Getting Started with ADOBE® AIR® for TV

Getting Started with Adobe® AIR® for TV

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Last updated 9/29/2011

Contents

Chapter 1: Introducing Adobe AIR 3 for TV

This guide introduces C++ and Linux® developers to the Adobe® AIR® 3 for TV platform development kit. It highlights AIR for TV features. It also describes how to install and build the source distribution in a Linux environment. Finally, it discusses how to run an AIR application on AIR for TV.

Developer documentation

The following types of developers use AIR for TV documentation:

System developers These developers customize AIR for TV for their platform. They configure how to launch AIR for TV. They are experienced C++/Linux developers. These developers do these customizations for silicon vendors or original equipment manufacturers (OEMs).

Platform driver developers These developers provide optimized modules for AIR for TV. These modules replace provided software implementations with implementations that use a platform's hardware for graphics, video, audio, images, and digital rights management. They are experienced C++/Linux developers. These developers provide these optimized modules for silicon vendors or OEMs.

AIR application developers These developers create AIR applications that AIR for TV runs. Various ways are available to install these applications on a TV device. For example, the OEM can install the applications. Alternatively, an application store can install applications.

Native extension developers These developers create native extensions for Adobe AIR. The extensions allow AIR application developers to use a device's native features or code. These developers, possibly working as a team of developers, have both C/C++/Linux development experience and Flash/ActionScript development experience. They typically work closely with an OEM, because the OEM installs the extension's C/C++ libraries on the device.

The following documentation describes the different AIR for TV documents:

Document	Audience	Purpose
Getting Started with Adobe AIR for TV (PDF)	System developers and platform driver developers	An introduction to AIR for TV features, and to installing, building, and testing AIR for TV on your platform
Optimizing and Integrating Adobe AIR for TV (PDF)	System developers and platform driver developers	• Describes how to develop platform-specific modules, including how to use the C++ APIs. • Describes how to integrate AIR for TV with your product, including C++ APIs and configuration tasks. • Describes the process for building AIR for TV with platform-specific modules
Porting ADOBE® FLASH® ACCESS™ to Adobe AIR for TV	System developers and platform driver developers	Describes how to implement a Flash Access adaptor for your platform. The Flash Access adaptor allows AIR for TV applications to access protected content using Flash Access 2.0 digital rights management.

Document	Audience	Purpose
Getting Started with the DEVICE CERTIFICATION TEST SUITE	System developers and platform driver developers	Describes how to use the Device Certification Suite (DCTS) to test and verify your implementation of AIR for TV. If you don't yet have DCTS credentials, the web page at *Getting Started with the DEVICE CERTIFICATION TEST SUITE* provides a link to the document. The document has information on getting your DCTS login credentials.
Developing Native Extensions for Adobe AIR	Native extension developers	Describes how to develop and deploy native extensions. for Adobe AIR.
Building Adobe AIR Applications	AIR application developers	Describes how to author AIR applications.
Optimizing Performance for the ADOBE® FLASH® PLATFORM	AIR application developers	Best practices for developing applications for mobile devices and digital home devices.
ActionScript 3.0 Reference for the Adobe Flash Platform	AIR application developers	Contains the ActionScript language elements, core libraries, and component packages and classes for the tools, runtimes, services, and servers in the Flash Platform.

About AIR for TV

Adobe® AIR® for TV is the Adobe® AIR® runtime optimized for hardware and software architectures of digital home electronics. These devices include, for example, television sets, Blu-ray players, game consoles, and set-top boxes.

Adobe® Flash® developers can create applications for AIR for TV that stream and play high-definition video from the Internet. These developers can also create games, social content, rich Internet applications, and graphical user interfaces that AIR for TV runs. AIR for TV runs these AIR applications on the target digital home device.

- **Delivery and playback of HD video from the Internet**

  AIR for TV enables streaming video content from the Internet directly to TV sets and other AIR for TV platforms, without the use of a web browser. AIR for TV supports 1080p high definition video.

