорируется

Содержимое

Дочерние элементы, определяющие вид и поведение приложения.

Пример

```
<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 и более поздние версии — необязательно

Определяет подкаталог в установочной папке по умолчанию.

В Windows установочным подкаталогом по умолчанию является папка Program Files. В Mac OS это папка /Applications. В Linux это /opt/. Например, если значение свойства `installFolder` равно "Acme", а приложение называется "ExampleApp", то оно будет установлено в каталог C:\Program Files\Acme\ExampleApp в Windows, в /Applications/Acme/Example.app в MacOS и в /opt/Acme/ExampleApp в Linux.

Свойство `installFolder` не является обязательным. Если свойство `installFolder` не задано, приложение устанавливается в подкаталог установочной папки по умолчанию на основании свойства `name`.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

Свойство `installFolder` может содержать любые символы в кодировке Unicode (UTF-8), кроме запрещенных к использованию в именах папок в различных файловых системах (список исключений см. выше в абзаце о свойстве `filename`).

Если необходимо указать вложенный каталог, используйте косую черту (/) в качестве разделителя.

Пример

```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, только iOS — необязательно

Определяет свойства приложения, специфические для ОС iOS.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы:

- [«Entitlements»](#) на странице 235
- [«externalSwfs»](#) на странице 237
- [«forceCPURenderModeForDevices»](#) на странице 240
- [«InfoAdditions»](#) на странице 244
- [«requestedDisplayResolution»](#) на странице 254

Дополнительные разделы справки

[«Настройки iOS»](#) на странице 86

manifest

Только Adobe AIR 2.5 и более поздние версии, Android — необязательно

Указывает информацию, которая должна быть добавлена в файл манифеста Android для приложения.

Родительский элемент: [«manifestAdditions»](#) на странице 248

Дочерние элементы: определены в Android SDK.

Содержимое

Технически элемент `manifest` не является частью структуры дескриптора приложения AIR. Это корневой элемент манифеста Android в формате XML. Все содержимое, включенное в элемент `manifest`, должно соответствовать схеме `AndroidManifest.xml`. При создании файла APK с помощью инструментов AIR информация из элемента `manifest` копируется в соответствующий раздел сгенерированного файла `AndroidManifest.xml` приложения.

Если указать значения манифеста Android, которые доступны только в более поздней версии SDK, чем той, которую поддерживает AIR, то при упаковке приложения необходимо установить флажок `-platformsdk` для ADT. Установите флаг для пути файловой системы к версии Android SDK, которая поддерживает добавляемые значения.

Сам элемент `manifest` должен быть заключен в блок `CDATA` в дескрипторе приложения AIR.

Пример

```
<![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>  
]]>
```

Дополнительные разделы справки

«[Настройки Android](#)» на странице 80

[Файл AndroidManifest.xml](#)

manifestAdditions

Только Adobe AIR 2.5 и более поздние версии, Android

Указывает информацию, которая должна быть добавлена в файл манифеста Android.

Любое приложение Android включает файл манифеста, в котором определены базовые свойства приложения. По своей концепции манифест Android похож на дескриптор приложения AIR. Приложения AIR for Android содержат одновременно дескриптор приложения и автоматически сгенерированный файл манифеста. При упаковке приложения AIR for Android информация из элемента `manifestAdditions` добавляется в соответствующие разделы манифеста Android.

Родительский документ: «[android](#)» на странице 225

Дочерние элементы: «[manifest](#)» на странице 247

Содержимое

Информация из элемента `manifestAdditions` добавляется в XML-документ `AndroidManifest`.

AIR настраивает несколько параметров в созданном манифесте Android, чтобы обеспечить корректную работу приложения и среды выполнения. Следующие настройки не могут быть переопределены:

Следующие атрибуты элемента `manifest` не доступны для настройки:

- `package`

- android:versionCode
- android:versionName

Следующие атрибуты элемента activity не доступны для настройки:

- android:label
- android:icon

Следующие атрибуты элемента application не доступны для настройки:

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

Пример

```
<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>
```

Дополнительные разделы справки

«[Настройки Android](#)» на странице 80

[Файл AndroidManifest.xml](#)

maximizable

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет возможность разворачивания окна.

Примечание. В некоторых операционных системах, например в Mac OS X, где разворачивание окна является операцией изменения размера, рекомендуется задать для свойств `maximizable` и `resizable` значения `false`, чтобы размер окна не менялся.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

true (по умолчанию) или false

Пример

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 и более поздние версии — необязательно

Максимальные размеры окна. Если максимальный размер не задан, он определяется операционной системой.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

Два целых числа, определяющие максимальную ширину и высоту, которые разделены пробелом.

Примечание. В AIR 2 максимальный размер окна, который поддерживается в AIR, был увеличен с 2048 x 2048 до 4096 x 4096 пикселей (поскольку точкой начала координат экрана является ноль, максимальное значение ширины и высоты равно 4095).

Пример

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет возможность сворачивания окна.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

true (по умолчанию) или false

Пример

```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 и более поздние версии — необязательно

Указывает минимально разрешенный размер окна.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

Два целых числа, определяющие минимальную ширину и высоту, которые разделены пробелом. Обратите внимание, что минимальный размер экрана, разрешенный операционной системой, имеет более высокий приоритет, чем значение, установленное в дескрипторе приложения.

Пример

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 и более поздние версии — необязательно

Название приложение, которое отображается в установщике приложения AIR.

Если элемент `name` не задан, установщик приложения AIR отображает в качестве имени приложения значение `filename`.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: «[text](#)» на странице 258

Содержимое

Если задан единый текстовый узел (вместо нескольких элементов `<text>`), установщик приложения AIR использует это имя вне зависимости от языка системы.

Схема дескриптора приложения AIR 1.0 позволяет задать для имени только один простой текстовый узел (а не несколько элементов `text`). Версия AIR 1.1 (или более поздняя) позволяет задать в элементе `name` несколько языков.

Атрибут `xml:lang` для каждого языка задает код языка согласно документу [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Установщик AIR использует имя, наиболее близкое к языку пользовательского интерфейса операционной системы. Например, рассмотрим процесс установки, где элемент `name` файла дескриптора приложения содержит значение для английской локали (`en`). Установщик приложения AIR использует значение `en`, если язык пользовательского интерфейса определяется как английский (`en`). Также имя `en` используется, если языком системы является английский США (`en-US`). Однако если в системе используется английский США (`en-US`), а в файле дескриптора приложения определяются имена как для американского (`en-US`), так и британского (`en-GB`) английского, то установщик приложения AIR использует значение `en-US`. Если в приложение не задано имя, соответствующее языку пользовательского интерфейса, установщик приложения AIR использует первое значение `name` из списка в файле дескриптора приложения.

Элемент `name` определяет только заголовок приложения AIR, используемый при установке. Установщик приложений AIR поддерживает несколько языков: английский, испанский, итальянский, китайский (традиционный), китайский (упрощенный), корейский, немецкий, португальский (Бразилия), русский, французский, японский, чешский, голландский, шведский, турецкий, польский. Установщик приложения AIR выбирает отображаемый язык (для всего текста, кроме заголовка и описания приложения) на основе языка интерфейса системы. Выбор языка не зависит от параметров в файле дескриптора приложения.

Элемент `name` *не определяет* локали, доступные для установленного и работающего приложения. Дополнительные сведения о разработке многоязычных приложений см. в разделе «[Локализация приложений AIR](#)» на странице 316».

Пример

В следующем примере имя определено с помощью простого текстового узла:

```
<name>Test Application</name>
```

В следующем примере, действительном для AIR 1.1 и более поздних версий, имя указано на трех языках (английском, французском и испанском) с помощью узлов элемента `<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 и более поздние версии — обязательно

Указывает название типа файла.

Родительский элемент: «[fileType](#)» на странице 238

Дочерние элементы: нет

Содержимое

Строка, определяющая название типа файла.

Пример

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет расположение ярлыков приложения в меню «Программы» в ОС Windows или в меню «Приложения» (Applications) в Linux (эта настройка в настоящее время не учитывается в других операционных системах).

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: нет

Содержимое

Строка, используемая в качестве значения элемента `installFolder`, может содержать любые символы в кодировке Unicode (UTF-8), кроме запрещенных к использованию в именах папок в различных файловых системах (список исключений см. выше в абзаце о свойстве `filename`). Это значение *не должно* заканчиваться косой чертой (/).

Пример

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 и более поздние версии — необязательно

Определяет идентификатор издателя для обновления приложения AIR, которое изначально было создано с помощью AIR 1.5.2 или более поздней версии.

При создании обновления для приложения следует указывать только идентификатор издателя. В качестве значения элемента `publisherID` следует указать идентификатор издателя, сгенерированный инструментом AIR для более ранней версии приложения. Идентификатор издателя для установленного приложения можно найти в папке, в которой установлено приложение, в файле `META-INF/AIR/publisherid`.

В новых приложениях, созданных с помощью AIR 1.5.3 и более поздних версий, идентификатор издателя указывать не следует.

Дополнительные сведения см. в разделе «[Об идентификаторах издателя AIR](#)» на странице 204».

Родительский элемент: `application` на странице 226

Дочерние элементы: нет

Содержимое

Строка с идентификатором издателя.

Пример

```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 и более поздние версии — необязательно

Определяет, используется ли ускорение с помощью графического процессора, если эта возможность поддерживается устройством.

Родительский элемент: `initialWindow` на странице 245

Дочерние элементы: нет

Содержимое

Одно из следующих значений:

- `auto` (по умолчанию) — в настоящее время устанавливает режим центрального процессора.
- `cpu` — аппаратное ускорение не используется.
- `direct` — компоновка визуализации осуществляется в ЦП; для блитования используется графический процессор. Доступно в AIR 3 и более поздних версий.

***Примечание.** Чтобы использовать возможности графического ускорителя в содержимом Flash с AIR для мобильных платформ, компания Adobe рекомендует использовать параметр `renderMode="direct"` (т. е. `Stage3D`), а не `renderMode="gpu"`. Компания Adobe официально поддерживает и рекомендует среды разработки `Starling (2D)` и `Away3D (3D)`. Дополнительные сведения о `Stage3D` и `Starling/Away3D` см. на странице <http://gaming.adobe.com/getstarted/>.*

- `gpu` — аппаратное ускорение используется, если этот режим доступен.

***Важная информация.** Не используйте режим визуализации с помощью графического процессора для приложений `Flex`.*

Пример

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

requestedDisplayResolution

Adobe AIR 2.6 и более поздние версии, только в ОС iOS; Adobe AIR 3.6 и более поздние версии, ОС OS X —

Необязательно

Определяет разрешение, которое используется приложением на устройствах и компьютерах с дисплеем высокого разрешения: стандартное (*standard*) или высокое (*high*). Если установлено значение *standard*, по умолчанию приложение считает, что экран имеет стандартное разрешение. Если установлен параметр *high*, в приложении можно будет использовать все доступные пиксели.

К примеру, в случае дисплея iPhone с разрешением 640x960, при значении *standard* полное разрешение экрана будет 320x480, и каждый пиксел приложения будет занимать четыре пиксела экрана. При значении *high* полное разрешение экрана будет 640x960.

На устройствах со стандартным разрешением экрана размер рабочей области соответствует размерам экрана независимо от заданных настроек.

Если элемент `requestedDisplayResolution` вложен в элемент `iPhone`, он относится к устройствам iOS. В этом случае может использоваться атрибут `excludeDevices` для указания тех устройств, к которым этот параметр не применяется.

Если элемент `requestedDisplayResolution` вложен в элемент `initialWindow`, он относится к настольным приложениям AIR на компьютерах MacBook Pro, поддерживающих дисплей высокого разрешения. Указанное значение действительно для всех собственных окон платформы в данном приложении. Вложение элемента `requestedDisplayResolution` в `initialWindow` поддерживается с версии AIR 3.6.

Родительский элемент: «[iPhone](#)» на странице 247, «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

standard (по умолчанию) или *high*.

Атрибут:

`excludeDevices` — разделяемый пробелами список наименований или префиксов моделей устройств под управлением iOS. Этот атрибут позволяет разработчикам назначать для одних устройств высокое разрешение, а для других — стандартное. Этот атрибут доступен только для ОС iOS (когда элемент `requestedDisplayResolution` размещен в элементе `iPhone`). Атрибут `excludeDevices` доступен в версии AIR 3.6 и более поздних.

Для всех устройств, наименование модели которых указано в этом атрибуте, значение `requestedDisplayResolution` будет противоположно указанному. Другими словами, если значение `requestedDisplayResolution` равно *high* (высокое разрешение), устройства-исключения, указанные в этом атрибуте, будут применять стандартное разрешение. Если значение `requestedDisplayResolution` равно *standard* (стандартное), устройства-исключения будут применять высокое разрешение.

Значениями являются наименования или префиксы моделей устройств iOS. К примеру, значение «`iPad3,1`» означает конкретно iPad 3-го поколения с поддержкой Wi-Fi (но не с поддержкой GSM или CDMA). И наоборот, значение «`iPad3`» будет означать любой iPad 3-го поколения. Неофициальный список моделей устройств iOS доступен на [странице моделей iPhone-вики](#).

Примеры

Настольный компьютер:

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

iOS:

```
<iPhone>  
  <requestedDisplayResolution excludeDevices="iPad3  
iPad4">high</requestedDisplayResolution>  
</iPhone>
```

resizable

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет возможность изменения размеров окна.

Примечание. В некоторых операционных системах, например в Mac OS X, где разворачивание окна является операцией изменения размера, рекомендуется задать для свойств `maximizeable` и `resizable` значения `false`, чтобы размер окна не менялся.

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы:

Содержимое

`true` (по умолчанию) или `false`

Пример

```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 и более поздние версии, профиль мобильного устройства — необязательно

Определяет поведение приложения по умолчанию при отображении виртуальной клавиатуры. По умолчанию окно приложения смещается вверх. Среда выполнения сохраняет фокус в текстовом поле или интерактивном объекте на экране. Используйте параметр `rap`, если в приложении нет собственной логики для обработки клавиатуры.

Автоматическое поведение можно отключить, установив для элемента `softKeyboardBehavior` значение `none`. В этом случае текстовые поля и интерактивные элементы передают событие `SoftKeyboardEvent` по нажатию программной клавиши, однако среда выполнения не сдвигает и не изменяет размер окна приложения. Видимость области ввода обеспечивает приложение.

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

`none` или `rap`. По умолчанию установлено значение `rap`.

Пример

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Дополнительные разделы справки

[SoftKeyboardEvent](#)

supportedLanguages

Adobe AIR 3.2 и более поздние версии – необязательно

Идентифицирует языки, поддерживаемые приложением. Этот элемент используется только в iOS, связанной среде выполнения Mac и приложениях Android. Элемент игнорируется приложениями любых других типов.

Если данный элемент не указан, по умолчанию упаковщик выполняет следующие действия в зависимости от типа приложения:

- iOS — все языки, поддерживаемые средой выполнения AIR, представлены в магазине приложений iOS как поддерживаемые языки приложения.
- Связанная среда Mac — приложения, упакованные в пакет связанной среды, не содержат информации о языке.
- Android — комплект приложений содержит ресурсы для всех языков, поддерживаемых исполнительной средой AIR.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

Разделенный пробелами список поддерживаемых языков. Допустимые значения языков представляют собой значения ISO 639-1, поддерживаемые средой выполнения AIR: en, de, es, fr, it, ja, ko, pt, ru, cs, nl, pl, sv, tr, zh, da, nb, iw.

Упаковщик выдает ошибку для пустого значения элемента `<supportedLanguages>`.

Примечание. В локализованных тегах (например, теге имени) значение языка игнорируется, если используется тег `<supportedLanguages>` и он не содержит данного языка. Если собственное расширение содержит ресурсы для языка, не указанного в теге `<supportedLanguages>`, то будет выдано предупреждение, а ресурсы для этого языка будут проигнорированы.

Пример

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 и более поздние версии — необязательно

Идентифицирует профили, поддерживаемые для приложения.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

Любое из следующих значений можно включить в элемент `supportedProfiles`:

- `desktop` — профиль настольного компьютера, предназначенный для приложений AIR, которые установлены на настольном компьютере с использованием файла AIR. Эти приложения не имеют доступа к классу `NativeProcess` (обеспечивающему взаимодействие со стандартными приложениями).
- `extendedDesktop` — расширенный профиль настольного компьютера, предназначенный для приложений AIR, которые установлены на настольном компьютере с помощью стандартной программы установки приложений. Эти приложения имеют доступа к классу `NativeProcess` (обеспечивающему взаимодействие со стандартными приложениями).
- `mobileDevice` — профиль мобильных устройств, предназначенный для мобильных приложений.
- `extendedMobileDevice` — профиль расширенного мобильного устройства в настоящее время не используется.

Свойство `supportedProfiles` не является обязательным. Если этот элемент не включен в файл дескриптора приложения, его можно откомпилировать и развернуть для любого профиля.

Чтобы указать несколько профилей, разделите их между собой пробелами. Например, установка следующего параметра указывает, что приложение доступно только в профилях настольного компьютера и расширенных профилях.

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

Примечание. Если приложение запускается с помощью ADL и значение параметра `ADL -profile` не указано, используется первый профиль из дескриптора приложения (если профили в дескрипторе приложения не указаны, используется профиль настольного компьютера).

Пример

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

Дополнительные разделы справки

«[Профили устройств](#)» на странице 263

«[Поддерживаемые профили](#)» на странице 79

systemChrome

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет, используются ли при создании исходного окна приложения стандартная панель заголовка, границы и элементы управления, предоставленные операционной системой.

Настройки системного хрома могут быть изменены во время выполнения.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

Одно из следующих значений:

- `none` — системный хром не предоставлен. Отображение хрома окна выполняется приложением (или платформой приложения, например Flex).

- `standard` (по умолчанию) — системный хром предоставлен операционной системой.

Пример

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 и более поздние версии — необязательно

Определяет локализованную строку.

Атрибут `xml:lang` текстового элемента задает код языка согласно документу [RFC4646](http://www.ietf.org/rfc/rfc4646.txt) (<http://www.ietf.org/rfc/rfc4646.txt>).

Установщик приложений AIR использует элемент `text` со значением атрибута `xml:lang`, которое наиболее точно соответствует языку операционной системы пользователя.

Например, можно выполнить установку, используя элемент `text` со значением «en», задающим английский язык. Установщик приложения AIR использует значение `en`, если язык пользовательского интерфейса определяется как английский (`en`). Также имя `en` используется, если языком системы является английский США (`en-US`). Однако если для пользовательского интерфейса установлен английский США (`en-US`), а в файле дескриптора приложения определяются имена как для американского (`en-US`), так и британского (`en-GB`) английского, то установщик приложения AIR использует значение `en-US`.

Если в приложении не определен элемент `text`, соответствующий языку пользовательского интерфейса, установщик приложения AIR использует первое значение элемента `name` из списка в файле дескриптора приложения.

Родительские элементы:

- «[name](#)» на странице 251
- «[description](#)» на странице 233

Дочерние элементы: нет

Содержимое

Атрибут `xml:lang`, указывающий локаль и строку локализованного текста.

Пример

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет заголовок, отображаемый в панели заголовка исходного окна приложения.

Заголовок отображается только в том случае, если для элемента `systemChrome` установлено значение `standard`.

Родительский элемент: «[initialWindow](#)» на странице 245

Дочерние элементы: нет

Содержимое

Строка с заголовком окна.

Пример

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет, применяется ли альфа-блендинг при наложении исходного окна приложения на рабочий стол.

Окно с прозрачными областями прорисовывается медленнее и требует больше ресурсов памяти. Параметр прозрачности нельзя изменить во время выполнения.

Важная информация. Свойству `transparent` можно задать значение `true`, только когда `systemChrome` имеет значение `none`.

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

`true` или `false` (по умолчанию)

Пример

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

version

Adobe AIR версии с 1.0 по 2.0 — обязательно; не разрешен в AIR 2.5 и более поздних версиях

Указывает номер версии приложения.

Строка версии зависит от приложения. AIR никоим образом не интерпретирует строку версии.

Следовательно, версия «3.0» не обязательно будет новее, чем «2.0». Примеры: «1.0», «.4», «0.5», «4.9», «1.3.4a».

В AIR 2.5 и более поздних версиях элемент `version` заменяется элементами `versionNumber` и `versionLabel`.

Родительский элемент: [«application»](#) на странице 226

Дочерние элементы: нет

Содержимое

Строка с версией приложения.

Пример

```
<version>0.1 Alpha</version>
```


versionLabel

Adobe AIR 2.5 и более поздние версии — необязательно

Указывает строку с версией в удобном для восприятия представлении.

Значение метки версии отображается в диалоговых окнах установки вместо значения элемента `versionNumber`. Если элемент `versionLabel` не используется, в обоих случаях будет использоваться элемент `versionNumber`.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: нет

Содержимое

Строка, содержащая общедоступный текст с номером версии.

Пример

```
<versionLabel>0.9 Beta</versionLabel>
```

versionNumber

Adobe AIR 2.5 и более поздние версии — обязательно

Номер версии приложения.

Родительский элемент: «[application](#)» на странице 226

Дочерние элементы: нет

Содержимое

Номер версии, содержащий последовательность из не более чем трех целых чисел, разделенных точкой. Каждое целое число должно быть в диапазоне от 0 до 999 (включительно).

Примеры

```
<versionNumber>1.0.657</versionNumber>
```

```
<versionNumber>10</versionNumber>
```

```
<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 и более поздние версии — необязательно

Определяет, является ли окно видимым сразу после его создания.