  Video encoded with the H.264 codec is recommended, because AIR for TV can use a platform's hardware accelerators for improved performance decoding and presenting H.264 videos. The H.264 videos can use the AAC audio codec or multichannel audio codecs. AIR for TV also supports Sorenson H.263, and On2 VP6 codecs using software decoding.

  AIR for TV supports progressive downloading of a video from an HTTP server. It also supports HTTP streaming and RTMP streaming from an Adobe® Flash® Media Server, including dynamic streaming.

- **Games and social content**

  AIR for TV can run games and social content, taking advantage of the big screen and convenience of Internet-connected home electronics devices. AIR for TV supports ActionScript APIs that allow applications to communicate with attached game input devices, such as joysticks, gamepads, and wands.

- **Rich Internet applications (RIAs)**

  AIR for TV is a vehicle for providing RIAs for use on Internet-connected home electronics devices. AIR for TV enables the use of AIR applications as the interface to web services.

- **High-performance graphical user interfaces**

AIR for TV supports dynamic, feature-rich user experiences. AIR for TV uses a platform's graphical hardware accelerators to provide a high-performance 2D and 3D graphical user interface for a 10-foot user experience. Such user interfaces are designed for a large television screen with user input from a remote control device.

AIR for TV platform development

AIR for TV requires minimal effort to recompile on any Linux distribution that is based on version 2.6.x of the Linux kernel. AIR for TV can be recompiled for most target platforms in a single working day. This effort does not require extensive porting of C++ code or knowledge of AIR.

AIR for TV provides a streamlined C++ Application Programming Interface (API) that does not require knowledge of AIR details. You can develop and build an AIR for TV platform for a target Linux operating system entirely in a Linux environment.

The highly modular design of AIR for TV facilitates the creation of platform-specific modules. To use hardware capabilities that are unique to your target platform, you create a module. Sometimes your platform-specific module replaces a module provided by AIR for TV. In some cases, AIR for TV requires that you create a module to support a feature.

For example, AIR for TV requires that you create a module that accesses your platform's hardware accelerators for processing H.264 streaming video.

AIR for TV capabilities

AIR for TV is AIR, optimized for hardware and software architectures of digital home electronics. AIR applications that run on AIR for TV use the AIR application profiles `TV` or `extendedTV`.

AIR for TV also provides the following overall capabilities:

- AIR application content appears in an area of the screen known as the *Stage*.
- AIR for TV can run AIR applications installed on the local file system. The root SWF file (the starting SWF file) can load additional SWF content from the local filesystem or from http://URLs. You can also enable AIR for TV to allow SWF content to load additional SWF files from https://URLs.

  Note: *AIR for TV supports only SWF content in AIR applications. The applications cannot contain HTML, JavaScript, CSS, Ajax, or PDF content.*
- You can create your own platform-specific *driver* that enables your AIR for TV platform to interact with hardware-based functionality of your target platform. Typically, you create drivers to replace functionality that AIR for TV provides in software with a hardware-based equivalent. For example, to use hardware accelerators for decompressing and presenting video, you create a driver. The driver enables your AIR for TV platform to interact with specific video-processing hardware. Support of some features, such as H.264 support and digital rights management, require a platform-specific driver.
- AIR for TV supports H.264 by interacting with a platform-specific driver that uses the device hardware's H.264 support. Typically, the device provides full 1080p HD video.
- AIR for TV supports SWF-embedded video, and video content delivered from the local filesystem or `http://` protocols. It also supports video content from Adobe Flash Media Server using the `rtmp://`, `rtmps://`, `rtmpt://`, `rtmpe://`, `rtmpte://`, and `rtmfp://` protocols.