Окна AIR, включая исходное окно, по умолчанию при создании являются невидимыми. Затем окно можно отобразить, вызвав для окна метод `activate()` объекта `NativeWindow` или задав для свойства `visible` значение `true`. Иногда полезно скрывать основное окно вначале, чтобы изменения его положения, размера и компоновки его элементов не отображались.

Компонент Flex `mxml:WindowedApplication` автоматически отображает и активирует окно непосредственно перед отправкой события `applicationComplete`, кроме случаев, когда атрибут `visible` имеет значение `false` в определении MXML.

В профилях мобильных устройств, которые не поддерживают окна, элемент `visible` игнорируется.

Родительский элемент: `<initialWindow>` на странице 245

Дочерние элементы: нет

Содержимое

`true` или `false` (по умолчанию)

Пример

```
<visible>true</visible>
```

width

Adobe AIR 1.0 и более поздние версии — обязательно

Исходная ширина главного окна приложения.

Если ширина не задана, она определяется настройками в корневом SWF-файле, а для приложений AIR на основе HTML высоту определяет операционная система.

В AIR 2 максимальная ширина окна увеличена с 2048 до 4096 пикселей.

Родительский элемент: `<initialWindow>` на странице 245

Дочерние элементы: нет

Содержимое

Положительное целое число. Максимально разрешенное значение — 4095.

Пример

```
<width>1024</width>
```

X

Adobe AIR 1.0 и более поздние версии — обязательно

Положение исходного окна приложения по горизонтали.

В большинстве случаев рекомендуется вместо задания фиксированного значения рекомендуется разрешить операционной системе определять исходное положение.

Точка начала координат экрана (0,0) — это левый верхний угол основного окна экрана (определяется операционной системой).

Родительский элемент: `<initialWindow>` на странице 245

Дочерние элементы: нет

Содержимое

Целое число.

Пример

`<x>120</x>`

У

Adobe AIR 1.0 и более поздние версии — необязательно

Положение исходного окна приложения по вертикали.

В большинстве случаев рекомендуется вместо задания фиксированного значения рекомендуется разрешить операционной системе определять исходное положение.

Точка начала координат экрана (0,0) — это левый верхний угол основного окна экрана (определяется операционной системой).

Родительский элемент: [«initialWindow»](#) на странице 245

Дочерние элементы: нет

Содержимое

Целое число.

Пример

`<y>250</y>`

Глава 15. Профили устройств

Adobe AIR 2 и более поздние версии

Профиль — это механизм определения классов компьютерных устройств, на которых работает приложение. В профиле задается набор API-интерфейсов и функций, которые обычно поддерживаются определенным классом устройств. Доступны следующие профили:

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Профили для приложения можно определить в дескрипторе приложения. Пользователи смогут установить приложение на свои компьютеры и устройства, если их профили включены в дескриптор. В противном случае они не смогут установить приложение. Например, если в дескриптор приложения включен только профиль desktop, пользователи смогут установить и запускать приложение только на настольных компьютерах.

Если включен профиль, который приложение не поддерживает, в такой среде приложение может работать с ошибками. Если в дескрипторе приложения не определено ни одного профиля, среда AIR не накладывает для приложения никаких ограничений по типам устройств. Приложение может быть упаковано в любом из поддерживаемых форматов, и пользователи смогут установить его на устройства любых типов, однако приложение может работать некорректно в среде выполнения.

В возможных случаях ограничения по профилям применяются при упаковке приложения. Например, если включен только профиль extendedDesktop, приложение невозможно упаковать в виде файла AIR. Можно создать только собственный файл установщика. Аналогичным образом, если профиль mobileDevice не включен, приложение не может быть упаковано в формате Android APK.

Одно компьютерное устройство может поддерживать несколько профилей. Например, среда AIR на настольных компьютерах поддерживает приложения с профилями desktop и extendedDesktop. Однако приложения с расширенным профилем настольного компьютера могут взаимодействовать с собственными процессами и ДОЛЖНЫ быть упакованы в собственном формате установщика (exe, dmg, deb или rpm). С другой стороны, приложение с профилем настольного компьютера не может взаимодействовать с собственными процессами. Приложение с профилем настольного компьютера можно упаковать в виде файла AIR или собственного файла установщика.

Включение в профиль функции указывает на то, что эта функция преимущественно поддерживается классом устройств, для которых определен профиль. Однако это не означает, что каждое устройство с данным профилем поддерживает все функции. Например, большинство, но не все мобильные телефоны содержат акселерометр. Классы и функции, которые не являются универсальными, обычно имеют логическое свойство, которое можно включить для использования данной функции. Например, в случае акселерометра можно проверить статическое свойство `Accelerometer.isSupported`, чтобы определить, поддерживает ли данное устройство акселерометр.

Для приложений AIR, использующих элемент `supportedProfiles` в дескрипторе приложения, можно назначить следующие профили.

Настольный компьютер Профиль рабочего стола задает набор возможностей приложений AIR, устанавливаемых как файлы AIR на настольном компьютере. Эти приложения устанавливаются и выполняются на поддерживаемых платформах рабочих столов (ОС Mac OS, Windows и Linux). Приложения

AIR, созданные в версиях, предшествующих AIR 2, имеют именно этот профиль. Некоторые API в этом профиле не работают. Например, настольные приложения не могут взаимодействовать с собственными процессами.

Расширенный профиль настольного компьютера Профиль расширенного рабочего стола задает набор возможностей приложений AIR, упакованных и устанавливаемых с помощью собственной программы установки. Собственные программы установки представляют собой EXE-файлы в ОС Windows, DMG-файлы в ОС Mac OS и BIN-, DEB- или RPM-файлы в ОС Linux. Приложения с профилем расширенного настольного компьютера обладают дополнительными возможностями, которые недоступны в приложениях с профилем настольного компьютера. Дополнительные сведения см. в разделе «[Упаковка собственного установщика для настольной системы](#)» на странице 59».

Профиль мобильного устройства Профиль мобильного устройства определяет набор возможностей для приложений, устанавливаемых на мобильных устройствах, таких как сотовые телефоны и планшетные ПК. Эти приложения устанавливаются и работают на поддерживаемых мобильных платформах, включая Android, Blackberry Tablet OS и iOS.

Расширенное мобильное устройство Расширенный профиль мобильного устройства определяет расширенный набор возможностей для приложений, устанавливаемых на мобильных устройствах. В настоящее время устройства, поддерживающие этот профиль, не существуют.

Ограничение целевых профилей в файле дескриптора приложения

Adobe AIR 2 и более поздние версии

Начиная с версии AIR 2 в файле дескриптора приложения содержится элемент `supportedProfiles`, позволяющий ограничивать целевые профили. Например, установка следующего параметра указывает, что приложение можно откомпилировать и развернуть только для профиля настольного компьютера:

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

Если этот элемент настроен, приложение можно упаковать только в перечисленные профили. Значения элемента `supportedProfiles`.

- `desktop` — профиль рабочего стола
- `extendedDesktop` — профиль расширенного рабочего стола
- `mobileDevice` — профиль мобильного устройства

Элемент `supportedProfiles` необязателен. Если этот элемент не включен в файл дескриптора приложения, его можно упаковать откомпилировать и развернуть для любого профиля.

Чтобы указать несколько профилей в элементе `supportedProfiles`, разделите их между собой пробелами, как это показано ниже:

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

Возможности различных профилей

Adobe AIR 2 и более поздние версии

В следующей таблице представлен список классов и функций, которые поддерживаются не во всех профилях.

Класс или функция	desktop	extendedDesktop	mobileDevice
Акселерометр (Accelerometer.isSupported)	Нет	Нет	Проверка
Специальные возможности (Capabilities.hasAccessibility)	Да	Да	Нет
Акустическое эхоподавление (Microphone.getEnhancedMicrophone())	Да	Да	Нет
ActionScript 2	Да	Да	Нет
Матрица CacheAsBitmap	Нет	Нет	Да
Камера (Camera.isSupported)	Да	Да	Да
CameraRoll	Нет	Нет	Да
CameraUI (CameraUI.isSupported)	Нет	Нет	Да
Пакеты со связанными средами выполнения	Да	Да	Да
ContextMenu (ContextMenu.isSupported)	Да	Да	Нет
DatagramSocket (DatagramSocket.isSupported)	Да	Да	Да
DockIcon (NativeApplication.supportsDockIcon)	Проверка	Проверка	Нет
Drag-and-drop (NativeDragManager.isSupported)	Да	Да	Проверка
EncryptedLocalStore (EncryptedLocalStore.isSupported)	Да	Да	Да
Flash Access (DRMManager.isSupported)	Да	Да	Нет
GameInput (GameInput.isSupported)	Нет	Нет	Нет
Geolocation (Geolocation.isSupported)	Нет	Нет	Проверка
HTMLLoader (HTMLLoader.isSupported)	Да	Да	Нет
IME (IME.isSupported)	Да	Да	Проверка
LocalConnection (LocalConnection.isSupported)	Да	Да	Нет
Микрофон (Microphone.isSupported)	Да	Да	Проверка
Многоканальный звук (Capabilities.hasMultiChannelAudio())	Нет	Нет	Нет
Собственные расширения	Нет	Да	Да
NativeMenu (NativeMenu.isSupported)	Да	Да	Нет

Класс или функция	desktop	extendedDesktop	mobileDevice
NativeProcess (NativeProcess.isSupported)	Нет	Да	Нет
NativeWindow (NativeWindow.isSupported)	Да	Да	Нет
NetworkInfo (NetworkInfo.isSupported)	Да	Да	Проверка
Открытие файлов в приложении по умолчанию	Ограничения	Да	Нет
PrintJob (PrintJob.isSupported)	Да	Да	Нет
SecureSocket (SecureSocket.isSupported)	Да	Да	Проверка
ServerSocket (ServerSocket.isSupported)	Да	Да	Да
Шейдер	Да	Да	Ограничения
Stage3D (Stage.stage3Ds.length)	Да	Да	Да
Ориентация рабочей области (Stage.supportsOrientationChange)	Нет	Нет	Да
StageVideo	Нет	Нет	Проверка
StageWebView (StageWebView.isSupported)	Да	Да	Да
Запуск приложения при входе (NativeApplication.supportsStartAtLogin)	Да	Да	Нет
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Да	Да	Нет
Режим бездействия системы	Нет	Нет	Да
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Проверка	Проверка	Нет
Ввод с помощью Text Layout Framework	Да	Да	Нет
Updater (Updater.isSupported)	Да	Нет	Нет
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Да	Да	Нет

Обозначения, используемые в таблице:

- *Проверка* — функция поддерживается на некоторых, но не на всех устройствах профиля. Во время выполнения необходимо проверить, поддерживается ли эта функция, прежде чем использовать ее.
- *Ограничения* — функция поддерживается, но для нее действуют значительные ограничения. Дополнительные сведения см. в соответствующей документации.
- *Нет* — функция не поддерживается в профиле.
- *Да* — функция поддерживается в профиле. Обратите внимание, что аппаратные возможности отдельных компьютерных могут быть недостаточными для работы этой функции. Например, не на всех телефонах может быть камера.

Указание профилей при отладке с помощью ADL

Adobe AIR 2 и более поздние версии

ADL проверяет профили, указанные в элементе `supportedProfiles` файла дескриптора приложения. Если профили указаны, то при отладке ADL по умолчанию использует первый поддерживаемый профиль.

Можно указать профиль в сеансе отладки ADL с помощью аргумента командной строки `-profile`. (См. раздел «[AIR Debug Launcher \(ADL\)](#)» на странице 170.) Этот аргумент можно использовать независимо от того, указан ли профиль в элементе `supportedProfiles` в файле дескриптора приложения. Однако, если используется элемент `supportedProfiles`, указанный в нем профиль должен совпадать с профилем, указанным в командной строке. В противном случае ADL создает ошибку.

Глава 16. AIR.SWF, встроенный в браузер интерфейс API

Настройка файла непрерывной установки badge.swf

Помимо файла badge.swf, включенного в SDK, можно создать собственный SWF-файл для использования на странице в обозревателе. Этот SWF-файл может взаимодействовать с средой выполнения несколькими способами.

- Он может устанавливать приложение AIR. См. раздел «[Установка приложений AIR из обозревателя](#)» на странице 274».
- Он может проверять, установлено ли данное приложение AIR. См. раздел «[Проверка наличия установленного AIR с веб-страницы](#)» на странице 273».
- Он может проверять, установлена ли среда выполнения. См. раздел «[Проверка наличия установленной среды выполнения](#)» на странице 272».
- Он может запускать установленное приложение AIR в системе пользователя. См. раздел «[Запуск установленных приложений AIR из обозревателя](#)» на странице 275».

Для использования этих возможностей необходимо вызвать API-интерфейсы в SWF-файле на сайте adobe.com: air.swf. Можно настроить файл badge.swf и вызвать прикладные интерфейсы программирования air.swf из собственного SWF-файла.

Кроме того, SWF-файл, запущенный в обозревателе, может взаимодействовать с запущенным приложением AIR с помощью класса LocalConnection. Дополнительные сведения см. в разделе «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков ActionScript) или «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков HTML).

Важная информация. Функции, описанные в данном разделе (а также интерфейсы API в файле air.swf), требуют, чтобы в браузере конечного пользователя был установлен проигрыватель Adobe® Flash® 9 с обновлением 3 или более поздняя версия (в операционных системах Windows или Mac OS). В Linux для поддержки функции непрерывной установки требуется Adobe Flash Player 10 (не ниже версии 10.0.12.36). Можно написать код для проверки установленной версии Flash Player и создать интерфейс для случая, если необходимая версия не установлена. Например, если установлена более ранняя версия проигрывателя Flash Player, можно добавить ссылку для загрузки версии файла AIR (вместо использования файла badge.swf или API-интерфейса air.swf для установки приложения).

Установка приложения AIR с помощью файла badge.swf

Файл badge.swf включен в AIR SDK и Flex SDK, который позволяет легко использовать возможность автоматической установки. Файл badge.swf может установить среду выполнения и приложение AIR по ссылке с веб-страницы. Файл badge.swf и его исходный код предоставляются для распространения через веб-сайт.

Встраивание файла badge.swf на веб-страницу

- 1 Найдите следующие файлы, находящиеся в каталоге samples/badge пакета AIR SDK или Flex SDK, а затем добавьте их к своему веб-серверу.
 - badge.swf
 - default_badge.html
 - AC_RunActiveContent.js
- 2 Откройте страницу default_badge.html в текстовом редакторе.
- 3 На странице default_badge.html в функции JavaScript `AC_FL_RunContent()` настройте определение параметра `FlashVars` следующим образом:

Параметр	Описание
<code>appName</code>	Имя приложения, отображаемое в SWF-файле, если среда выполнения не установлена.
<code>appurl</code>	(Обязательное.) URL-адрес файла AIR для загрузки. Необходимо использовать абсолютный, а не относительный URL-адрес.
<code>airversion</code>	(Обязательное.) Для среды выполнения версии 1.0 укажите «1.0».
<code>imageurl</code>	URL-адрес изображения для значка (не обязательно).
<code>buttoncolor</code>	Цвет кнопки загрузки (задается в шестнадцатеричном формате, например <code>FFCC00</code>).
<code>messagecolor</code>	Цвет текстового сообщения, которое отображается под кнопкой, если среда выполнения не установлена (задается в шестнадцатеричном формате, например <code>FFCC00</code>).

- 4 Минимальный размер файла badge.swf составляет 217 пикселей в ширину и 180 в высоту. Задайте нужные значения для параметров `width` и `height` функции `AC_FL_RunContent()`.
- 5 Переименуйте файл default_badge.html и измените его код в соответствии со своими требованиями (или включите его в другую HTML-страницу).

Примечание. Для тега HTML `embed`, который загружает файл badge.swf, не устанавливайте атрибут `wmode`; оставьте для него заданное по умолчанию значение ("window"). Другие значения атрибута `wmode` помешают установке в различных системах. Также установка других значений для `wmode` может приводить к ошибке: «Error #2044: Unhandled ErrorEvent.: text=Error #2074: The stage is too small to fit the download ui» («Ошибка №2044: необработанное событие ошибки.: text=Error #2074: рабочая область недостаточна для размещения пользовательского интерфейса загрузки».)

Также можно редактировать и перекомпилировать файл badge.swf. Сведения см. в разделе «[Изменение файла badge.swf](#)» на странице 270».

Установка приложения AIR с веб-страницы по ссылке для непрерывной установки

После добавления на страницу ссылки для непрерывной установки пользователь может установить приложение AIR, щелкнув по ссылке в SWF-файле.

- 1 Перейдите на страницу HTML в веб-обозревателе, в котором установлен Adobe Flash Player (не ниже версии 9 с обновлением 3 в Windows или Mac OS или версии 10 в Linux).
- 2 На веб-странице щелкните по ссылке в файле badge.swf.
 - Если среда выполнения уже установлена, пропустите этот шаг.

- Если среда выполнения не установлена, отображается диалоговое окно, запрашивающее согласие на установку. Установите среду выполнения (см. раздел «Установка Adobe AIR» на странице 3») и переходите к следующему шагу.

3 В окне установки оставьте все параметры по умолчанию и нажмите кнопку «Продолжить».

В ОС Windows AIR автоматически выполняет следующее:

- устанавливает приложение в папку C:\Program Files\
- создает ярлык приложения на рабочем столе;
- создает ярлык в меню «Пуск»;
- записывает приложение в раздел панели управления «Установка и удаление программ»

В Mac OS программа установки добавляет приложение в каталог программ (например, /Applications в Mac OS).

На компьютере с ОС Linux AIR автоматически выполняет следующее:

- Приложение устанавливается в каталог /opt.
- создает ярлык приложения на рабочем столе;
- создает ярлык в меню «Пуск»;
- Добавляет для приложения запись в диспетчере пакетов системы

4 Выберите нужные параметры и нажмите кнопку «Установить».

5 После завершения установки нажмите кнопку «Готово».

Изменение файла badge.swf

Flex SDK и AIR SDK предоставляют исходные файлы для badge.swf. Эти файлы содержатся в папке samples/badge пакета SDK:

Исходные файлы	Описание
badge fla	Исходный файл Flash для компиляции файла badge.swf. Файл badge fla компилирует файл формата SWF 9 (для загрузки во Flash Player).
AIRBadge.as	Класс ActionScript 3.0, определяющий базовый класс, который используется в файле badge fla.

С помощью Flash Professional можно изменить дизайн графического интерфейса файла badge fla.

Функция-конструктор AIRBadge(), заданная классом AIRBadge, загружает файл air.swf, размещенный по адресу <http://airdownload.adobe.com/air/browserapi/air.swf>. Файл air.swf содержит код для использования функции непрерывной установки.