- AIR for TV is tailored for the system-on-a-chip (SOC) that controls an HD TV. AIR for TV models its graphics and video planes in a way that is a natural fit with HD video. AIR for TV supports H.264 using your platform's hardware accelerators, and Sorenson H.263, and On2 VP6 video codecs using its internal software decoders. It supports vector-based graphics, but AIR application developers are encouraged to use bitmap graphics whenever possible for best performance on lower-end processors.
- AIR for TV supports these audio codecs: AAC, mp3, NellyMoser, and Speex. It also supports these multichannel (more than two channels) audio codecs: AC-3, E-AC-3, DTS Digital Surround, DTS Express, DTS-HD High Resolution Audio, and DTS-HD Master Audio.

  Note: *AIR for TV supports multichannel audio for videos that are progressively downloaded from an HTTP server. Support for multichannel audio in videos streamed from an Adobe Flash Media Server is not yet available.*
- For graphics rendering operations, AIR for TV supports 2-D hardware graphics engines that perform blit and fill graphics operations.
- AIR for TV supports Stage 3D accelerated graphics rendering.
- AIR for TV supports digital rights management using Flash Access 2.0 if a platform-specific driver for this purpose is provided.
- You can develop and deploy native extensions for AIR for TV devices. These extensions allow AIR for TV applications to use ActionScript to access functionality or code specific to a device.
- When AIR for TV runs an AIR application, the application runs in the AIR application sandbox. Like any AIR application, files installed as part of the application load into the application sandbox. Any other files loaded by the application have security restrictions corresponding to the restrictions specified by the regular Flash Player security model. For more information, see The AIR application sandbox in the *ActionScript 3.0 Developer's Guide*.

  However, AIR for TV restricts application access to the file system beyond the normal application sandbox restrictions. These restrictions protect the target device's file system. For more information, see *Filesystem usage* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

System requirements

The system that runs your product that uses AIR for TV is the *target system*. For example, a target system is the Linux-based system-on-a-chip that controls a digital TV. The system you use to write C++ code and build an executable file is the *development system*.

Target system requirements

The consumer electronics device that runs your AIR for TV platform must meet these minimum hardware requirements:

Component	Minimum Requirement
Processor	600 mHz or equivalent dedicated to Flash operation
Screen	1280 x 720 ARGB8888 graphics plane with minimum 30-fps hardware compositing DirectFB or equivalent windowing system
Graphics	2-D hardware graphics engine that performs blit and fill graphics operations. OpenGL ES 2.0 library if supporting Stage 3D rendering.
Memory	128-MB RAM that makes 32 MB available to each application instance

Component	Minimum Requirement
Storage	32-MB persistent read-only local storage (ROM, flash RAM, or hard disk) for libraries and code 4 MB persistent read/write local storage (flash RAM or hard disk) for shared objects
Video	H.264 hardware decoder
Audio	Advanced Audio Coding (AAC) hardware decoder or software equivalent
Connectivity	Internet connection (100BaseT Ethernet, 802.11 wireless, or other connection)
Remote	Traditional directional or free-space

The consumer electronics device that runs your AIR for TV platform must meet these minimum software requirements:

- Linux operating system running a version 2.6-based kernel
- `glibc` or `uclibc` runtime libraries

Development system requirements

The system you use to develop and build your AIR for TV platform must meet these minimum software requirements:

- Linux operating system running a version 2.6-based kernel

  The examples in this document are based on use of the Ubuntu 10.04 LTS 32-bit Desktop distribution.
- `glibc` or `uclibc` runtime libraries
- GNU compiler collection, binutils, and the make utility

Chapter 2: Installing and building the source distribution

Develop Adobe® AIR® for TV platforms in a Linux® development environment. First, you install the source distribution in a directory you choose. Then you build AIR for TV with the make utility. The make utility uses Linux environment variables and command-line options to determine how to build AIR for TV.

If you are an OEM developer evaluating a binary distribution of AIR for TV that a silicon vendor provided, see "Working with the stagecraft binary executable" on page 16.

Development environments

You can develop *and* deploy AIR for TV platforms on any Linux distribution that is based on a version 2.6 kernel. Adobe uses the bash shell to build and test AIR for TV.

In the early phases of development, you typically build and run AIR for TV on an x86 Linux development platform that emulates your target Linux platform. You install the AIR for TV distribution onto the development platform and build AIR for TV. The build includes the modules that you develop. At some point, you begin testing on your target platform. Typically, you still build on the development platform, but use cross-compilers in the make utility to create binaries for your target platform. (Often the target platform does not have the necessary build tools).

Make sure that you can easily transfer binary executables from the build machine to the target machine. Your solution can be as simple as a USB drive that can connect to either machine. Alternatively, your solution can be as elaborate as a shared network file system. Plan ahead for the eventual need to transfer executables between distinct environments for building and testing your platform.

More Help topics

"AIR for TV directories" on page 11

Quick start on Linux

Getting started with AIR for TV on an x86 desktop Linux development system consists of the following tasks:

The example command-line instructions use the Bash shell.

Install the source distribution

To install the source distribution, take the following steps:

1 Copy the source distribution .tgz file to your Linux development machine. Copy it to the directory into which you want to install the source distribution. For suggestions on choosing your source directory, see "AIR for TV directories" on page 11.

2 Change directories to the directory that contains the AIR for TV .tgz file. In this example, the .tgz file is called stagecraft-source-distribution.tgz.

Note: *The filename includes the information about the distribution such as the release version, package date, and distribution type. A distribution type indicates, for example, that the distribution is a source code distribution.*

3 Untar the AIR for TV source distribution.

```
tar xfz stagecraft-source-distribution.tgz
```

The tar command creates a subdirectory in the working directory. The name of the subdirectory depends on the .tgz file. However, as this example continues, it uses a subdirectory named *installDir*. The new installDir subdirectory contains the subdirectories and files that make up the source distribution.

4 If your distribution includes a thirdparty-private .tgz file, copy it to the same directory as the source distribution .tgz file. Use the tar command to untar the file.

Your next step is to build the source distribution.

Note: *Do not modify the source distribution until you have built and run it successfully on at least one of the provided platforms.*

Install third-party software

Building the unpacked AIR for TV source requires some third-party software. Install the following packages:

- The X11 and the ALSA development packages, available as open source software. For example, execute the following commands to install these development packages in Ubuntu:

```
## install X11
apt-get install libx11-dev
## install ALSA
apt-get install libasound2-dev
```

The apt-get command uses the packaging system for Debian distributions. Use the packaging system for your Linux distribution to install these development packages.

Note: *The X11 and ALSA packages are necessary only for the x86Desktop platform build of AIR for TV that runs on x86 Linux platforms. These packages are typically unnecessary for building for production platforms.*

- The OpenSSL library.

Create the directory installDir/products/stagecraft/thirdparty-private/openssl/.

Retrieve the OpenSSL version 1.0.0d tarball from http://www.openssl.org/source, and put the tarball in installDir/products/stagecraft/thirdparty-private/openssl/. On many Linux systems, you can receive the tarball with the following command:

```
wget http://www.openssl.org/source/openssl-1.0.0d.tar.gz
```

The make utility defaults to building and statically linking the OpenSSL library, expecting the tarball to be in this directory. You can choose instead to dynamically link to the OpenSSL library if you already have the library on your root filesystem. For more information, see the chapter *Networking* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

- The XML2 library.

  Create the directory installDir/products/stagecraft/thirdparty-private/xml2.

  Retrieve the libXML2 library tarball libxml2-2.7.3.tar.gz from ftp://xmlsoft.org/libxml2/, and put the tarball in the directory installDir/products/stagecraft/thirdparty-private/xml2.

  The make utility builds and statically links the libxml2 library when the tarball is in this directory.

- The AIR 3 SDK

  Download the ZIP file of the AIR 3 SDK from http://www.adobe.com/products/air/sdk/.

  Create a directory to contain the ZIP file's contents. For example:

  ```
  /usr/AIRSDK
  ```

  Extract the ZIP file's contents into the directory. Set the `PATH` environment variable to include the AIR SDK bin directory. In this example, the bin directory is /usr/AIRSDK/bin.

- The Open Source Flex® SDK.