Метод onInit() (в классе AIRBadge) вызывается при успешной загрузке файла 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;
    }
}
```

Код задает глобальную переменную `_air` основному классу загруженного файла `air.swf`. Этот класс содержит ряд публичных методов, к которым обращается файл `badge.swf` для вызова функциональности непрерывной загрузки.

Метод	Описание
<code>getStatus()</code>	<p>Определяет, установлена ли на компьютер среда выполнения (или может ли она быть установлена). Дополнительные сведения см. в разделе «Проверка наличия установленной среды выполнения» на странице 272.</p> <ul style="list-style-type: none"> <code>runtimeVersion</code> — строка, указывающая версию среды выполнения (например, «1.0.М6»), которая необходима для установки приложения.
<code>installApplication()</code>	<p>Устанавливает указанное приложение на компьютер пользователя. Дополнительные сведения см. в разделе «Установка приложений AIR из обозревателя» на странице 274.</p> <ul style="list-style-type: none"> <code>url</code> — строка, задающая URL-адрес. Необходимо использовать абсолютный, а не относительный URL-адрес. <code>runtimeVersion</code> — строка, указывающая версию среды выполнения (например, «2.5»), которая необходима для установки приложения. <code>arguments</code> — аргументы, передаваемые приложению, если оно запускается после установки. Приложение запускается сразу после установки, если значение элемента <code>allowBrowserInvocation</code> равно <code>true</code> в файле дескриптора приложения. (Дополнительные сведения о файле дескриптора приложения см. в разделе «Файлы дескриптора приложения AIR» на странице 220.) Если приложение запускается в конце непрерывной установки из обозревателя (когда пользователь выбрал автоматический запуск после установки), объект приложения <code>NativeApplication</code> отправляет объект <code>BrowserInvokeEvent</code>, только если передаются какие-либо аргументы. Не забывайте об ограничениях безопасности, наложенных на данные, которые вы отправляете приложению. Дополнительные сведения см. в разделе «Запуск установленных приложений AIR из обозревателя» на странице 275».

Параметры `url` и `runtimeVersion` передаются в SWF-файл через параметры `FlashVars` в контейнере HTML-страницы.

Если приложение автоматически запускается после установки, то подключение `LocalConnection` можно использовать для связи установленного приложения с файлом `badge.swf` после его вызова. Дополнительные сведения см. в разделе «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков ActionScript) или «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков HTML).

Проверить, установлено ли приложение, можно и с помощью метода `getApplicationVersion()` в файле `air.swf`. Этот метод можно вызвать или до начала процесса установки приложения, или после его начала. Дополнительные сведения см. в разделе «[Проверка наличия установленного AIR с веб-страницы](#)» на странице 273».

Загрузка файла `air.swf`

Можно создать собственный SWF-файл, использующий API-интерфейсы файла `air.swf` для взаимодействия с средой выполнения и приложениями AIR со страницы в обозревателе. Файл `air.swf` расположен по адресу <http://airdownload.adobe.com/air/browserapi/air.swf>. Чтобы сослаться на API-интерфейсы файла `air.swf` из своего SWF-файла, загрузите файл `air.swf` на тот же домен, что и свой SWF-файл. В коде ниже показано, как загружать файл `air.swf` на тот же домен приложения, что и SWF-файл:

```
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;
}
```

Как только загружается файл `air.swf` (объект `contentLoaderInfo` объекта `Loader` отправляет событие `init`), можно вызывать любые API-интерфейсы `air.swf`, описанные в следующих разделах.

Примечание. Файл `badge.swf`, поставляемый с AIR SDK и Flex SDK, автоматически загружает файл `air.swf`. См. раздел «[Установка приложения AIR с помощью файла `badge.swf`](#)» на странице 268». Инструкции в этом разделе применимы к созданию собственного SWF-файла, загружающего файл `air.swf`.

Проверка наличия установленной среды выполнения

SWF-файл может проверять, установлена ли среда выполнения. Для этого он вызывает метод `getStatus()` в файле `air.swf`, загружаемом с домена <http://airdownload.adobe.com/air/browserapi/air.swf>. Дополнительные сведения см. в разделе «[Загрузка файла `air.swf`](#)» на странице 272».

После загрузки файла `air.swf` ваш SWF-файл может вызывать его метод `getStatus()`, как показано ниже:

```
var status:String = airSWF.getStatus();
```

Метод `getStatus()` возвращает одно из показанных ниже строковых значений, в зависимости от состояния среды выполнения:

Строковое значение	Описание
"available"	Среда выполнения может быть установлена, но в настоящий момент не установлена на компьютер.
"unavailable"	Среда выполнения не может быть установлена на компьютер.
"installed"	Среда выполнения установлена на компьютер.

Метод `getStatus()` возвращает ошибку, если требуемая версия Adobe Flash Player (не ниже версии 9 с обновлением 3 в Windows и Mac OS или версия 10 в Linux) не установлена в обозревателе.

Проверка наличия установленного AIR с веб-страницы

SWF-файл может проверять, установлено ли приложение AIR (с совпадающими ID приложения и ID издателя), с помощью метода `getApplicationVersion()` в файле `air.swf`, загружаемом с <http://airdownload.adobe.com/air/browserapi/air.swf>. Дополнительные сведения см. в разделе «[Загрузка файла air.swf](#)» на странице 272».

После загрузки файла `air.swf` SWF-файл может вызывать его метод `getApplicationVersion()`, как показано ниже:

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EED35F84C264A183921344EEA353A629FD.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.
    }
}
```

Для метода `getApplicationVersion()` предусмотрены следующие параметры:

Параметры	Описание
appID	ID данного приложения. Дополнительные сведения см. в разделе « id » на странице 243».
pubID	ID издателя данного приложения. Дополнительные сведения см. в разделе « publisherID » на странице 252». Если в рассматриваемом приложении отсутствует идентификатор издателя, задайте для параметра pubID пустую строку ("").
callback	Функция обратного вызова, играющая роль функции-обработчика. Метод <code>getApplicationVersion()</code> работает асинхронно, он вызывается при обнаружении установленной версии (или ее отсутствии). Метод обратного вызова должен включать один параметр в виде строки, содержащей версию установленного приложения. Если приложение не установлено, то функции передается значение <code>null</code> , как показано в предыдущем примере.

Метод `getApplicationVersion()` возвращает ошибку, если требуемая версия Adobe Flash Player (не ниже версии 9 с обновлением 3 в Windows и Mac OS или версия 10 в Linux) не установлена в обозревателе.

Примечание. Начиная с версии AIR 1.5.3, идентификатор издателя не используется. Идентификаторы издателей больше не назначаются приложению автоматически. В целях обеспечения обратной совместимости в приложениях по-прежнему может использоваться идентификатор издателя.

Установка приложений AIR из обозревателя

SWF-файл может устанавливать приложение AIR, вызывая метод `installApplication()` в файле `air.swf`, загружаемом с <http://airdownload.adobe.com/air/browserapi/air.swf>. Дополнительные сведения см. в разделе «[Загрузка файла air.swf](#)» на странице 272».

После загрузки файла `air.swf` ваш SWF-файл может вызывать его метод `installApplication()`, как показано ниже:

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

Метод `installApplication()` устанавливает заданное приложение на компьютер пользователя. Ниже перечислены параметры этого метода:

Параметр	Описание
url	Строка, задающая URL-адрес устанавливаемого файла AIR. Необходимо использовать абсолютный, а не относительный URL-адрес.
runtimeVersion	Строка, указывающая версию среды выполнения (например, «1.0»), которая необходима для установки приложения.
arguments	Аргументы, передаваемые приложению, если оно запускается после установки. В аргументах распознаются только алфавитно-цифровые символы. Если требуется передать другие значения, рассмотрите возможность применения схемы кодирования. Приложение запускается сразу после установки, если значение элемента <code>allowBrowserInvocation</code> равно <code>true</code> в файле дескриптора приложения. (Дополнительные сведения о файле дескриптора приложения см. в разделе « Файлы дескриптора приложения AIR » на странице 220».) Если приложение запускается в конце непрерывной установки из обозревателя (когда пользователь выбрал автоматический запуск после установки), объект приложения <code>NativeApplication</code> отправляет объект <code>BrowserInvokeEvent</code> только если были переданы какие-либо аргументы. Дополнительные сведения см. в разделе « Запуск установленных приложений AIR из обозревателя » на странице 275».

Метод `installApplication()` работает только при вызове в обработчике пользовательского события, например щелчка мыши.

Метод `installApplication()` возвращает ошибку, если требуемая версия Adobe Flash Player (не ниже версии 9 с обновлением 3 в Windows и Mac OS или версия 10 в Linux) не установлена в обозревателе.

В Mac OS для установки обновленной версии приложения пользователю требуются права, разрешающие установку в каталог приложений (и администраторские права, если требуется обновить среду выполнения). В Windows пользователю нужны права администратора.

Проверить, установлено ли приложение, можно и с помощью метода `getApplicationVersion()` файла `air.swf`. Этот метод можно вызвать или до начала процесса установки приложения, или после его начала. Дополнительные сведения см. в разделе «[Проверка наличия установленного AIR с веб-страницы](#)» на странице 273». Когда приложение уже запущено, оно может взаимодействовать с SWF-содержимым в обозревателе с помощью класса `LocalConnection`. Дополнительные сведения см. в разделе «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков ActionScript) или «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков HTML).

Запуск установленных приложений AIR из обозревателя

Для использования функции вызова из обозревателя (разрешающей запуск из обозревателя) в файл дескриптора целевого приложения необходимо включить следующий параметр:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Дополнительные сведения о файле дескриптора приложения см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».

SWF-файл может запускать в обозревателе приложение AIR, вызывая метод `launchApplication()` в файле `air.swf`, загружаемом с <http://airdownload.adobe.com/air/browserapi/air.swf>. Дополнительные сведения см. в разделе «[Загрузка файла air.swf](#)» на странице 272».

После загрузки файла `air.swf` ваш SWF-файл может вызывать его метод `launchApplication()`, как показано ниже:

```
var appID:String = "com.example.air.myTestApplication";  
var pubID:String = "02D88EED35F84C264A183921344EEA353A629FD.1";  
var arguments:Array = ["launchFromBrowser"]; // Optional  
airSWF.launchApplication(appID, pubID, arguments);
```

Метод `launchApplication()` определен на верхнем уровне файла `air.swf` (загружаемого с домена приложения пользовательского интерфейса SWF-файла). Вызов этого метода заставляет AIR запускать указанное приложение (если оно установлено и разрешена функция запуска из обозревателя, что осуществляется с помощью параметра `allowBrowserInvocation` в файле дескриптора приложения). Ниже перечислены параметры этого метода:

Параметр	Описание
appID	ID запускаемого приложения. Дополнительные сведения см. в разделе « id » на странице 243».
pubID	ID издателя запускаемого приложения. Дополнительные сведения см. в разделе « publisherID » на странице 252». Если в рассматриваемом приложении отсутствует идентификатор издателя, задайте для параметра pubID пустую строку ("")
arguments	Массив аргументов, передаваемых приложению. Объект приложения NativeApplication отправляет событие BrowserInvokeEvent, свойство arguments которого указывает на этот массив. В аргументах распознаются только алфавитно-цифровые символы. Если требуется передать другие значения, рассмотрите возможность применения схемы кодирования.

Метод `launchApplication()` работает только при вызове в обработчике пользовательского события, например щелчка мыши.

Метод `launchApplication()` возвращает ошибку, если требуемая версия Adobe Flash Player (не ниже версии 9 с обновлением 3 в Windows и Mac OS или версия 10 в Linux) не установлена в обозревателе.

Если элемент `allowBrowserInvocation` имеет значение `false` в файле дескриптора приложения, вызов метода `launchApplication()` ни к чему не приведет.

Прежде чем выводить пользовательский интерфейс для запуска приложения, может потребоваться вызвать метод `getApplicationVersion()` в файле `air.swf`. Дополнительные сведения см. в разделе «[Проверка наличия установленного AIR с веб-страницы](#)» на странице 273».

Когда приложение вызывается с помощью функции вызова из обозревателя, его объект `NativeApplication` отправляет объект `BrowserInvokeEvent`. Дополнительные сведения см. в разделе «[Вызов приложения AIR из обозревателя](#)» (для разработчиков ActionScript) или «[Вызов приложения AIR из обозревателя](#)» (для разработчиков HTML).

Если используется функция вызова обозревателя, не забывайте об ограничениях безопасности. Эти ограничения описаны в разделе «[Вызов приложения AIR из обозревателя](#)» (для разработчиков ActionScript) и «[Вызов приложения AIR из обозревателя](#)» (для разработчиков HTML).

Когда приложение уже запущено, оно может взаимодействовать с SWF-содержимым в обозревателе с помощью класса `LocalConnection`. Дополнительные сведения см. в разделе «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков ActionScript) или «[Взаимодействие с другим экземплярами Flash Player и AIR](#)» (для разработчиков HTML).

Примечание. Начиная с версии AIR 1.5.3, идентификатор издателя не используется. Идентификаторы издателей больше не назначаются приложению автоматически. В целях обеспечения обратной совместимости в приложениях по-прежнему может использоваться идентификатор издателя.

Глава 17. Обновление приложений AIR

Пользователи могут установить или обновить приложение AIR, дважды нажав файл AIR на компьютере или в обозревателе (пользуясь функцией автоматической установки). Программа установки Adobe® AIR® будет управлять установкой, уведомив пользователя о том, что выполняется обновление уже существующего приложения.

Однако обновить приложение можно и при помощи класса `Updater`. (Установленное приложение может проверять наличие новой версии для загрузки и установки.) Класс `Updater` имеет метод `update()`, который позволяет указать файл AIR на компьютере пользователя и обновить приложение до этой версии. Для использования класса `Updater` необходимо, чтобы приложение было упаковано в файл AIR. Приложения, упакованные в виде собственного исполняемого файла или пакета, должны использовать средства обновления, предоставляемые собственной платформой.

Идентификатор приложения и идентификатор издателя файла обновления AIR должны соответствовать обновляемому приложению. Идентификатор издателя можно получить из сертификата подписи. Обновление, и обновляемое приложение должны быть подписаны одним сертификатом.

В AIR 1.5.3 или более поздней версии файл дескриптора приложения содержит элемент `<publisherID>`. Этот элемент необходимо использовать, если версии приложения создавались в AIR 1.5.2 или более ранней версии. Дополнительные сведения см. в разделе «[publisherID](#)» на странице 252».

Что касается AIR 1.1 и более поздних версий, можно выполнить перенос приложения для использования нового сертификата подписи кода. Перенос приложения в целях использования новой подписи включает в себя подписывание файла обновления AIR новым и исходным сертификатом. Перенос сертификата является односторонним процессом. После переноса только файлы AIR, подписанные новым сертификатом (или обоими сертификатами), будут распознаны как обновления существующей установки.

Управление обновлением приложений может быть достаточно сложным. AIR 1.5 содержит новую инфраструктуру обновления для приложений Adobe AIR. Эта инфраструктура обеспечивает прикладные интерфейсы программирования, помогающие разработчикам создавать удобные возможности обновления в приложениях AIR.

Перенос сертификата можно выполнить для замены самозаверяющего сертификата коммерческим сертификатом подписи кода или смены одного самозаверяющего или коммерческого сертификата на другой. Если не выполнить перенос сертификата, пользователи должны будут удалить текущую версию приложения перед установкой новой. Дополнительные сведения см. в разделе «[Замена сертификатов](#)» на странице 208».

Рекомендуется включить новый механизм обновления в приложение. При появлении новой версии приложения, механизм обновления выведет пользователю приглашение на ее установку.

Программа установки приложения AIR создает файлы журнала при установке, обновлении или удалении приложения AIR. Журналы установок позволяют определить причины проблем при установке или обновлении. См. статью [Журналы установок](#).

Примечание. Новые версии среды Adobe AIR могут включать в себя обновленные версии WebKit. Обновленная версия WebKit может привести к неожиданным изменениям в содержимом HTML развернутого приложения AIR. Поэтому может потребоваться обновить приложение. Механизм обновления может уведомлять пользователей о выходе новой версии приложения. Дополнительные сведения см. в разделе [Сведения о среде HTML](#) (для разработчиков ActionScript) или [Сведения о среде HTML](#) (для разработчиков HTML).

Об обновлении приложений

Класс `Updater` (в пакете `flash.desktop`) содержит один метод `update()`, который можно использовать для обновления выполняемого приложения. Например, если на рабочем столе пользователя размещена версия файла AIR («`Sample_App_v2.air`»), следующий код обновляет приложение.

Пример 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);
```

Пример JavaScript:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Перед использованием класса `Updater` приложением пользователь или приложение должны загрузить обновленную версию файла AIR на компьютер. Дополнительные сведения см. в разделе «[Загрузка файла AIR на компьютер пользователя](#)» на странице 280».

Результаты вызова метода `Updater.update()`

Когда приложение вызывает в среде выполнения метод `update()`, среда выполнения закрывает приложение, а затем пытается установить новую версию из файла AIR. Среда выполнения проверяет идентификатор приложения и идентификатор издателя, указанные в файле AIR, на соответствие идентификатору приложения и идентификатору издателя приложения, вызывающего метод `update()`. (Сведения об идентификаторе приложения и идентификаторе издателя см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».) Она также проверяет соответствие строки `version` строке `version`, передаваемой методу `update()`. Если установка прошла успешно, среда выполнения открывает новую версию приложения. В противном случае (если установка не может быть завершена) снова открывается существующая (до установки) версия приложения.

В случае с системой Mac OS для установки обновленной версии приложения пользователь должен иметь соответствующие системные привилегии для установки в каталог приложения. В Windows и Linux пользователь должен обладать правами администратора.

Если обновленная версия приложения требует обновленной версии среды выполнения, устанавливается новая версия среды выполнения. Для обновления среды выполнения на компьютере необходимы права администратора.

При тестировании приложения в ADL вызов метода `update()` выдает исключение среды выполнения.

О строке `version`

Строка, представленная параметром `version` метода `update()`, должна соответствовать строке в элементе `version` или `versionNumber` в файле дескриптора приложения устанавливаемого файла AIR. Задание параметра `version` является обязательным в целях защиты. Заставив приложение проверять номер версии в файле AIR, можно предотвратить случайную установку старой версии приложения. (В более старой версии приложения возможны уязвимые места в системе защиты, которые были исправлены в установленном приложении.) Приложение должно сверить строку `version` в файле AIR со строкой `version` установленного приложения во избежание атак с установкой устаревшей версии.

До версии AIR 2.5 строка `version` может иметь любой формат. Например, это может выглядеть как «2.01» или как «version 2». Начиная с версии AIR 2.5 строка `version` должна содержать последовательность из не более чем трех трехзначных чисел, разделенных точкой. Например, допустимыми номерами версий могут быть следующие: «.0», «1.0» и «67.89.999». Перед обновлением приложения необходимо задать корректную строку `version`.

Если приложение Adobe AIR загружает файл AIR из Интернета, рекомендуется иметь механизм, при помощи которого веб-служба может сообщить приложению Adobe AIR номер загружаемой версии. Приложение может затем использовать эту строку в качестве параметра `version` метода `update()`. При получении файла AIR из другого источника, который не предоставляет информацию о версии файла AIR, приложение AIR может проверить файл AIR для получения информации о версии. (Файл AIR является сжатым ZIP-архивом, а файл дескриптора приложения занимает вторую позицию в архиве.)

Сведения о файле дескриптора приложения см. в разделе «[Файлы дескриптора приложения AIR](#)» на странице 220».

Процедура подписания для обновлений приложений

Публикация обновлений специальным образом усложняет управление несколькими версиями приложений, а также создает сложности в отслеживании срока действия сертификата. Срок действия сертификата может заканчиваться до публикации обновления.

Среда выполнения Adobe AIR рассматривает обновления приложений, опубликованные без подписи переноса, как новые приложения. Пользователям необходимо удалить текущую версию приложений AIR, прежде чем они смогут установить обновление.

Для разрешения проблемы загрузите все обновленные приложения с последним сертификатом на отдельный URL развертывания. Добавьте механизм, который будет напоминать вам применять подпись переноса до окончания 180-дневного льготного периода. Дополнительные сведения см. в разделе «[Подписание обновленной версии приложения AIR](#)» на странице 213».

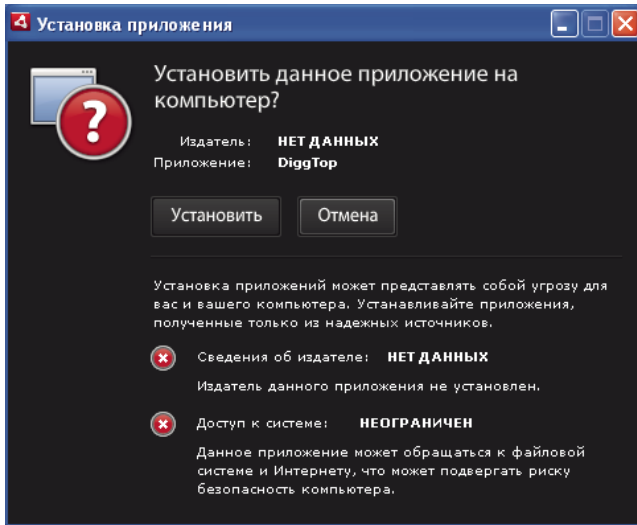
Дополнительные сведения см. в разделе «[Команды ADT](#)» на странице 176».

Выполнение следующих операций позволит упростить процесс применения подписи переноса.

- Загрузка каждого обновленного приложения на отдельный URL развертывания.
- Загрузка обновленного XML-файла дескриптор и последнего сертификата для обновления на один URL.
- Подписание обновленного приложения с помощью последнего сертификата.
- Применение подписи переноса для загруженных приложений с помощью сертификата, который использовался для подписания предыдущей версии, расположенной на другом URL.

Использование настраиваемого пользовательского интерфейса обновления приложения

AIR включает в себя интерфейс обновления по умолчанию:



Этот интерфейс используется при первой установке пользователем на компьютер версии приложения. Однако можно задать собственный интерфейс для последующих версий. Если в приложении определяется настраиваемый пользовательский интерфейс обновления, укажите элемент `customUpdateUI` в файле дескриптора приложения для текущего установленного приложения.

```
<customUpdateUI>true</customUpdateUI>
```

После установки приложения и открытия пользователем файла AIR с идентификатором приложения и идентификатором издателя, соответствующими установленному приложению, среда выполнения открывает приложение, а не программу установки приложения AIR по умолчанию. Дополнительные сведения см. в разделе «[customUpdateUI](#)» на странице 232».

При выполнении приложения оно может принять решение (при отправке объектом `NativeApplication.nativeApplication` события `load`) об установке обновления (при помощи класса `Updater`). Если принято решение в пользу обновления, приложение может предоставить пользователю собственный интерфейс установки (который отличается от стандартного интерфейса выполнения).

Загрузка файла AIR на компьютер пользователя

Для использования класса `Updater` пользователь или приложение должны сначала сохранить файл AIR на компьютер пользователя.

***Примечание.** AIR 1.5 содержит инфраструктуру обновления, помогающую разработчикам создавать удобные возможности обновления в приложениях AIR. Использовать эту инфраструктуру гораздо проще, чем напрямую применять метод `update()` класса `Update`. Дополнительные сведения см. в разделе «[Использование инфраструктуры обновления](#)» на странице 284».*

Следующий код выполняет чтение файла AIR с URL-адреса (http://example.com/air/updates/Sample_App_v2.air) и сохраняет этот файл AIR в каталог хранилища приложения.

Пример 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.");
}
```

Пример 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.");
}
```

Дополнительные сведения см. в разделе:

- [Технологический процесс чтения и записи файлов](#) (для разработчиков ActionScript)
- [Технологический процесс чтения и записи файлов](#) (для разработчиков HTML)

Проверка факта первичного запуска приложения

После обновления приложения можно отобразить экран приветствия. При запуске приложение проверяет факт первичного запуска и принимает решение о том, отображать приветствие или нет.

***Примечание.** AIR 1.5 содержит инфраструктуру обновления, помогающую разработчикам создавать удобные возможности обновления в приложениях AIR. Эта инфраструктура предоставляет удобные методы проверки, первый ли раз запущена версия приложения. Дополнительные сведения см. в разделе «[Использование инфраструктуры обновления](#)» на странице 284».*

Для этого можно сохранить файл в каталог хранилища приложения при инициализации приложения. При каждом запуске приложения оно должно проверять наличие этого файла. Если файл не существует, то приложение выполняется первый раз для данного пользователя. Существование файла означает, что приложение уже выполнялось по крайней мере один раз. Если файл существует и содержит номер более старой версии, чем номер текущей версии, значит пользователь выполняет новую версию приложения в первый раз.

Следующий пример Flex иллюстрирует данную концепцию:

```
<?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>
```

В следующем примере показана реализация этой концепции в 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; height: 100px; width: 100%;">
```



```
    }  
    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>
```

Если приложение сохраняет данные локально (например, в каталоге хранилища приложения), можно проверить ранее сохраненные данные (предыдущими версиями) при первом запуске.

Использование инфраструктуры обновления

Управление обновлениями приложений может быть трудоемкой задачей. *Инфраструктура обновления для приложений Adobe AIR* предоставляет API-интерфейсы, которые позволяют разработчикам реализовывать возможности обновления в приложениях AIR. Инфраструктура обновления AIR выполняет следующие задачи для разработчиков:

- Периодическая проверка обновлений через определенный период времени или по запросам пользователей.
- Загрузка файлов AIR (обновлений) из веб-источника.
- Оповещение пользователя о первом запуске вновь установленной версии.
- Подтверждение того, что пользователю требуется проверять наличие обновлений.
- Отображение информации о новой версии обновления пользователю.
- Отображение хода выполнения загрузки и информации об ошибках пользователю.

Инфраструктура обновления AIR предлагает пользовательский интерфейс, который может использоваться приложением. Она предоставляет пользователю основную информацию и параметры конфигурации для обновлений приложений. Приложение также может определить настраиваемый пользовательский интерфейс для использования с инфраструктурой обновления.

Инфраструктура обновления AIR позволяет сохранять информацию по обновлению версии приложения AIR в простых файлах конфигурации XML. Благодаря настройке файлов конфигурации, позволяющей включить базовый код, большинство приложений способны предложить своим конечным пользователям хорошие функциональные возможности обновления.

Даже без использования инфраструктуры обновлений Adobe AIR содержит класс Updater, который можно применять для перехода на новые версии. Класс Updater позволяет приложению выполнить обновление до версии, содержащейся в файле AIR на компьютере пользователя. Но управление обновлениями может включать задачи, более сложные, чем обновление из локально сохраненного файла AIR.

Файлы инфраструктуры обновления AIR

Инфраструктура обновления AIR содержится в каталоге frameworks/libs/air пакета AIR 2 SDK. Она включает следующие файлы:

- applicationupdater.swc: определяет основные функциональные возможности библиотеки обновления, используемые в ActionScript. В этой версии отсутствует интерфейс пользователя.
- applicationupdater.swf: определяет основные функциональные возможности библиотеки обновления, используемые в JavaScript. В этой версии отсутствует интерфейс пользователя.
- applicationupdater_ui.swc: определяет основные функциональные возможности библиотеки обновления версии Flex 4, включая интерфейс пользователя, с помощью которого приложение может отображать параметры обновления.
- applicationupdater_ui.swf: определяет основные функциональные возможности библиотеки обновления версии JavaScript, включая интерфейс пользователя, с помощью которого приложение может отображать параметры обновления.

Дополнительные сведения см. в следующих разделах:

- [«Настройка среды разработки Flex»](#) на странице 285
- [«Включение файлов инфраструктуры в HTML-приложение AIR»](#) на странице 286
- [«Простой пример: использование версии ApplicationUpdaterUI \(с интерфейсом пользователя\)»](#) на странице 286

Настройка среды разработки Flex

В файлах SWC в каталоге frameworks/libs/air пакета AIR 2 SDK определены классы, которые можно использовать при разработке в средах Flex и Flash.

Чтобы использовать инфраструктуру обновления при компиляции с помощью пакета Flex SDK, необходимо включить файл ApplicationUpdater.swc или ApplicationUpdater_UI.swc в вызов компилятора amxmlc. В следующем примере компилятор загружает файл ApplicationUpdater.swc в подкаталог lib пакета Flex SDK.

```
amxmlc -library-path+=lib/ApplicationUpdater.swc -- myApp.mxml
```

В следующем примере компилятор загружает файл ApplicationUpdater_UI.swc в подкаталог lib пакета Flex SDK.

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc -- myApp.mxml
```

При разработке с помощью Flash Builder добавьте SWC-файл на вкладке «Путь к библиотеке» в настройках путей к библиотекам Flex Builder в диалоговом окне «Свойства».

Обязательно скопируйте SWC-файлы в каталог, который будет указан при вызове компилятора amxmlc (во Flex SDK) или Flash Builder.

Включение файлов инфраструктуры в HTML-приложение AIR

В каталоге `frameworks/html` инфраструктуры обновления размещаются следующие SWF-файлы:

- `applicationupdater.swf` — определяет основные функциональные возможности библиотеки обновления без использования какого-либо интерфейса пользователя.
- `applicationupdater_ui.swf` — определяет основные функциональные возможности библиотеки обновления, включая интерфейс пользователя, с помощью которого приложение может отображать параметры обновления.

Код JavaScript в приложениях AIR может использовать классы, определенные в SWF-файлах.

Чтобы использовать инфраструктуру обновления, добавьте файл `applicationupdater.swf` или `applicationupdater_ui.swf` в каталог приложения (или его подкаталог). Затем включите в HTML-файл, который будет использовать инфраструктуру (в коде JavaScript), тег `script`, загружающий этот файл.

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

Также можно использовать тег `script` для загрузки файла `applicationupdater_ui.swf`.

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

Прикладной интерфейс программирования, определенный в этих двух файлах, описывается в оставшейся части данного документа.

Простой пример: использование версии ApplicationUpdaterUI (с интерфейсом пользователя)

Входящая в инфраструктуру обновления версия файла `ApplicationUpdater` с интерфейсом пользователя предлагает простой интерфейс, который можно использовать в приложении. Далее приведен простейший пример его использования.

Вначале создайте приложение AIR, которое вызывает инфраструктуру обновления.

- 1 Если это HTML-приложение AIR, загрузите файл `applicationupdaterui.swf`:

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

- 2 Иницируйте объект `ApplicationUpdaterUI` в программной логике приложения AIR.

В ActionScript используйте следующий программный код.

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

В JavaScript используйте следующий программный код.

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Возможно, потребуется добавить этот код в функцию инициализации, которая выполняется при загрузке приложения.

- 3 Создайте текстовый файл с именем `updateConfig.xml` и добавьте к нему следующий код:

```
<?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>
```

Отредактируйте элемент URL файла `updateConfig.xml` в соответствии с фактическим местоположением файла дескриптора на веб-сервере (см. следующую процедуру).

Свойство `delay` указывает число дней, которое приложение ждет, прежде чем выполнить следующую проверку наличия обновлений.

- 4 Добавьте файл `updateConfig.xml` к каталогу проекта своего приложения AIR.
- 5 Проверьте ссылку на файл `updateConfig.xml` в объекте `Updater` и вызовите для него метод `initialize()`.

В ActionScript используйте следующий программный код.

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

В JavaScript используйте следующий программный код.

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

- 6 Создайте вторую версию приложения AIR, отличающуюся номером версии от первого приложения. (Версия указывается в файле дескриптора приложения, в элементе `version`.)

Затем добавьте обновленную версию приложения AIR на веб-сервер.

- 1 Поместите на веб-сервер обновленную версию файла AIR.
- 2 Создайте текстовый файл с именем `updateDescriptor.2.5.xml` и добавьте к нему следующий код:

```
<?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>
```

Отредактируйте значения `versionNumber`, URL и `description` в файле `updateDescriptor.xml` для соответствия с обновленным файлом AIR. Данный формат дескриптора обновления используется приложением, в которых применяется инфраструктура обновления, включенная в пакет AIR 2.5 SDK (и более поздние версии).

- 3 Создайте текстовый файл с именем `updateDescriptor.1.0.xml` и добавьте к нему следующий код:

```
<?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>
```

Отредактируйте значения `version`, URL и `description` в файле `updateDescriptor.xml` для соответствия с обновленным файлом AIR. Данный формат дескриптора обновления используется приложением, в которых применяется инфраструктура обновления, включенная в пакет AIR 2 SDK (и более ранние версии).

Примечание. Второй файл дескриптора обновления требуется создавать только в случаях, когда реализуется поддержка обновлений для приложений, разработанных с помощью версий до AIR 2.5.

- 4 Добавьте файлы `updateDescriptor.2.5.xml` и `updateDescriptor.1.0.xml` в тот же каталог на веб-сервере, в котором содержится файл обновления AIR.

Это простейший пример, но в нем показано, как работает функция обновления, очень важная для многих приложений. В оставшейся части этого документа описывается, как оптимально использовать инфраструктуру обновления.

Другой пример использования инфраструктуры обновления см. в следующем демонстрационном приложении в центре разработки Adobe AIR:

- [Инфраструктура обновления в приложении Flash](http://www.adobe.com/go/learn_air_qs_update_framework_flash_ru)(http://www.adobe.com/go/learn_air_qs_update_framework_flash_ru)

Обновление до AIR 2.5

Поскольку начиная с версии AIR 2.5 были изменены правила назначения номеров версии, инфраструктура обновления AIR 2 не может обрабатывать информацию о версии, указанную в дескрипторе приложений AIR 2.5. По причине такой несовместимости необходимо обновить приложение для использования новой инфраструктуры обновления, ПРЕЖДЕ чем настраивать в нем использование AIR 2.5 SDK. Таким образом, процедура обновления приложения версии более ранней, чем AIR 2.5, до версии AIR 2.5 и более поздней состоит из ДВУХ шагов. Первое обновление должно использовать пространство имен AIR 2 и включать библиотеку инфраструктуры обновления AIR 2.5 (при этом пакет приложения можно создать с помощью AIR 2.5 SDK). Второе обновление может использовать пространство имен AIR 2.5 и включать новые функции приложения.

Кроме того, можно применить промежуточное обновление, которое используется только для обновления приложения AIR 2.5 с помощью класса AIR Updater.

В следующем примере представлен процесс обновления приложения с версии 1.0 до версии 2.0. В версии 1.0 используется старое пространство имен 2.0. В версии 2.0 используется пространство имен 2.5 и новые функции, реализованные с помощью API-интерфейсов AIR 2.5.

- 1 Создайте промежуточную версию приложения — версию 1.0.1 на базе приложения версии 1.0.
 - a При создании приложения используйте инфраструктуру AIR 2.5 Application Updater.

***Примечание.** Используйте файл `applicationupdater.swc` или `applicationupdater_ui.swc` для Flash-приложений AIR и файл `applicationupdater.swf` или `applicationupdater_ui.swf` для HTML-приложений AIR.*

- b Создайте файл дескриптора приложения для версии 1.0.1, используя старое пространство имен и версию, как показано ниже:

```
<?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 Создайте версию 2.0 приложения, в которой используются API-интерфейсы AIR 2.5 или пространство имен 2.5.

- 3 Создайте дескриптор для обновления приложения с версии 1.0.1 до версии 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>
```

Определение файла дескриптора обновления и добавление файла AIR на веб-сервер

При использовании инфраструктуры обновления AIR основные сведения о доступном обновлении определяются в файлах дескрипторов обновления, хранящихся на веб-сервере. Файл дескриптора обновления является простым XML-файлом. Инфраструктура обновления, входящая в приложение, проверяет этот файл, чтобы определить, не выложена ли новая версия.

Формат файла дескриптора обновления был изменен в версии AIR 2.5. В новом формате используется другое пространство имен. Исходное пространство имен: «<http://ns.adobe.com/air/framework/update/description/1.0>». Пространство имен AIR 2.5: «<http://ns.adobe.com/air/framework/update/description/2.5>».

Приложения AIR, созданные в версии среды до AIR 2.5, могут считывать только дескриптор обновления версии 1.0. Приложения AIR, созданные с использованием инфраструктуры обновления, включенной в AIR 2.5 и более поздние версии, могут считывать только дескриптор обновления версии 2.5. По причине несовместимости версий часто требуется создавать два файла дескриптора обновления. В соответствии с логикой обновления в версиях AIR 2.5 необходимо загрузить дескриптор обновления, который использует новый формат. В более ранних версиях приложения AIR следует по-прежнему использовать исходный формат. Для каждого выпускаемого обновления требуется изменять оба файла (до тех пор, пока поддерживаются версии, созданные с помощью AIR версии до 2.5).

Файл дескриптора обновления содержит следующие данные:

- `versionNumber` — новая версия приложения AIR. Используйте элемент `versionNumber` в дескрипторе обновления, который применяется для обновления приложений AIR 2.5. В качестве значения следует использовать строку из элемента `versionNumber` нового файла дескриптора приложения AIR. Если версия в файле дескриптора обновления не соответствует версии обновленного файла AIR, инфраструктура обновления вызывает исключение.
- `version` — новая версия приложения AIR. Используйте элемент `version` в дескрипторе обновления для приложений, созданных в среде AIR до версии 2.5. В качестве значения следует использовать строку из элемента `version` нового файла дескриптора приложения AIR. Если версия в файле дескриптора обновления не соответствует версии обновленного файла AIR, инфраструктура обновления вызывает исключение.
- `versionLabel` — удобочитаемая строка с версией, которая отображается для пользователей. Элемент `versionLabel` является необязательным, однако его можно указывать только в файлах дескрипторов обновлений версии 2.5. Используйте данный элемент, если в дескрипторе приложения определен элемент `versionLabel`, и установите для него такое же значение.
- `url` — Местоположение обновленного файла AIR. Это файл, который содержит обновленную версию приложения AIR.
- `description` — Подробные сведения о новой версии. Эта информация может отображаться пользователю во время выполнения обновления.

Элементы `version` и `url` обязательны. Элемент `description` не обязателен.

Здесь приведен пример файла дескриптора обновления версии 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>
```

Здесь приведен пример файла дескриптора обновления версии 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>
```

Если требуется определить `text` для нескольких языков, используйте несколько элементов `text`, определяющих атрибут `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>
```

Поместите файл дескриптора обновления вместе с обновленным файлом AIR на веб-сервер.

В создаваемом с дескриптором обновления каталоге `templates` также располагаются демонстрационные файлы дескриптора обновления. Среди них есть дескрипторы для одного языка и для нескольких языков.

Создание экземпляра объекта Updater

После загрузки инфраструктуры обновления AIR в программном коде (см. разделы «[Настройка среды разработки Flex](#)» на странице 285» и «[Включение файлов инфраструктуры в HTML-приложение AIR](#)» на странице 286») необходимо создать экземпляр объекта, как показано в следующем примере.

Пример ActionScript:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

Пример JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

В предыдущем примере используется класс `ApplicationUpdater` (который не предоставляет интерфейс пользователя). Если требуется использовать класс `ApplicationUpdaterUI` (предоставляющий интерфейс пользователя), код должен быть следующим.

Пример ActionScript:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Пример JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

В оставшихся примерах кода в этом документе предполагается, что уже создан экземпляр объекта `Updater` с именем `appUpdater`.

Настройка параметров обновления

Как класс `ApplicationUpdater`, так и класс `ApplicationUpdaterUI` можно настроить с помощью файла конфигурации, поставляемого с приложением, используя ActionScript или JavaScript.

Определение параметров обновления в файле конфигурации XML

Конфигурационный файл обновления — это XML-файл. В нем содержатся следующие элементы:

- `updateURL` — Значение типа `String`. Задает местоположения дескриптора обновления на удаленном сервере. Разрешено любое допустимое местоположение `URLRequest`. Необходимо определить свойство `updateURL` либо с помощью файла конфигурации, либо с помощью сценария (см. раздел «[Определение файла дескриптора обновления и добавление файла AIR на веб-сервер](#)» на странице 289). Необходимо определить это свойство до того, как будет использоваться объект `Updater` (перед вызовом метода `initialize()` для объекта `Updater`, описанного в разделе [Инициализация инфраструктуры обновления](#) на странице 294).
- `delay` — Значение типа `Number`. Задает интервал времени в днях (разрешены такие значения, как `0.25`) для проверки наличия обновлений. Значение `0` (задаваемое по умолчанию) означает, что объект `Updater` не выполняет периодической автоматической проверки.

Файл конфигурации для класса `ApplicationUpdaterUI` может содержать следующий элемент в дополнение к элементам `updateURL` и `delay`.

- `defaultUI`: Список элементов `dialog`. Каждый элемент `dialog` имеет атрибут `name`, соответствующий диалоговому окну в интерфейсе пользователя. Каждый элемент `dialog` имеет атрибут `visible`, определяющий, видимо ли это диалоговое окно. Значение по умолчанию — `true`. Для атрибута `name` возможны следующие значения:
 - `"checkForUpdate"` — Соответствует диалоговым окнам «Проверка обновления», «Нет обновлений» и «Ошибка обновления»
 - `"downloadUpdate"` — Соответствует диалоговому окну «Загрузка обновления»
 - `"downloadProgress"` — Соответствует диалоговым окнам «Выполнение загрузки» и «Ошибка загрузки»
 - `"installUpdate"` — Соответствует диалоговому окну «Установка обновления»
 - `"fileUpdate"` — Соответствует диалоговым окнам «Обновление файла», «Без обновления файла» и «Ошибка файла»
 - `"unexpectedError"` — Соответствует диалоговому окну «Непредвиденная ошибка»

Если установлено значение `false`, соответствующее диалоговое окно не отображается в ходе процедуры обновления.

Здесь приведен пример файла конфигурации для инфраструктуры `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>
```

Здесь приведен пример файла конфигурации для инфраструктуры `ApplicationUpdaterUI`, которая содержит определение элемента `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>
```

Укажите местоположение этого файла в свойстве `configurationFile`:

Пример ActionScript:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

Пример JavaScript:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

Каталог `templates` инфраструктуры обновления содержит демонстрационный файл конфигурации `config-template.xml`.