  Download the ZIP file of the Open Source Flex 4.5.1 SDK from http://opensource.adobe.com/wiki/display/flexsdk/Downloads.

  Create a directory to contain the ZIP file's contents. For example:

  ```
  /usr/flexSDK
  ```

  Extract the ZIP file's contents into the directory. Set the `PATH` environment variable to include the Flex SDK bin directory. In this example, the bin directory is /usr/flexSDK/bin.

- The Java™ runtime. The Flex SDK requires a recent Java runtime. If your development system does not yet have a Java runtime, downloads and installation instructions are at http://www.java.com/en/download/manual.jsp.

  Before building AIR for TV, set the `PATH` environment variable to include the Java bin directory.

Note: *The AIR SDK, the Flex SDK, and the Java runtime are required only if you are building native extensions for AIR for TV.*

More Help topics

"Third-party libraries" on page 13

Build on the Linux command line

To build the unpacked AIR for TV source for an x86 desktop platform such as Ubuntu, take the following steps:

1. Change directories to the installDir/products/stagecraft/build/linux/ directory.
2. Execute the make command with the quiet option.

   ```
   make quiet
   ```

   The make utility prompts you to specify the platform to build:

   ```
   Enter a valid SC_PLATFORM from the following choices:
   --> x86Desktop
   ```

Because the `SC_PLATFORM` environment variable is not set, the make utility prompts you to enter a value. The prompt list includes the `x86Desktop` platform provided with the source distribution, plus platforms you add. For more information, see "Platform-specific builds" on page 11.

Note: *Execute the make utility from the installDir/products/stagecraft/build/linux directory only. Executing the make utility from any other directory results in the following message to standard output:*

```
make: *** No targets specified and no makefile found.  Stop.
```

3 Type **x86Desktop** and then press Enter.

```
Enter a valid SC_PLATFORM from the following choices:
--> x86Desktop

--> x86Desktop
```

The make utility prompts you to specify the build mode:

```
Enter a valid SC_BUILD_MODE from the following choices:
--> debug release <--
```

The first time you build AIR for TV, create a debug build, as the next step specifies. The other build mode is `release`, which builds a non-debug version of the specified target. To suppress the make utility prompt for debug or release mode, set the environment variable `SC_BUILD_MODE` to `debug` or `release`.

4 Type **debug** and then press Enter.

```
Enter a valid SC_BUILD_MODE from the following choices:
--> debug release <--
--> debug
```

The make utility reports progress as it compiles, builds, and links each module in the AIR for TV distribution. The make utility puts the resulting object files, executable files, and libraries in the following directory:

```
installDir/build/stagecraft/linux/x86Desktop/debug
```

Build fails due to missing iconv library

One module of AIR for TV is the AEKernel module. The AEKernel module is libIAEKernel.so, and it depends on the iconv library routines (http://www.gnu.org/software/libiconv/). However, some Linux systems do not have the iconv library as part of their standard library packages. In this case, modify the necessary linking parameters of the build command that creates the libIAEKernel.so. To do so, modify the AEKernel.mk file for your platform as follows:

1 Copy the default AEKernel.mk file from this directory:

installDir/products/stagecraft/build/linux/modules

Copy it to the x86Desktop platform build directory:

installDir/products/stagecraft/build/linux/platforms/x86Desktop

Note: *If you have similar issues for your platform-specific builds, copy the default AEKernel.mk file to your platform build directory. For example:*

installDir/products/stagecraft/thirdparty-private/<*yourCompany*>/stagecraft-platforms/<*yourPlatform*>

See "Platform-specific builds" on page 11 for more information about the platform build directory.