Код ActionScript или JavaScript для определения параметров обновления

Эти параметры конфигурации также могут быть установлены с помощью программного кода приложения, как в следующем примере:

```
appUpdater.updateURL = " http://example.com/updates/update.xml ";
appUpdater.delay = 1;
```

Свойствами объекта `updater` являются `updateURL` и `delay`. Эти свойства определяют те же параметры, что и элементы `updateURL` и `delay` в файле конфигурации: адрес URL файла дескриптора обновления и интервал времени между проверками обновлений. Если в программном коде указывается файл конфигурации и параметры, то любое свойство, установленное в коде, переопределяет соответствующие параметры файла конфигурации.

Необходимо определить свойство `updateURL` либо с помощью файла конфигурации, либо с помощью сценария (см. раздел «[Определение файла дескриптора обновления и добавление файла AIR на веб-сервер](#)» на странице 289») перед использованием объекта `updater` (до вызова метода `initialize()` объекта `updater`, как описано в разделе «[Инициализация инфраструктуры обновления](#)» на странице 294).

Инфраструктура `ApplicationUpdaterUI` определяет эти дополнительные свойства объекта `updater` следующим образом:

- `isCheckForUpdateVisible` — Соответствует диалоговым окнам «Проверка обновления», «Нет обновлений» и «Ошибка обновления»
- `isDownloadUpdateVisible` — Соответствует диалоговому окну «Загрузка обновления»
- `isDownloadProgressVisible` — Соответствует диалоговым окнам «Выполнение загрузки» и «Ошибка загрузки»
- `isInstallUpdateVisible` — Соответствует диалоговому окну «Установка обновления»
- `isFileUpdateVisible` — Соответствует диалоговым окнам «Обновление файла», «Без обновления файла» и «Ошибка файла»
- `isUnexpectedErrorVisible` — Соответствует диалоговому окну «Непредвиденная ошибка»

Каждое свойство соответствует одному или нескольким диалоговым окнам в интерфейсе пользователя `ApplicationUpdaterUI`. Каждое свойство является логическим значением, для которого по умолчанию задано значение `true`. Если установлено значение `false`, соответствующие диалоговые окна не отображаются в ходе процедуры обновления.

Эти свойства диалоговых окон переопределяют параметры, установленные в файле конфигурации обновления.

Процесс обновления

Инфраструктура обновления AIR выполняет процесс обновления, используя следующие шаги:

- 1 При инициализации объекта `updater` выполняется проверка, определяющая, проверялось ли наличие обновлений в течение заданного интервала задержки (см. раздел «[Настройка параметров обновления](#)» на странице 290»). Если проверка наличия обновлений только предстоит, процесс обновления продолжается.
- 2 Объект `updater` загружает и интерпретирует файл дескриптора обновления.
- 3 Объект `updater` загружает обновленный файл AIR.
- 4 Объект `updater` устанавливает обновленную версию приложения.

Объект `updater` отправляет соответствующие события при завершении каждого из описанных шагов. В версии `ApplicationUpdater` (без интерфейса пользователя) можно отменить события, сообщающие об успешном выполнении шагов этого процесса. Если какое-то из этих событий отменяется, то отменяется и следующий шаг в процессе. В версии `ApplicationUpdaterUI` (с интерфейсом пользователя) объект `updater` открывает диалоговые окна, позволяющие пользователю отменить или выполнить каждый шаг в процессе обновления.

Если отменяется событие, можно вызвать соответствующие методы объекта `updater` для возобновления процесса.

По мере того как версия `ApplicationUpdater` объекта `updater` выполняет различные этапы процесса обновления, она фиксирует его текущее состояние в свойстве `currentState`. Для этого свойства устанавливается переменная типа `string` со следующими возможными значениями:

- "UNINITIALIZED" — Объект `updater` не был инициализирован.
- "INITIALIZING" — Объект `updater` инициализируется.
- "READY" — Объект `updater` инициализирован
- "BEFORE_CHECKING" — Объект `updater` еще не выполнил поиск файла дескриптора обновления.
- "CHECKING" — Объект `updater` выполняет поиск файла дескриптора обновления.
- "AVAILABLE" — Файл дескриптора обновления доступен.
- "DOWNLOADING" — Объект `updater` выполняет загрузку файла AIR.
- "DOWNLOADED" — Объект `updater` выполнил загрузку файла AIR.
- "INSTALLING" — Объект `updater` выполняет установку файла AIR.
- "PENDING_INSTALLING" — Объект `updater` инициализирован и есть неоконченные обновления.

Некоторые методы объекта `updater` выполняются только в определенном состоянии этого объекта.

Инициализация инфраструктуры обновления

После установки параметров конфигурации (см. раздел «[Простой пример: использование версии ApplicationUpdaterUI \(с интерфейсом пользователя\)](#)» на странице 286») вызовите метод `initialize()` для инициализации обновления:

```
appUpdater.initialize();
```

Этот метод выполняет следующие действия:

- Инициализируется инфраструктура обновления, автоматически в синхронном режиме устанавливаются все незавершенные обновления. Необходимо вызывать этот метод во время запуска приложения, поскольку при его вызове приложение может быть перезапущено.
- Выполняется проверка, нет ли отложенных обновлений, а затем завершается их установка.
- Если в ходе процесса обновления происходит ошибка, файл обновления и информация о версии удаляются из области хранения приложения.
- Если истекает время задержки, начинается процесс обновления. Иначе происходит перезапуск таймера.

Вызов этого метода может привести к отправке следующих событий объектом `updater`:

- `UpdateEvent.INITIALIZED` — Событие отправляется, когда выполнена инициализация.
- `ErrorEvent.ERROR` — Событие отправляется, если в ходе инициализации произошла ошибка.

После отправки события `UpdateEvent.INITIALIZED` процесс обновления завершается.

При вызове метода `initialize()` объект `updater` запускает процесс обновления и выполняет все шаги, учитывая параметр задержки таймера. Однако также можно запустить процесс обновления в любой момент времени, вызвав метод `checkNow()` объекта `updater`:

```
appUpdater.checkNow();
```

Этот метод не делает ничего, если процесс обновления уже запущен. Иначе происходит запуск процесса обновления.

Объект `updater` может отправлять следующие события в результате вызова метода `checkNow()`:

- Событие `UpdateEvent.CHECK_FOR_UPDATE` отправляется точно перед тем, как будет предпринята попытка загрузить файл дескриптора обновления.

Если отменяется событие `checkForUpdate`, можно вызвать метод `checkForUpdate()` объекта `updater`. (См. следующий раздел.) Если событие не отменяется, процесс обновления выполняет проверку файла дескриптора обновления.

Управление процессом обновления в версии ApplicationUpdaterUI (с интерфейсом пользователя)

В версии `ApplicationUpdaterUI` (с интерфейсом пользователя) пользователь может отменить процесс с помощью кнопки «Отмена» в диалоговом окне интерфейса пользователя. Также процесс обновления можно отменить программно, вызвав метод `cancelUpdate()` объекта `ApplicationUpdaterUI`.

Можно задать свойства объекта `ApplicationUpdaterUI` или определить элементы в файле конфигурации обновления, указав какие подтверждения в диалоговых окнах будут отображаться объектом `updater`. Дополнительные сведения см. в разделе «[Настройка параметров обновления](#)» на странице 290».

Управление процессом обновления в версии ApplicationUpdater (без интерфейса пользователя)

Можно вызвать метод `preventDefault()` объектов событий, созданных объектом `ApplicationUpdater`, чтобы отменить шаги процесса обновления (см. раздел «[Процесс обновления](#)» на странице 293»). Отмена заданного по умолчанию поведения позволяет приложению отобразить сообщение пользователю, запросив у него подтверждение выполнения.

В следующих разделах описано, как продолжить процесс выполнения, если какой-либо шаг этого процесса был отменен.

Загрузка и интерпретирование файла дескриптора обновления

Объект `ApplicationUpdater` отправляет событие `checkForUpdate` перед началом процесса обновления, до того момента, как объект `updater` попытается загрузить файл дескриптора обновления. Если отменяется заданное по умолчанию поведение события `checkForUpdate`, то объект `updater` не загружает файл дескриптора обновления. Можно вызвать метод `checkForUpdate()`, чтобы возобновить процесс обновления:

```
appUpdater.checkForUpdate();
```

Вызов метода `checkForUpdate()` заставляет объект `updater` загружать и интерпретировать файл дескриптора обновления в асинхронном режиме. В результате вызова метода `checkForUpdate()` объект `updater` может отправлять следующие события:

- `StatusUpdateEvent.UPDATE_STATUS` — Объект `updater` успешно загрузил и интерпретировал файл дескриптора обновления. Это событие имеет следующие свойства:
 - `available` — Логическое значение. Установите значение `true`, если доступна другая версия, кроме текущей версии приложения; иначе устанавливается значение `false` (версии совпадают).
 - `version` — Значение типа `String`. Номер версии в файле дескрипторе приложения из файла обновления
 - `details` — Массив. Если нет локализованной версии описания, этот массив возвращает пустую (" ") в качестве первого элемента и описание в качестве второго элемента.

При наличии нескольких версий описания (в файле дескриптора обновления), этот массив содержит несколько подмассивов. Каждый массив содержит два элемента: первый с кодом языка (таким, как "en"), а второй с соответствующим описанием (значением типа `String`) для этого языка. См. раздел «[«Определение файла дескриптора обновления и добавление файла AIR на веб-сервер»](#)» на странице 289».

- `StatusUpdateErrorEvent.UPDATE_ERROR` — произошла ошибка, и объект `updater` не смог загрузить и интерпретировать файл дескриптора обновления.

Загрузка AIR-файла обновления

Объект `ApplicationUpdater` отправляет событие `updateStatus` после того, как объект `updater` успешно загрузит и интерпретирует файл дескриптора обновления. По умолчанию, если доступен файл обновления, должна запускаться его загрузка. Если поведение по умолчанию отменено, можно вызвать метод `downloadUpdate()`, чтобы возобновить процесс обновления.

```
appUpdater.downloadUpdate();
```

Вызов этого метода заставляет объект `updater` асинхронно загружать обновленную версию AIR-файла.

Метод `downloadUpdate()` может отправлять следующие события:

- `UpdateEvent.DOWNLOAD_START` — Установлено соединение с сервером. При использовании библиотеки `ApplicationUpdaterUI` это событие открывает диалоговое окно с индикатором выполнения для отслеживания текущего состояния загрузки.
- `ProgressEvent.PROGRESS` — Отправляется периодически, по мере выполнения загрузки файла.
- `DownloadErrorEvent.DOWNLOAD_ERROR` — Отправляется, если произошла ошибка при подключении или загрузке файла обновления. Также отправляется в случае недопустимых HTTP-состояний (например, «404 — Файл не найден»). У этого события есть свойство `errorID`, целое значение, определяющее дополнительную информацию об ошибке. Дополнительное свойство `subErrorID` может содержать дополнительные сведения об ошибке.
- `UpdateEvent.DOWNLOAD_COMPLETE` — Объект `updater` успешно загрузил и интерпретировал файл дескриптора обновления. Если это событие не отменено, в версии `ApplicationUpdater` (без интерфейса пользователя) продолжается установка обновления. В версии `ApplicationUpdaterUI` (с интерфейсом пользователя) пользователю предлагается диалоговое окно, позволяющее им подтвердить продолжение обновления.

Обновление приложения

Объект `ApplicationUpdater` отправляет событие `downloadComplete` после завершения загрузки файла обновления. Если поведение по умолчанию отменено, можно вызвать метод `installUpdate()`, чтобы возобновить процесс обновления:

```
appUpdater.installUpdate(file);
```

Вызов этого метода заставляет объект `updater` устанавливать обновленную версию AIR-файла. Этот метод включает один параметр, `file`, который является объектом `File` и содержит ссылку на AIR-файл, используемый для обновления.

Объект `ApplicationUpdater` может отправлять событие `beforeInstall` в результате вызова метода `installUpdate()`.

- `UpdateEvent.BEFORE_INSTALL` — Отправляется непосредственно перед установкой обновления. Иногда бывает полезно приостановить немедленную установку обновления, чтобы пользователь мог завершить текущую работу перед началом процесса обновления. Вызов метода `preventDefault()` объекта `Event` откладывает установку до следующего перезапуска, при этом никакой дополнительный процесс обновления не может быть запущен. (Сюда входят обновления, сведения о которых появляются в результате вызова метода `checkNow()` или в результате периодических проверок.)

Установка из произвольно выбранного AIR-файла

Можно вызвать метод `installFromAIRFile()` для установки обновленной версии из AIR-файла на компьютере пользователя.

```
appUpdater.installFromAIRFile();
```

Этот метод заставляет объект `updater` устанавливать обновленную версию приложения из указанного AIR-файла.

Метод `installFromAIRFile()` может отправлять следующие события:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS` — Отправляется после того, как `ApplicationUpdater` подтвердит успешную отправку файла с помощью метода `installFromAIRFile()`. Это событие обладает следующими свойствами.
 - `available` — Установите значение `true`, если доступна другая версия, кроме текущей версии приложения; иначе устанавливается значение `false` (версии совпадают).
 - `version` — В этой переменной типа `string` указывается новая доступная версия.
 - `path` — Передает исходный путь к файлу обновления.

Это событие можно отменить, если для свойства `Available` объекта `StatusFileUpdateEvent` установлено значение `true`. Отмена этого события отменяет процесс обновления. Вызовите метод `installUpdate()`, чтобы продолжить отмененное обновление.

- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR` — произошла ошибка и объект `updater` не смог установить приложение AIR.

Отмена процесса обновления

Можно вызвать метод `cancelUpdate()`, чтобы отменить процесс обновления.

```
appUpdater.cancelUpdate();
```

Этот метод отменяет любые незавершенные загрузки, удаляет любые незавершенные файлы загрузки и перезапускает таймер периодических проверок.

Этот метод не выполняет никаких действий, если инициализирован объект `updater`.

Локализация интерфейса ApplicationUpdaterUI

Класс `ApplicationUpdaterUI` обеспечивает интерфейс для выполнения процесса обновления, предоставляемый пользователю по умолчанию. Этот интерфейс пользователя содержит диалоговые окна, позволяющие запускать и отменять процесс, а также выполнять связанные действия.

Элемент `description` файла дескриптора обновления позволяет определить описание приложения на нескольких языках. Можно использовать несколько элементов `text`, определяющих атрибуты `lang`, как показано в следующем примере:

```
<?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>
```

Инфраструктура обновления использует описание, которое лучше соответствует цепочке языковых настроек пользователя. Дополнительные сведения см. в разделе «Определение файла дескриптора обновления и добавление файла AIR на веб-сервер».

Разработчики Flex могут добавить поддержку нового языка непосредственно к пакету `"ApplicationUpdaterDialogs"`.

Разработчики JavaScript могут вызвать метод `addResources()` объекта `updater`. Этот метод динамически добавляет новый пакет ресурсов для языка. Этот пакет ресурсов определяет локализованные строки для языка. Эти строки используются в различных текстовых полях диалоговых окон.

Разработчики JavaScript могут использовать свойство `localeChain` класса `ApplicationUpdaterUI` для определения цепочки локалей, используемой интерфейсом пользователя. Обычно разработчики JavaScript (HTML) используют это свойство. Разработчики Flex могут применять `ResourceManager` для управления цепочкой локалей.

Например, в следующем коде JavaScript определяются пакеты ресурсов для румынского и венгерского языков:

```
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;
```

Дополнительные сведения см. в описании метода `addResources()` класса `ApplicationUpdaterUI` в справочнике ActionScript® 3.0 для Adobe® Flash® Professional CS5.

Глава 18. Просмотр исходного кода

Подобно тому как пользователь может просматривать исходный код для HTML-страницы в веб-обозревателе, пользователи могут просматривать исходный код HTML-приложений AIR. Adobe® AIR® SDK включает файл AIRSourceViewer.js на языке JavaScript, который можно использовать в приложении для удобной демонстрации исходного кода конечным пользователям.

Загрузка, настройка и открытие объекта просмотра исходного кода

Код объекта просмотра исходного кода содержится в файле JavaScript (AIRSourceViewer.js), который находится в каталоге frameworks пакета AIR SDK. Чтобы использовать объект просмотра исходного кода в своем приложении, скопируйте файл AIRSourceViewer.js в каталог проекта приложения и загрузите файл с помощью тега скрипта в основной HTML-файл приложения.

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

Файл AIRSourceViewer.js определяет класс SourceViewer, к которому можно обращаться из кода JavaScript путем вызова `air.SourceViewer`.

Класс SourceViewer определяет три метода: `getDefault()`, `setup()` и `viewSource()`.

Метод	Описание
<code>getDefault()</code>	Статический метод. Запускает экземпляр SourceViewer, который можно использовать для вызова других методов.
<code>setup()</code>	Применяет настройки конфигурации к объекту просмотра исходного кода. Дополнительные сведения см. в разделе « Настройка объекта просмотра исходного кода » на странице 299»
<code>viewSource()</code>	Открывает новое окно, в котором пользователь может выполнить поиск и открыть файлы источника приложения.

Примечание. Код, использующий класс SourceViewer, должен находиться в изолированной программной среде приложения (в файле, находящемся в каталоге приложения).

Например, следующий код JavaScript создает экземпляр объекта SourceViewer и открывает окно просмотра кода источника, в котором перечислены все файлы источников:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Настройка объекта просмотра исходного кода

Применение метода `config()` передает настройки объекту просмотра исходного кода. Этот метод использует один параметр: `configObject`. Объект `configObject` содержит свойства, определяющие настройки конфигурации для объекта просмотра исходного кода. К этим свойствам относятся `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` и `typesToAdd`.

default

Строка, указывающая относительный путь к исходному файлу, который будет отображаться в объекте просмотра исходного кода.

Например, следующий код JavaScript открывает окно Source Viewer с файлом index.html в качестве отображаемого исходного файла:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

Массив строк, в котором указаны файлы или каталоги, которые будут исключены из списка отображаемых в объекте просмотра исходного кода. Эти пути указываются относительно каталога приложения.

Подстановочные знаки не поддерживаются.

Например, следующий код JavaScript открывает окно объекта просмотра исходного кода и выводит список всех файлов источников за исключением файла AIRSourceViewer.js и файлов в подкаталогах Images и Sounds:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images", "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

Массив, в каждой строке которого содержатся два числа, указывающие исходные координаты X и Y для окна объекта просмотра исходного кода.

Например, следующий код JavaScript открывает окно просмотра исходного кода с координатами на экране [40, 60] (X = 40, Y = 60):

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

Логическое значение, указывающее, должно ли окно объекта просмотра исходного кода быть модальным (TRUE) или немодальным (FALSE). По умолчанию окно объекта просмотра исходного кода модально.

Например, в следующем коде JavaScript окно объекта просмотра исходного кода открывается таким образом, что пользователь может взаимодействовать как с этим окном объекта просмотра исходного кода, так и с любыми другими окнами приложений:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

Массив строк, указывающий типы файлов, которые включены в список объекта просмотра исходного кода в дополнение к заданным по умолчанию типам.

По умолчанию в списке объекта просмотра исходного кода указаны следующие типы файлов:

- Текстовые файлы: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Файлы изображений: JPG, JPEG, PNG, GIF

Если значений не указано, в список включены все заданные по умолчанию типы (за исключением тех, что указаны в свойстве `typesToExclude`).