2 Edit the copy of IAEKernel.mk. Look for the lines that set `SC_LDFLAGS_SHAREDLIB`. After these lines, add the changes required to link against your system's iconv library. For example:

```
SC_LDFLAGS_SHAREDLIB += -liconv
```

Test the build for linkage errors

Once you've built a debug version of AIR for TV without error, execute the following commands. These commands determine whether all AIR for TV modules linked properly:

```
## Set the working directory to the one that
## holds executable binaries for your platform.
## This cd command assumes that you installed the source distribution
## under your home directory.
cd ~/installDir/build/stagecraft/linux/x86Desktop/debug/bin
## Run the kerneltest program.
./kerneltest
```

After it loads the AIR for TV kernel, the kerneltest program loads and then unloads each AIR for TV module. The kerneltest program displays output similar to the following listing:

```
kerneltest: Hello Kernel Test World
Loading Module ICacheManager... ICacheManager = 0x9eccc78
Loading Module IFlashRuntimeLib... IFlashRuntimeLib = 0x9ecc910
Loading Module IFlashRuntimeGLLib...Failed to load libIFlashRuntimeGLLib.so:
libIFlashRuntimeGLLib.so: cannot open shared object file: No such file or directory
IFlashRuntimeGLLib = 0x0
Loading Module IIO...                   IIO = 0x9ecf4c0
Loading Module IJSONParserLib...        IJSONParserLib = 0x9ecc838
Loading Module IShell...                IShell = 0x9ecbca0
Loading Module IStream...               IStream = 0x9ecc830
Loading Module IURLOps...               IURLOps = 0x9ecc818
Loading Module IXMLReaderLib...         IXMLReaderLib = 0x9eceaf8
Loading Module IAudioMixer...           IAudioMixer = 0x9ee3cc8
Loading Module IGraphicsDriver...       IGraphicsDriver = 0x9ee2c78
Loading Module IImageDecoder...         IImageDecoder = 0x9ee2bc8
Loading Module IStreamPlayer...         IStreamPlayer = 0x9ed5728
Loading Module ISystemDriver...         ISystemDriver = 0x9ee07d8
Loading Module IFileSystem...           IFileSystem = 0x9ee0480
Loading Module INet...                  INet = 0x9eda330
Loading Module IProcess...              IProcess = 0x9eeb508
Loading Module IStagecraft...           IStagecraft = 0x9ede9f0
kerneltest: PASSED
kerneltest: Goodbye Kernel Test World
```

Each hexadecimal value represents the handle to a module that is loaded into memory. Any module that displays 0x0 as its handle did not load successfully. For each module that does not load, stdout displays a message saying that AIR for TV failed to load *moduleName*.so. The following example output shows such a failure:

```
Failed to load libIStream.so:
libIStream.so: cannot open shared object file:
No such file or directory
IStream = 0x0
```

If the kerneltest program runs without error, you're ready to run AIR for TV.

Run an AIR application

A device runs AIR for TV by running a binary executable file called *stagecraft*. The AIR for TV make utility builds the stagecraft binary executable.

AIR for TV displays the content of an AIR application. This content can be video, audio, animated images, or user-interface controls, such as buttons that run Adobe® ActionScript® functions. An AIR for TV application includes one or more SWF files.

To initially verify that your build of AIR for TV can execute an application, use a provided AIR application. The application is called "1" and has one SWF file called 1.swf.

Take the following steps:

1. Change directories to the one containing the stagecraft binary executable. The executable is in installDir/build/stagecraft/linux/x86Desktop/debug/bin.
2. Copy the AIR application directory called "1" to the directory containing the stagecraft binary executable. The "1" directory is in installDir/products/stagecraft/source/executables/cppunittest/testfiles. Copying the directory is a convenience for the next step. You can also use a full or relative pathname for the "1" directory in the next step.
3. Execute the stagecraft binary, passing the "1" directory as its argument:

   ```
   ./stagecraft 1
   ```

   The stagecraft binary loads and runs AIR for TV. It passes the command-line argument 1 to AIR for TV to run. The 1.swf movie displays random images at a high frame rate and displays performance statistics when it completes.

AIR for TV directories

When AIR for TV runs, it restricts its file system usage to a specific set of directories on the device. Similarly, it restricts the file system usage of the AIR applications that it runs.