Например, следующий код JavaScript открывает окно объекта просмотра исходного кода и добавляет к списку файлы VCF и VCARD:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToAdd = ["text.vcf", "text.vcard"];
viewer.viewSource(configObj);
```

Для каждого типа файлов, добавляемого в список, необходимо указать «text» (для типов текстовых файлов) или «image» (для типов файлов изображений).

typesToExclude

Массив строк, указывающий типы файлов, которые будут исключены из списка объекта просмотра исходного кода.

По умолчанию в списке объекта просмотра исходного кода указаны следующие типы файлов:

- Текстовые файлы: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- Файлы изображений: JPG, JPEG, PNG, GIF

Например, следующий код JavaScript открывает окно объекта просмотра исходного кода, исключая из списка файлы GIF или XML:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

Для каждого типа файлов, добавляемого в список, необходимо указать `text` (для типов текстовых файлов) или `image` (для типов файлов изображений).

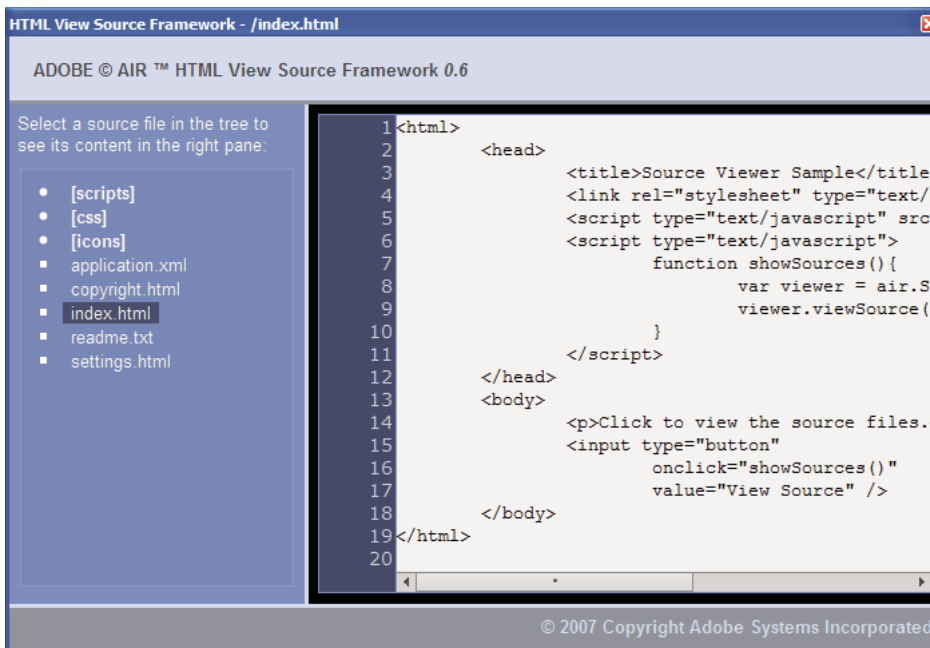
Открытие объекта просмотра исходного кода

Необходимо также добавить элемент пользовательского интерфейса, например ссылку, кнопку или команду меню, которые вызывают код объекта просмотра исходного кода при нажатии их пользователем. Например, в следующем простейшем приложении объект просмотра исходного кода открывается после того, как пользователь щелкает ссылку:

```
<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>
```

Интерфейс пользователя для объекта просмотра исходного кода

Когда приложение вызывает метод `viewSource()` объекта `SourceViewer`, приложение AIR открывает окно просмотра исходного кода. В окне отображаются список файлов источников и каталогов (слева), а также область просмотра исходного кода для выбранного файла (справа).



Каталоги в списке выделяются скобками. Пользователь может щелкнуть фигурную скобку, чтобы развернуть или свернуть список данного каталога.

Объект просмотра исходного кода может отображать источники для текстовых файлов с известными расширениями (такими как HTML, JS, TXT, XML и т.д.) или для файлов изображений с известными расширениями (JPG, JPEG, PNG и GIF). Если пользователь выбирает файл, расширение которого не известно, отображается сообщение об ошибке («Не удастся извлечь текстовое содержимое из файла данного типа»).

Любые файлы источников, которые исключены при помощи метода `setup()`, не указываются в списке (см. раздел «[Загрузка, настройка и открытие объекта просмотра исходного кода](#)» на странице 299).

Глава 19. Отладка с помощью AIR HTML Introspector

Пакет Adobe® AIR® SDK включает файл JavaScript, AIRIntrospector.js, который можно включить в приложение для помощи при отладке HTML-приложений.

О программе AIR Introspector

Программа Introspector для HTML/JavaScript-приложений Adobe AIR (называемая также AIR HTML Introspector) обеспечивает следующие полезные функциональные возможности, помогающие при разработке и отладке HTML-приложений.

- Включает инструмент самоанализа, который позволяет указать элемент пользовательского интерфейса в приложении и просмотреть его свойства разметки и DOM.
- Включает консоль для отправки ссылки на объекты для анализа, где можно настроить правильные значения и выполнить код JavaScript. Также можно сериализовать объекты при выводе их на консоль, что ограничит возможность редактирования данных. Кроме того, можно скопировать и сохранить текст с консоли.
- Консоль содержит три представления для свойств и функций DOM.
- Они позволяют редактировать атрибуты и текстовые узлы для элементов DOM.
- В консоли отображаются списки ссылок, стилей CSS, изображений и файлов JavaScript, загруженных в приложение.
- Можно просматривать HTML-источник и текущий источник разметки для интерфейса пользователя.
- Позволяет обращаться к файлам в каталоге приложения. (Эта возможность доступна только в консоли AIR HTML Introspector, открытой в изолированной программной среде приложения. Она недоступна для консолей, открытых для содержимого не в изолированной программной среде приложения.)
- Консоль включает средство просмотра объектов XMLHttpRequest и их свойств, включая свойства `responseText` и `responseXML` (если они доступны).
- Можно выполнить поиск соответствующего критериям текста в коде источника и в файлах.

Загрузка кода AIR Introspector

Код AIR Introspector содержится в файле JavaScript (AIRIntrospector.js), который находится в каталоге `frameworks` пакета AIR SDK. Чтобы использовать AIR Introspector в своем приложении, скопируйте файл AIRIntrospector.js в каталог проекта приложения и загрузите файл с помощью тега скрипта в основной HTML-файл приложения:

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

Также включите этот файл в любой HTML-файл, соответствующий различным исходным окнам приложения.

Важная информация. Включайте файл `AIRIntrospector.js` только при развертывании и отладке приложения. Удалите его из упакованного приложения AIR перед его распространением.

Файл `AIRIntrospector.js` определяет класс `Console`, к которому можно обращаться из кода JavaScript путем вызова `air.Introspector.Console`.

Примечание. Код, использующий AIR Introspector, должен находиться в изолированной программной среде приложения (в файле, находящемся в каталоге приложения).

Проверка объекта на вкладке Console (Консоль)

Класс `Console` определяет пять методов: `log()`, `warn()`, `info()`, `error()` и `dump()`.

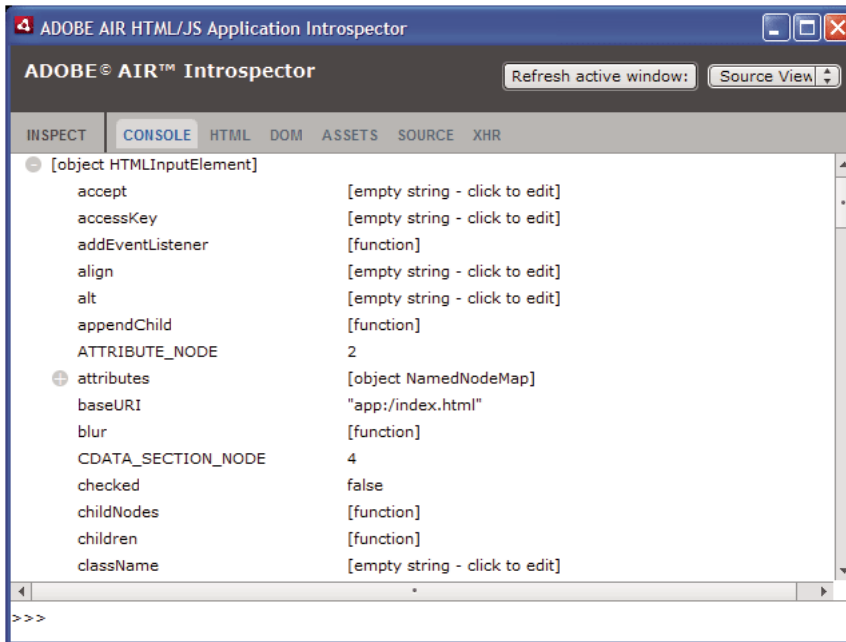
Методы `log()`, `warn()`, `info()` и `error()` все позволяют отправлять объект на вкладку Console (Консоль). Наиболее простым из этих методов является `log()`. В следующем примере простой объект, представленный переменной `test`, передается на вкладку Console (Консоль):

```
var test = "hello";  
air.Introspector.Console.log(test);
```

Но гораздо полезнее отправлять на вкладку Console (Консоль) сложные объекты. Например, следующая страница HTML содержит кнопку (`btn1`), которая вызывает функцию, отправляющую сам объект кнопки на вкладку Console (Консоль):

```
<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>
```

При нажатии этой кнопки на вкладке Console (Консоль) отображается объект btn1, можно развернуть древовидное представление этого объекта для проверки его свойств:



Можно изменить свойство объекта, щелкнув запись справа от имени свойства и изменив эту текстовую запись.

Методы `info()`, `error()` и `warn()` сходны с методом `log()`. Но при вызове этих методов на вкладке Console (Консоль) отображается значок в начале строки:

Метод	Значок
<code>info()</code>	
<code>error()</code>	
<code>warn()</code>	

Методы `log()`, `warn()`, `info()` и `error()` отправляют ссылку только на фактический объект, поэтому доступны только те свойства, которые актуальны на момент просмотра. Если требуется сериализовать этот фактический объект, используйте метод `dump()`. Метод содержит два параметра:

Параметр	Описание
<code>dumpObject</code>	Объект для сериализации.
<code>levels</code>	Максимальное число уровней, которые будут проверяться в дереве объекта (помимо корневого уровня). Значение по умолчанию 1 (означает, что показывается только один уровень за корневым уровнем дерева). Этот параметр не обязателен.

Вызов метода `dump()` сериализует объект перед его отправкой на вкладку Console (Консоль), поэтому свойства этого объекта нельзя редактировать. Например, рассмотрим следующий код:

```
var testObject = new Object();  
testObject.foo = "foo";  
testObject.bar = 234;  
air.Introspector.Console.dump(testObject);
```

При выполнении этого кода на консоли отображается объект `testObject` и его свойства, но значения свойств редактировать нельзя.

Настройка AIR Introspector

Можно настроить консоль, установив свойства глобальной переменной `AIRIntrospectorConfig`. Например, в следующем коде JavaScript выполняется настройка AIR Introspector для перехода на новую строку в столбцах через 100 символов:

```
var AIRIntrospectorConfig = new Object();  
AIRIntrospectorConfig.wrapColumns = 100;
```

Убедитесь, что свойства переменной `AIRIntrospectorConfig` установлены до загрузки файла `AIRIntrospector.js` (с помощью тега `script`).

Для переменной `AIRIntrospectorConfig` можно определить восемь следующих свойств:

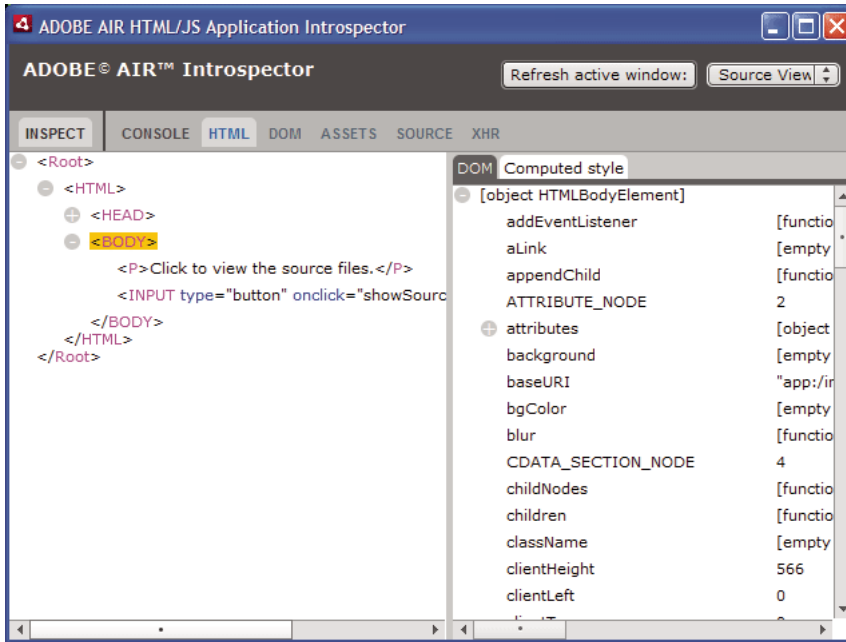
Свойство	Значение по умолчанию	Описание
<code>closeIntrospectorOnExit</code>	<code>true</code>	Задаёт, что окно Inspector будет закрываться, когда будут закрыты все другие окна приложения.
<code>debuggerKey</code>	123 (клавиша F12)	Код ключа для клавиши быстрого вызова, позволяющей отображать и скрывать окно AIR Introspector.
<code>debugRuntimeObjects</code>	<code>true</code>	Задаёт, чтобы Introspector развёртывал объекты выполнения в дополнение к объектам, определённым в JavaScript.
<code>flashTabLabels</code>	<code>true</code>	Задаёт, чтобы вкладки Console (Консоль) и XMLHttpRequest мигали, показывая, когда на них произошли изменения (например, когда изменяется текст на этих вкладках).
<code>introspectorKey</code>	122 (клавиша F11)	Код ключа для клавиши быстрого вызова, для открытия панели Inspect.
<code>showTimestamp</code>	<code>true</code>	Задаёт, чтобы в начале каждой строки на вкладке Console (Консоль) отображалась метка времени.
<code>showSender</code>	<code>true</code>	Задаёт, чтобы в начале каждой строки на вкладке Console (Консоль) отображалась информация об объекте, отправившем это сообщение.
<code>wrapColumns</code>	2000	Число столбцов, на которые разбиты исходные файлы.

Интерфейс AIR Introspector

Чтобы открыть окно AIR Introspector при отладке приложения, нажмите клавишу F12 или вызовите один из методов класса `Console` (см. раздел «[Проверка объекта на вкладке Console \(Консоль\)](#)» на странице 305»).

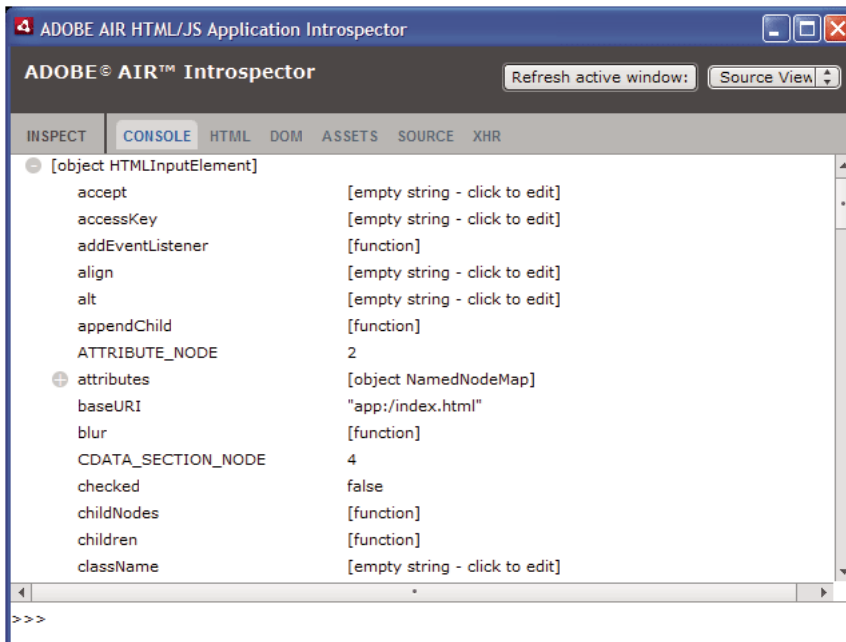
Можно настроить другую клавишу быстрого вызова вместо клавиши F12; см. раздел «[Настройка AIR Introspector](#)» на странице 307».

В окне AIR Introspector содержится шесть вкладок: Console (Консоль), HTML, DOM, Assets, Source и XHR, как показано на следующем рисунке:



Вкладка Console (Консоль)

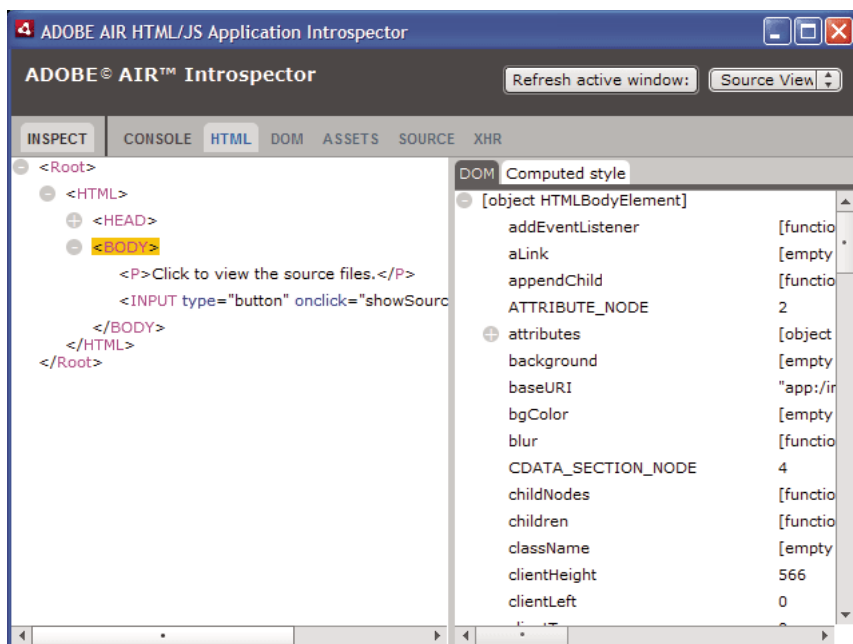
На вкладке Console (Консоль) отображаются значения свойств, переданные как параметры одному из методов класса `air.Introspector.Console`. Дополнительные сведения см. в разделе «[Проверка объекта на вкладке Console \(Консоль\)](#)» на странице 305».



- Чтобы очистить консоль, щелкните текст правой кнопкой мыши и выберите Clear Console (Очистить консоль).
- Чтобы сохранить текст из вкладки Console (Консоль) в файл, щелкните вкладку Console (Консоль) правой кнопкой мыши и выберите Save Console To File (Сохранить консоль в файл).
- Чтобы сохранить текст из вкладки Console (Консоль) в буфер обмена, щелкните вкладку Console (Консоль) правой кнопкой мыши и выберите Save Console To Clipboard (Сохранить консоль в буфер обмена). Чтобы скопировать в буфер обмена только выделенный текст, щелкните текст правой кнопкой мыши и выберите команду «Копировать».
- Чтобы сохранить текст из класса Console в файл, щелкните вкладку Console (Консоль) правой кнопкой мыши и выберите Save Console To File (Сохранить консоль в файл).
- Чтобы выполнить поиск соответствующего критерию текста, отображаемого на вкладке, нажмите CTRL+F в Windows или Command+F в Mac OS. (Узлы дерева, которые не отображаются, не доступны для поиска.)

Вкладка HTML

Вкладка HTML позволяет просматривать весь HTML DOM в древовидной структуре. Щелкните элемент, чтобы просмотреть его свойства в правой части вкладки. Щелкните значки «+» и «-», чтобы развернуть или свернуть узел в дереве.



Можно отредактировать любой атрибут или текстовый элемент на вкладке HTML и измененное значение отобразится в приложении.

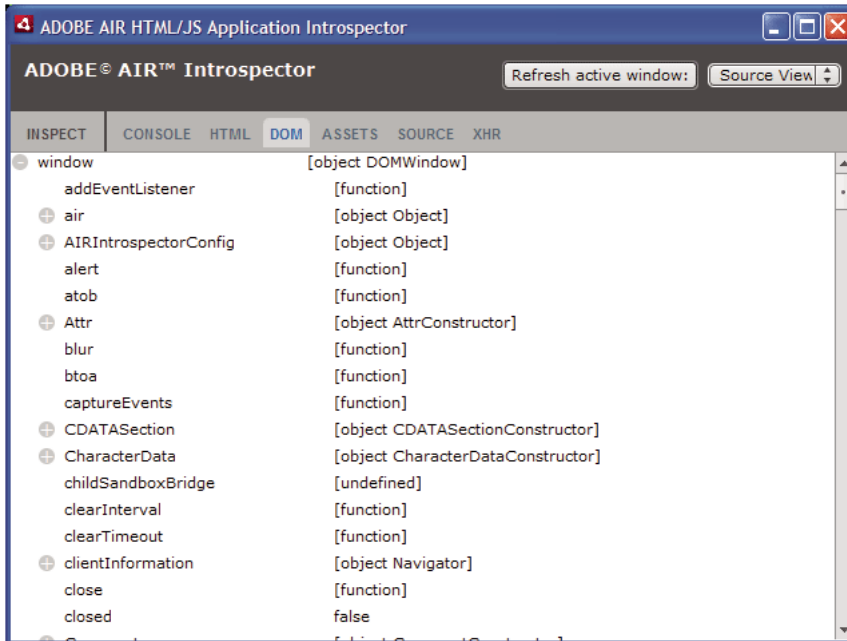
Нажмите кнопку Inspect (Проверить) (слева от списка вкладок в окне AIR Introspector). Можно щелкнуть любой элемент на странице HTML главного окна и связанный объект DOM отобразится на вкладке HTML. Если для главного окна включен фокус, можно также нажать клавишу быстрого вызова, чтобы включить или отключить кнопку Inspect. По умолчанию для этого используется клавиша быстрого вызова F11. Можно настроить другую клавишу быстрого вызова вместо клавиши F11. См. раздел «[Настройка AIR Introspector](#)» на странице 307».

Нажмите кнопку Refresh Active Window (Обновить активное окно) в верхней части окна AIR Introspector, чтобы обновить данные, отображаемые на вкладке HTML.

Нажмите CTRL+F в Windows или Command+F в Mac OS, чтобы выполнить поиск соответствующего критерию текста, отображаемого на вкладке. (Узлы дерева, которые не отображаются, не доступны для поиска.)

Вкладка DOM

На вкладке DOM показывается объект окна в древовидной структуре. Можно изменить любые строчные или численные свойства, отредактированные значения отображаются в приложении.

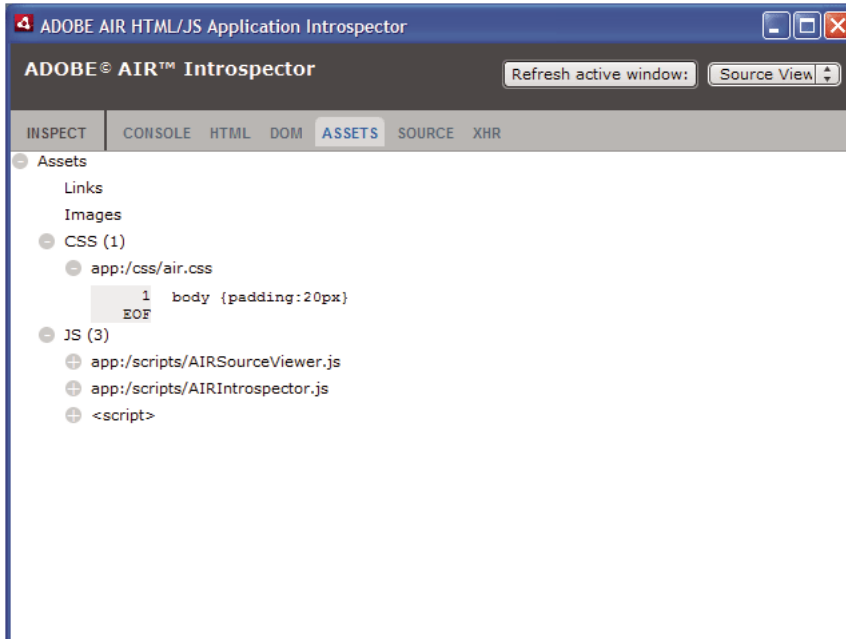


Нажмите кнопку Refresh Active Window (Обновить активное окно) в верхней части окна AIR Introspector, чтобы обновить данные, отображаемые на вкладке DOM.

Нажмите CTRL+F в Windows или Command+F в Mac OS, чтобы выполнить поиск соответствующего критерию текста, отображаемого на вкладке. (Узлы дерева, которые не отображаются, не доступны для поиска.)

Вкладка Assets (Активы)

На вкладке Assets (Активы) можно проверить ссылки, изображения, CSS и файлы JavaScript, загруженные в исходном окне. При разворачивании одного из этих узлов показывается содержимое файла или отображается фактически используемое изображение.



Нажмите кнопку Refresh Active Window (Обновить активное окно) в верхней части окна AIR Introspector, чтобы обновить данные, отображаемые на вкладке Assets (Активы).

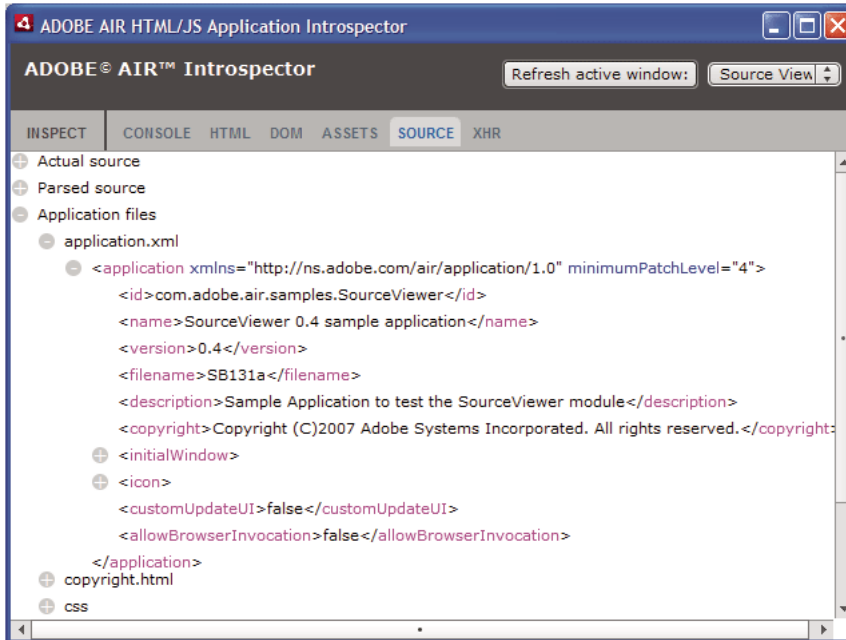
Нажмите CTRL+F в Windows или Command+F в Mac OS, чтобы выполнить поиск соответствующего критерию текста, отображаемого на вкладке. (Узлы дерева, которые не отображаются, не доступны для поиска.)

Вкладка Source (Источник)

Вкладка Source (Источник) содержит три раздела:

- Фактический источник: показывается исходный HTML-код страницы, загруженной в качестве основного содержимого при запуске приложения.
- Разобраный источник: показывается текущая разметка, образующая интерфейс пользователя приложения, которая может отличаться от фактического источника, поскольку в ходе работы приложение создает код разметки, используя технологии Ajax.

- **Файлы приложения:** отображается список файлов в каталоге приложения. Этот список доступен только для AIR Introspector, если он запущен из содержимого в изолированной программной среде приложения. В этом разделе можно просматривать содержимое текстовых файлов или изображения.

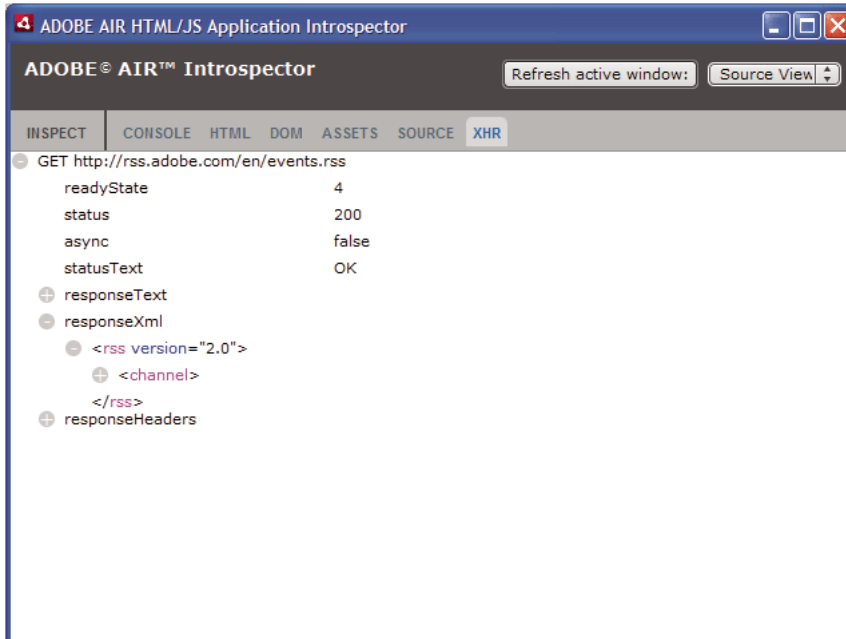


Нажмите кнопку Refresh Active Window (Обновить активное окно) в верхней части окна AIR Introspector, чтобы обновить данные, отображаемые на вкладке Source (Источник).

Нажмите CTRL+F в Windows или Command+F в Mac OS, чтобы выполнить поиск соответствующего критерию текста, отображаемого на вкладке. (Узлы дерева, которые не отображаются, не доступны для поиска.)

Вкладка XHR

Вкладка XHR перехватывает все обращения к XMLHttpRequest в приложении и протоколирует эту информацию. Это позволяет просматривать в древовидном представлении свойства XMLHttpRequest, включая `responseText` и `responseXML` (если они доступны).



Нажмите CTRL+F в Windows или Command+F в Mac OS, чтобы выполнить поиск соответствующего критерию текста, отображаемого на вкладке. (Узлы дерева, которые не отображаются, не доступны для поиска.)

Использование AIR Introspector с содержимым, находящимся вне изолированной программной среды приложения

Можно загрузить содержимое из каталога приложения в объект `iframe` или `frame`, который отображается на изолированную программную среду, не связанную с приложением. (См. раздел «Система защиты HTML в Adobe AIR» для разработчиков ActionScript или «Система безопасности HTML в Adobe AIR» для разработчиков HTML). Можно использовать AIR Introspector с таким содержимым, но соблюдая следующие правила:

- Файл `AIRIntrospector.js` должен быть включен в содержимое, находящееся как в изолированной программной среде приложения, так и вне ее (`iframe`).
- Не переопределяйте свойство `parentSandboxBridge`; оно используется кодом AIR Introspector. Добавьте необходимые свойства. Вместо того, чтобы писать следующий код:

```
parentSandboxBridge = mytrace: function(str) {runtime.trace(str)} ;
```

Используйте следующий синтаксис:

```
parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
```

- Из содержимого, находящегося вне изолированной программной среды приложения, нельзя открыть AIR Introspector нажатием клавиши F12 или вызовом одного из методов класса `air.Introspector.Console`. Можно открыть окно Introspector только нажатием кнопки Open Introspector (Открыть Introspector). Эта кнопка по умолчанию добавляется в правый верхний угол объекта `iframe` или `frame`. (Из-за ограничений безопасности, связанных с использованием содержимого вне изолированной программной среды приложения, новое окно может быть открыто только в результате действия пользователя, например нажатия кнопки.)
- Можно открыть отдельные окна AIR Introspector для содержимого, находящегося в изолированной программной среде приложения и вне его. Различать эти окна AIR Introspector можно по их заголовкам.
- На вкладке Source (Источник) файлы приложения не отображаются, если AIR Introspector запущен из содержимого, находящегося вне изолированной программной среды приложения
- AIR Introspector может просматривать только тот код в изолированной программной среде приложения, из которого он был запущен.

Глава 20. Локализация приложений AIR

Adobe AIR 1.1 и более новых версий

Adobe® AIR® предлагает поддержку нескольких языков.

Обзор локализованного содержимого ActionScript 3.0 и инфраструктуры Flex см. в разделе «Локализация приложений» в руководстве разработчика ActionScript 3.0.

Языки, поддерживаемые AIR

Поддержка локализации приложений AIR для следующих языков впервые появилась в версии AIR 1.1:

- Китайский (упрощенный)
- Китайский (традиционный)
- Французский
- Немецкий
- Итальянский
- Японский
- Корейский
- Португальский (Бразилия)
- Русский
- Испанский

В версии AIR 1.5 были добавлены следующие языки:

- Чешский
- Голландский
- Польский
- Шведский
- Турецкий

Дополнительные разделы справки

[Создание многоязычных приложений Flex в Adobe AIR](#)

[Создание многоязычных HTML-приложений](#)

Локализация названия и описания приложения в программе установки приложения AIR

Adobe AIR 1.1 и более новых версий

Для элементов `name` и `description` в файле дескриптора приложения можно задать несколько языков. Например, ниже показано как задать название на трех языках (английском, французском и немецком):

```
<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>
```

Атрибут `xml:lang` для каждого текстового элемента задает код языка согласно документу RFC4646 (<http://www.ietf.org/rfc/rfc4646.txt>).

Элемент `name` задает название приложения, которое отображается в программе установки AIR. Программа установки приложения AIR использует локализованное значение, лучше всего подходящее к языку пользовательского интерфейса, определенному параметрами операционной системы.

Таким же образом можно задать несколько языков для элемента `description` в файле дескриптора приложения. Этот элемент задает текст описания, который выводится в программе установки приложения.

Эти параметры относятся только к языкам, доступным в программе установки AIR. Они не определяют локали, доступные для установленного и работающего приложения. Приложения AIR могут иметь интерфейсы на многих языках, помимо тех, которые доступны в программе установки AIR.

Дополнительные сведения см. в разделе «[Элементы дескриптора приложения AIR](#)» на странице 224».

Дополнительные разделы справки

[Создание многоязычных приложений Flex в Adobe AIR](#)

[Создание многоязычных HTML-приложений](#)

Локализация HTML-содержимого с помощью инфраструктуры локализации AIR HTML

Adobe AIR 1.1 и более новых версий

В комплект AIR 1.1 SDK включена инфраструктура локализации HTML. Она определяется файлом `AIRLocalizer.js` JavaScript. Файл `AIRLocalizer.js` находится в каталоге `frameworks` комплекта AIR SDK. В нем задан класс `air.Localizer`, обеспечивающий поддержку многоязычных приложений.

Загрузка кода HTML-инфраструктуры локализации AIR

Чтобы воспользоваться многоязычной функциональностью, скопируйте файл `AIRLocalizer.js` в своей проект. Затем включите его в основной HTML-файл приложения, заключив в `ter script`:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

Код JavaScript может вызвать объект `air.Localizer.localizer`:

```
<script>
  var localizer = air.Localizer.localizer;
</script>
```

`air.Localizer.localizer` — это единый объект, определяющий методы и свойства использования локализованных ресурсов. Класс `Localizer` содержит ряд методов DRM:

Метод	Описание
<code>getFile()</code>	Получает текст определенного пакета ресурсов для заданной локали. См. раздел « Получение ресурсов для определенной локали » на странице 324».
<code>getLocaleChain()</code>	Возвращает языки в цепочке локалей. См. раздел « Определение цепочки локалей » на странице 323».
<code>getResourceBundle()</code>	Возвращает ключи пакета и соответствующие значения в виде объекта. См. раздел « Получение ресурсов для определенной локали » на странице 324».
<code>getString()</code>	Получает строку, определенную для данного ресурса. См. раздел « Получение ресурсов для определенной локали » на странице 324».
<code>setBundlesDirectory()</code>	Задаёт расположение каталога пакетов. См. раздел « Изменение параметров Localizer в AIR HTML » на странице 322».
<code>setLocalAttributePrefix()</code>	Задаёт префикс, используемый атрибутами localizer в HTML-элементах DOM. См. раздел « Изменение параметров Localizer в AIR HTML » на странице 322».
<code>setLocaleChain()</code>	Задаёт порядок языков в цепочке локалей. См. раздел « Определение цепочки локалей » на странице 323».
<code>sortLanguagesByPreference()</code>	Располагает локали в цепочке согласно порядку локалей в параметрах операционной системы. См. раздел « Определение цепочки локалей » на странице 323».
<code>update()</code>	Обновляет модель DOM HTML (или элемент DOM), используя локализованные строки из текущей цепочки локалей. Описание цепочек локалей см. в разделе « Управление цепочками локалей » на странице 320». Дополнительные сведения о методе <code>update()</code> см. в разделе « Обновление элементов DOM для использования текущей локали » на странице 321».

Класс Localizer содержит ряд статических свойств.

Свойство	Описание
<code>localizer</code>	Возвращает ссылку на единственный в приложении объект Localizer.
<code>ultimateFallbackLocale</code>	Локаль, используемая, когда приложение не поддерживает выбор пользователя. См. раздел « Определение цепочки локалей » на странице 323».

Указание поддерживаемых языков

Для идентификации языков, поддерживаемых приложением, в файле дескриптора приложения используется элемент `<supportedLanguages>`. Этот элемент можно применять только для iOS, связанной среды выполнения Mac и приложений Android. Он игнорируется в приложениях любых других типов.

Если элемент `<supportedLanguages>` не указан, по умолчанию упаковщик выполняет следующие действия в зависимости от типа приложения:

- iOS — все языки, поддерживаемые средой выполнения AIR, представлены в магазине приложений iOS как поддерживаемые языки приложения.
- Связанная среда Mac — приложения, упакованные в пакет связанной среды, не содержат информации о языке.
- Android — комплект приложений содержит ресурсы для всех языков, поддерживаемых исполнительной средой AIR.

Дополнительные сведения см. в описании элемента «[supportedLanguages](#)» на странице 256.

Определение пакетов ресурсов

Инфраструктура локализации HTML считывает локализованные строки из файлов *локализации*. Файл локализации — это набор значений, привязанных к ключам и последовательно собранных в текстовый файл. Файл локализации иногда называют *пакетом*.

Создайте в каталоге приложения подкаталог с названием locale. (Можно также использовать другое имя, см. раздел «[Изменение параметров Localizer в AIR HTML](#)» на странице 322.) Здесь будут храниться файлы локализации. Этот каталог известен как *каталог пакетов*.

Для каждой локали, которая поддерживается приложением, создайте подкаталог каталога пакетов. Назовите каждый подкаталог в соответствии с кодом локали. Например, для французского каталога используйте имя «fr», а для английского — «en». С помощью знака подчеркивания () можно определить локаль, содержащую язык и код страны. Например, каталог для английского (США) может называться «en_us». (Также можно использовать дефис, в этом случае каталог будет называться «en-us». Инфраструктура локализации распознает оба варианта.)

В подкаталог locale можно добавить сколь угодно много файлов ресурсов. Как правило, файл локализации создается для каждого языка (и помещается в каталог соответствующего языка). Инфраструктура локализации HTML содержит метод `getFile()`, позволяющий считывать содержимое файла (см. раздел «[Получение ресурсов для определенной локали](#)» на странице 324).

Файлы с расширением `.properties` называются файлами свойств локализации. Они используются для определения пар «ключ-значение» той или иной локали. Файл свойств задает строковое значение на каждой строке. Например, ниже показано, как задать строковое значение "Hello in English." для ключа с именем `greeting`:

```
greeting=Hello in English.
```

В файле свойств содержится следующий текст и задаются шесть пар «ключ-значение»:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Этом примере показана английская версия файла свойств из каталога `en`.

Французская версия этого файла будет помещена в каталог `fr`:

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Для разных типов информации можно задать разные файлы ресурсов. Скажем, в файле `legal.properties` может храниться какой-нибудь шаблонный юридический документ (к примеру, информация об авторских правах). Эти ресурсы можно повторно использовать в различных приложениях. Точно так же можно задать файлы для определения локализованного содержимого разных частей пользовательского интерфейса.

В этих файлах следует использовать кодировку UTF-8, чтобы языки отображались корректно.

Управление цепочками локалей

Когда приложение загружает файл AIRLocalizer.js, оно исследует доступные локали. Эти локали соответствуют подкаталогам каталога пакетов (см. раздел «[Определение пакетов ресурсов](#)» на странице 319»). Список поддерживаемых локалей называется *цепочкой локалей*. Файл AIRLocalizer.js автоматически определяет порядок локалей в цепочке на основании соответствующих параметров операционной системы. (Свойство `Capabilities.languages` содержит список языков пользовательского интерфейса операционной системы, в порядке предпочтения.)

Поэтому, если в приложении определены ресурсы для локалей «en», «en_US» и «en_UK», инфраструктура Localizer AIR HTML упорядочивает цепочку локалей соответственно. Если приложение запускается в системе, где основной локалью является «en», цепочка локалей будет иметь порядок ["en", "en_US", "en_UK"]. В этом случае приложение сначала будет искать ресурсы в пакете «en», потом в «en_US».

Однако, если основной локалью в системе является «en-US», то порядок будет таков: ["en_US", "en", "en_UK"]. В этом случае приложение сперва обратится к пакету «en_US», а потом к «en».

По умолчанию, первая локаль в цепочке считается стандартом по умолчанию. Можно попросить пользователя выбрать локаль при первом запуске приложения. Затем можно сохранить этот выбор в файле настроек и при последующих запусках сразу включать эту локаль.

Могут использоваться строки ресурсов любой локали из цепочки. Если в какой-то из локалей строка ресурсов не задана, приложение обратится к следующей подходящей строке из другой локали.

Для изменения настроек цепочки локалей вызовите метод `setLocaleChain()` объекта Localizer. См. раздел «[Определение цепочки локалей](#)» на странице 323».

Обновление элементов DOM с использованием локализованного содержимого

Элемент приложения может обращаться к ключу в файле свойств локализации. Например, элемент `title` в примере ниже задает атрибут `local_innerHTML`. Инфраструктура локализации использует этот атрибут для нахождения локализованного значения. По умолчанию, инфраструктура ищет атрибуты, имена которых начинаются с "local_". Она обновляет атрибуты с совпадающей частью имени после "local_". В этом случае инфраструктура задает атрибут `innerHTML` элемента `title`. Атрибут `innerHTML` использует значение, заданное для ключа `mainWindowTitle` в файле свойств по умолчанию (`default.properties`):

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Если в текущей локали нет подходящего значения, тогда инфраструктура локализации просматривает остальные локали цепочки. Используется ближайшая локаль в цепочке, в которой значение определено.

В примере ниже в тексте (атрибут `innerHTML attribute`) элемента `p` используется значение ключа `greeting`, заданное в файле свойств по умолчанию:

```
<p local_innerHTML="default.greeting" />
```

В примере ниже в атрибуте `value` (и отображаемом тексте) элемента `input` используется значение ключа `btnBlue` из файла свойств по умолчанию:

```
<input type="button" local_value="default.btnBlue" />
```

Чтобы обновить модель DOM HTML и использовать строки, заданные в текущей цепочке ключей, вызовите метод `update()` объекта Localizer. Вызов метода `update()` позволяет объекту Localizer разобрать DOM и выполнить необходимые действия над найденными атрибутами локализации ("local_..."):

```
air.Localizer.localizer.update();
```

Можно задать значения для атрибута (например, «innerHTML») и соответствующего ему атрибута локализации (например, «local_innerHTML»). В этом случае инфраструктура локализации переписывает значение атрибута, только если найдет совпадающее значение в цепочке. Например, следующий элемент задает атрибуты `value` и `local_value`:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

Можно обновить и отдельный элемент DOM. См. следующий раздел «[Обновление элементов DOM для использования текущей локали](#)» на странице 321».

По умолчанию Localizer AIR HTML использует `"local_"` в качестве префикса для атрибутов, определяющий параметры локализации элемента. Например, по умолчанию атрибут `local_innerHTML` определяет имя пакета и ресурса для значения `innerHTML` элемента. Кроме того, по умолчанию атрибут `local_value` определяет имя пакета и ресурса для значения `value` элемента. Localizer можно изменить, чтобы использовать другой префикс атрибута (не `"local_"`). См. раздел «[Изменение параметров Localizer в AIR HTML](#)» на странице 322».

Обновление элементов DOM для использования текущей локали

Когда объект Localizer обновляет модель DOM HTML, помеченные элементы вынуждены использовать значения атрибутов, основанные на строках, которые заданы в текущей цепочке локалей. Чтобы Localizer HTML обновил DOM HTML, вызовите метод `update()` объекта Localizer:

```
air.Localizer.localizer.update();
```

Чтобы обновить только отдельный элемент DOM, передайте его в качестве параметра методу `update()`. Метод `update()` имеет только один параметр, `parentNode`, и тот не обязателен. Если параметр `parentNode` задан, он определяет локализуемый элемент DOM. При вызове метода `update()` и указании параметра `parentNode` задаются локализованные значения для всех дочерних элементов, которые указывают атрибуты локализации.

Рассмотрим элемент `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>
```

Чтобы обновить этот элемент и использовать локализованные строки, заданные в цепочке локалей, воспользуйтесь следующим кодом JavaScript:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Если значение ключа не найдено в цепочке локалей, инфраструктура использует значение атрибута `"local_"`. Положим, в предыдущем примере инфраструктуре не удалось найти значение для ключа `lblColors` (ни в одном из файлов `default.properties` в цепочке локалей). Тогда в качестве значения `innerHTML` будет использоваться `"default.lblColors"`. Использование этого значения сообщает разработчику, что каких-то ресурсов не хватает.

Метод `update()` отправляет событие `resourceNotFound`, когда не может найти ресурс в цепочке локалей. Константа `air.Localizer.RESOURCE_NOT_FOUND` задает строку `"resourceNotFound"`. У этого события есть три свойства: `bundleName`, `resourceName` и `locale`. Свойство `bundleName` — это имя пакета, в котором не удалось обнаружить ресурс. Свойство `resourceName` — это имя ресурса, который не удалось обнаружить. Свойство `locale` — это имя локали, в которой не удалось обнаружить ресурс.

Метод `update()` отправляет событие `bundleNotFound`, когда не может найти указанный пакет. Константа `air.Localizer.BUNDLE_NOT_FOUND` задает строку `"bundleNotFound"`. У события есть два свойства: `bundleName` и `locale`. Свойство `bundleName` — это имя пакета, в котором не удалось обнаружить ресурс. Свойство `locale` — это имя локали, в которой не удалось обнаружить ресурс.

Метод `update()` работает асинхронно (и асинхронно отправляет события `resourceNotFound` и `bundleNotFound`). Следующий код задает прослушатели для событий `resourceNotFound` и `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);
}
```

Изменение параметров Localizer в AIR HTML

Метод `setBundlesDirectory()` объекта `Localizer` позволят изменить путь к каталогу пакетов. Метод `setLocalAttributePrefix()` объекта `Localizer` позволяет изменить путь к каталогу пакетов и значения атрибутов, используемых объектом `Localizer`.

Каталог пакетов по умолчанию определен как подкаталог `locale` в каталоге приложения. Можно задать другой каталог, вызвав метод `setBundlesDirectory()` объекта `Localizer`. Этот метод принимает один параметр, `path`, задающий в виде строки путь к нужному каталогу пакетов. Значением параметра `path` может быть одно из следующих:

- объект `String`, определяющий относительный путь к каталогу, например `"locales"`
- объект `String`, задающий действительный URL-адрес, использующий схемы URL-адресов `app`, `app-storage` или `file`, например `"app://languages"` (не используйте схему `http`)
- объект `File`

Сведения об URL-адресах и путях к каталогам см. в следующих источниках:

- [Пути объектов File](#) (для разработчиков ActionScript)
- [Пути объектов File](#) (для разработчиков HTML)

Например, в коде ниже в качестве каталога пакетов задается подкаталог `languages` каталога хранилища приложения (не каталога приложения):

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

В качестве параметра `path` должен передаваться действительный путь. В противном случае метод генерирует исключение `BundlePathNotFoundError`. `"BundlePathNotFoundError"` является значением свойства `name` этой ошибки, а свойство `message` указывает недействительный путь.

По умолчанию `Localizer AIR HTML` использует `"local_"` в качестве префикса для атрибутов, определяющий параметры локализации элемента. Например, атрибут `local_innerHTML` определяет имена пакета и ресурса в значении `innerHTML` следующего элемента `input`:

```
<p local_innerHTML="default.greeting" />
```

Метод `setLocalAttributePrefix()` объекта `Localizer` позволяет использовать префикс атрибута, отличный от `"local_"`. Статический метод принимает один параметр — строку, которая используется в качестве префикса атрибута. Например, в коде ниже инфраструктура локализации использует в качестве префикса атрибута `«loc_»`:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Этот префикс можно изменять. Например, это может потребоваться, если префикс по умолчанию (`"local_"`) приводит к конфликту с именем другого атрибута в коде. При вызове этого метода используйте допустимые символы для HTML-атрибутов. (Например, знак пробела использовать нельзя.)

Дополнительные сведения об использовании атрибутов локализации в HTML-элементах см. в разделе [«Обновление элементов DOM с использованием локализованного содержимого»](#) на странице 320.

Параметры каталога пакетов и префикса атрибута не сохраняются между разными сеансами приложения. Если вы используете собственный каталог пакетов или префикс атрибута, его потребуется задавать каждый раз при запуске приложения.

Определение цепочки локалей

По умолчанию, когда загружается файл `AIRLocalizer.js`, он задает цепочку локалей. Эту цепочку определяют локали, доступные в каталоге пакетов и параметры языка пользовательского интерфейса системы. (Дополнительные сведения см. в разделе [«Управление цепочками локалей»](#) на странице 320.)

Для изменения настроек цепочки локалей вызовите статический метод `setLocaleChain()` объекта `Localizer`. Например, этот метод можно вызвать, если пользователь указал, какой язык предпочитает. Метод `setLocaleChain()` принимает один параметр, `chain`, представляющий собой массив локалей, например `["fr_FR", "fr", "fr_CA"]`. Инфраструктура ищет ресурсы (в последовательности операций) в том порядке, в каком идут локали в массиве. Если ресурс не удастся найти в первой локали, то поиск продолжается в ресурсах других локалей. Если аргумент `chain` отсутствует, не является массивом или является пустым массивом, то функция дает сбой и генерирует исключение `IllegalArgumentsError`.

Статический метод `getLocaleChain()` объекта `Localizer` возвращает объект `Array` со списком локалей в текущей цепочке.

Приведенный ниже код читает текущую цепочку локалей и добавляет в начало две французских локали:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

При обновлении цепочки локалей метод `setLocaleChain()` отправляет событие `"change"`. Константа `air.Localizer.LOCALE_CHANGE` определяет строку `"change"`. Событие имеет единственное свойство, `localeChain`, представляющее собой массив кодов локалей в новой цепочке. В коде ниже задается прослушиватель для этого события:

```
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);
}
```


Статическое свойство `air.Localizer.ultimateFallbackLocale` представляет локаль, которая используется, если приложение не поддерживает пользовательские установки. Значение по умолчанию равно `"en"`. Можно задать и другую локаль, как показано ниже:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Получение ресурсов для определенной локали

Метод `getString()` объекта `Localizer` возвращает строку, определенную для ресурса в той или иной локали. При вызове этого метода значение `locale` задавать необязательно. В этом случае метод просматривает всю цепочку локалей и возвращает строку в первой же локали, которая содержит искомое имя ресурса. Ниже перечислены параметры этого метода:

Параметр	Описание
<code>bundleName</code>	Пакет, содержащий ресурс. Это имя файла свойств без расширения <code>.properties</code> . (Например, если этот параметр равен <code>"alerts"</code> , код <code>Localizer</code> просматривает файлы локализации <code>alerts.properties</code> .)
<code>resourceName</code>	Имя ресурса.
<code>templateArgs</code>	Необязательно. Массив строк для замены пронумерованных тегов в строке замещения. Например, рассмотрим вызов функции, где параметр <code>templateArgs</code> равен <code>["Raúl", "4"]</code> , а совпадающая строка ресурса — <code>"Hello, {0}. You have {1} new messages."</code> ("Здравствуйте, {0}. У вас {1} новых сообщения."). В этом случае функция возвратит запись <code>"Hello, Raúl. You have 4 new messages."</code> ("Здравствуйте, Рауль. У вас 4 новых сообщения"). Чтобы игнорировать это значение, передайте <code>null</code> .
<code>locale</code>	Необязательно. Код локали (например, <code>"en"</code> , <code>"en_us"</code> или <code>"fr"</code>). Если локаль задана, а подходящих значений не обнаружено, метод не будет искать значения в остальных локалях цепочки. Если код локали не задан, функция возвращает строку в первой локали из цепочки, в которой обнаруживается нужное значение имени ресурса.

Инфраструктура локализации может обновлять отмеченные атрибуты DOM HTML. Однако локализованные строки могут использоваться иными способами. Например, можно использовать строку в каком-нибудь динамически создаваемом HTML-файле или в качестве значения параметра при вызове функции. В приведенном ниже коде в вызове функции `alert()` участвует строка, определенная в ресурсе `error114` в файле свойств по умолчанию для локали `fr_FR`:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

Когда метод `getString()` не может найти ресурс в определенном пакете, он отправляет событие `resourceNotFound`. Константа `air.Localizer.RESOURCE_NOT_FOUND` задает строку `"resourceNotFound"`. У этого события есть три свойства: `bundleName`, `resourceName` и `locale`. Свойство `bundleName` — это имя пакета, в котором не удалось обнаружить ресурс. Свойство `resourceName` — это имя ресурса, который не удалось обнаружить. Свойство `locale` — это имя локали, в которой не удалось обнаружить ресурс.

Метод `getString()` отправляет событие `bundleNotFound`, когда не может найти указанный пакет. Константа `air.Localizer.BUNDLE_NOT_FOUND` задает строку `"bundleNotFound"`. У события есть два свойства: `bundleName` и `locale`. Свойство `bundleName` — это имя пакета, в котором не удалось обнаружить ресурс. Свойство `locale` — это имя локали, в которой не удалось обнаружить ресурс.

Метод `getString()` работает асинхронно (и асинхронно отправляет события `resourceNotFound` и `bundleNotFound`). Следующий код задает прослушиватели для событий `resourceNotFound` и `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);
}
```

Метод `getResourceBundle()` объекта `Localizer` возвращает указанный пакет для заданного языка (параметр `locale`). Возвращаемое методом значение является объектом со свойствами, соответствующими ключам в пакете. (Если приложение не может найти указанный пакет, метод возвращает значение `null`.)

Метод использует два параметра: `locale` и `bundleName`.

Параметр	Описание
<code>locale</code>	Локаль (например, "fr").
<code>bundleName</code>	Имя пакета.

Например, в следующем коде вызывается метод `document.write()` для загрузки пакета по умолчанию для локали (fr). Затем вызывается метод `document.write()` для записи значений ключей `str1` и `str2` в этом пакете:

```
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/>");
```

Метод `getResourceBundle()` отправляет событие `bundleNotFound`, когда не может найти указанный пакет. Константа `air.Localizer.BUNDLE_NOT_FOUND` задает строку "bundleNotFound". У события есть два свойства: `bundleName` и `locale`. Свойство `bundleName` — это имя пакета, в котором не удалось обнаружить ресурс. Свойство `locale` — это имя локали, в которой не удалось обнаружить ресурс.

Метод `getFile()` объекта `Localizer` возвращает содержимое пакета для данной локали в виде строки. Файл пакета считывается как файл UTF-8. У этого метода есть ряд параметров:

Параметр	Описание
resourceFileName	Имя файла ресурса (например, "about.html").
templateArgs	Необязательно. Массив строк для замены пронумерованных тегов в строке замещения. Например, рассмотрим вызов функции, где параметр <code>templateArgs</code> равен ["Raúl", "4"], а совпадающая строка ресурса содержит две строки: <pre><html> <body>Hello, {0}. You have {1} new messages.</body> </html></pre> В этом случае функция возвратит строку с двумя строками: <pre><html> <body>Hello, Raúl. You have 4 new messages. </body> </html></pre>
locale	Используемый код локали, например "en_GB". Если локаль задана, а подходящих файлов не обнаружено, метод не будет продолжать поиск в остальных локалях цепочки. Если код локали <i>не</i> задан, функция возвращает текст в первой же локали цепочки, в котором удается найти файл, соответствующий <code>resourceFileName</code> .

Например, в коде ниже метод `document.write()` вызывается с помощью содержимого файла `about.html` локали `fr`:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

Метод `getFile()` отправляет событие `fileNotFound`, когда не может найти ресурс в цепочке локалей. Константа `air.Localizer.FILE_NOT_FOUND` определяет строку `resourceNotFound`. Метод `getFile()` работает асинхронно (и асинхронно отправляет событие `fileNotFound`). У события есть два свойства: `fileName` и `locale`. Свойство `fileName` — это имя файла, который не удалось найти. Свойство `locale` — это имя локали, в которой не удалось обнаружить ресурс. В коде ниже задается прослушиватель для этого события:

```
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);
}
```

Дополнительные разделы справки

[Создание многоязычных HTML-приложений](#)

Глава 21. Переменные среды Path

Пакет AIR SDK содержит несколько программ, которые можно запускать из командной строки или окна терминала. Запускать эти программы будет намного удобнее, если определить путь к каталогу bin SDK с помощью переменной среды Path.

В данном разделе представлено общее описание процедуры настройки пути в Windows, Mac и Linux. Однако конфигурации компьютеров могут существенно отличаться, поэтому данная процедура работает не во всех системах. В этом случае нужную информацию можно найти в документации по операционной системе или в Интернете.

Настройка переменной среды PATH в Linux и Mac OS с использованием оболочки Bash

Когда команда вводится в окно терминала, оболочка (программа, которая считывает вводимые команды и пытается отреагировать соответствующим образом) сначала должна найти программу команды в файловой системе. Оболочка выполняет поиск команд в списке каталогов, который хранится в переменной среды с именем \$PATH. Чтобы посмотреть текущее значение переменной path, введите следующую команду:

```
echo $PATH
```

Будет выведен список разделенных двоеточием каталогов примерно в таком виде:

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

Путь к каталогу bin пакета AIR SDK необходимо добавить в этот список, чтобы оболочка могла находить ADT и инструменты ADT. Предположим, что AIR SDK находится в каталоге /Users/fred/SDKs/AIR. В этом случае для добавления нужных каталогов в переменную среды path используется следующая команда:

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Примечание. Если путь содержит символы пробелов, перед ними следует вставить символ обратной косой черты, как показано ниже:

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

Чтобы проверить результаты добавления, можно еще раз выполнить команду echo:

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Каталоги добавлены успешно. Теперь вы сможете вызывать следующие команды и получать корректный ответ:

```
adt -version
```

Если переменная среды \$PATH была изменена правильно, в результате выполнения команды должна быть выдана версия среды ADT.

Однако имеется одна проблема: при открытии нового окна терминала вы заметите, что переменная path больше не содержит новые элементы, которые ранее были в нее добавлены. Команду настройки пути необходимо выполнять каждый раз при запуске нового терминала.

Общим решением этой проблемы будет добавление команды в один из сценариев запуска, которые используются оболочкой. В ОС Mac OS можно создать файл `.bash_profile` в каталоге `~/username`, который будет запускаться каждый раз при открытии нового окна терминала. В Ubuntu сценарием, который запускается при открытии нового окна терминала, является `.bashrc`. В других дистрибутивах Linux и оболочках применяются аналогичные правила.

Процедура добавления команды в сценарий запуска оболочки

- 1 Измените домашний каталог:

```
cd
```

- 2 Создайте профиль конфигурации оболочки (при необходимости) и перенаправьте вводимый текст в конец файла с помощью команды «`cat >>`». Используйте соответствующий файл для своей операционной системы и оболочки. Например, в ОС Mac OS можно использовать `.bash_profile`, а в Ubuntu — `.bashrc`.

```
cat >> .bash_profile
```

- 3 Введите текст, который требуется добавить в файл:

```
export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
```

- 4 Завершите перенаправление текста, нажав клавиши `CTRL-SHIFT-D` на клавиатуре.

- 5 Отобразите файл, чтобы убедиться в правильности выполнения операции:

```
cat .bash_profile
```

- 6 Откройте новое окно терминала, чтобы проверить путь:

```
echo $PATH
```

Добавленные элементы должны быть выведены на экране.

Если позднее будет создана новая версия одного из пакетов SDK, которая будет помещена в другой каталог, команду настройки пути в файле конфигурации необходимо будет обновить соответствующим образом. В противном случае оболочка продолжит использование старой версии.

Настройка переменной среды PATH в ОС Windows

При открытии командное окно в Windows наследует глобальные переменные среды, определенные в свойствах системы. Одной из важных переменных является переменная `path`, содержащая список каталогов, в которых выполняется поиск введенной программы. Чтобы посмотреть текущий список каталогов в переменной `path`, в командном окне можно ввести следующую команду:

```
set path
```

Будет выведен список разделенных двоеточием каталогов, который выглядит примерно следующим образом:

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

Путь к каталогу `bin` пакета AIR SDK необходимо добавить в этот список, чтобы программа команд могла находить ADT и инструменты ADT. Предположим, что пакет AIR SDK находится в каталоге `C:\SDKs\AIR`. В этом случае путь можно добавить следующим образом:

- 1 В разделе «Панель управления» откройте диалоговое окно «Свойства системы» или нажмите правой кнопкой мыши на значке «Мой компьютер» и в меню выберите пункт «Свойства».
- 2 На вкладке «Дополнительно» нажмите кнопку «Переменные среды».
- 3 В разделе «Системные переменные» диалогового окна «Переменные среды» выберите элемент «Path».

- 4 Нажмите кнопку «Изменить».
- 5 Перейдите в конец строки в поле «Значение переменной».
- 6 В самом конце текущего значения введите следующий текст:

```
;C:\SDKs\AIR\bin
```

- 7 Нажмите кнопку «ОК» во всех диалоговых окнах, чтобы сохранить путь.

Если в этот момент имеются открытые командные окна, изменения для них применены не будут. Откройте новое командное окно и введите следующую команду, чтобы убедиться в правильности настройки переменной среды:

```
adt -version
```

Если позднее расположение пакета AIR SDK будет изменено или будет добавлена новая версия, переменную среды path необходимо будет изменить соответствующим образом.