On your target device, put all AIR for TV files and AIR application files in the following directory:

```
/opt/adobe/stagecraft
```

For more information, see the chapter *Filesystem usage* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Platform-specific builds

AIR for TV provides a mechanism for building multiple AIR for TV platforms from a single source tree. A *platform* is an optimized deployment of AIR for TV. The deployment is optimized for a specific combination of operating system, hardware platform, and product feature set.

For example, you might implement entry-level and deluxe versions of a product as distinct AIR for TV platforms. The deluxe platform might provide hardware acceleration of features that the entry-level platform provides in software. To implement the hardware acceleration, the deluxe platform substitutes its own modules for the software-based modules provided in the source distribution. Implementing the two products as distinct platforms allows them to coexist in the same source tree, sharing modules or providing their own as necessary.

You can use cross-compilers to build a particular platform. You specify the platform to built by setting the appropriate value for the `SC_PLATFORM` environment variable.

Put your platform-specific source and header files in a directory you create under the following directory:

```
installDir/products/stagecraft/thirdparty-private/<yourCompany>/stagecraft-platforms/<yourPlatform>
```

For example:

```
installDir/products/stagecraft/thirdparty-private/yourCompanyName/stagecraft-
platforms/yourPlatformA/
installDir/products/stagecraft/thirdparty-private/yourCompanyName/stagecraft-
platforms/yourPlatformB/
```

For details about building your platform-specific version of AIR for TV, see *Optimizing and Integrating Adobe AIR for TV (PDF)*.

AIR for TV distributions

AIR for TV distributions are tar files. The name of the tar file provides information about the distribution such as:

- the AIR for TV version number
- build number
- package date
- the distribution type, such as binary distribution or source distribution
- the target platform
- the build mode (release or debug)

Source code distributions

The standard source code distribution contains the source for only the following platforms:

- x86Desktop
- generic -- a template platform you can copy as a basis for your platform. The make utility, however, does not list it when prompting for a platform.

Silicon vendors create platform-specific source code for OEMs to use. This source distribution is in a tar file for which the filename is appended with `thirdparty-private-<silicon vendor name>`.

Note: *If you are a silicon vendor, typically you provide the platform-specific source code to Adobe, who incorporates it into the Adobe build, and redistributes a .tgz file back to you. The build date in the platform-specific .tgz filename corresponds to the build date in the standard source distribution .tgz filename. If you are an OEM, the silicon vendor provides the platform-specific .tgz file to you.*

To use a silicon vendor's source distribution, extract the .tgz file into the same directory into which you extracted the standard source code distribution. For more information, see "Install the source distribution" on page 7.

For example:

```
tar xfz stagecraft-3.0.0.20110825-src.tgz
tar xfz stagecraft-3.0.0.20110825-thirdparty-private-siliconvendora.tgz
```

Extracting the first file creates the directory stagecraft-3.0.0.20110218-src. Extracting the second file adds the directory stagecraft-3.0.0.20110218-src/thirdparty-private/siliconvendora.

This extraction into thirdparty-private provides the code and makefiles necessary to build AIR for TV for that particular target platform.

Binary distributions

Adobe provides binary distributions of AIR for TV for the hardware evaluation reference platforms of several silicon venders. Adobe delivers these binary distributions as compressed tar files to the silicon vendors. The silicon vendors typically include the binary distribution in the Linux distribution that they deliver with their hardware evaluation reference platform.

If you are an OEM developer, you receive the appropriate AIR for TV binary distribution with the hardware reference platform from your silicon vendor. You can use this binary distribution to evaluate AIR for TV and to develop your solution. For a deployment contract, contact Adobe.

See "Working with the stagecraft binary executable" on page 16 for more information about running AIR for TV.

Platform SDK distributions

AIR for TV provides distributions of platform software development kits (SDKs). These development kits are:

- The Driver Development Kit (DDK). For more information, see *Optimizing and Integrating Adobe AIR for TV (PDF)*.
- The Extensions Development Kit (EDK) for building native extensions. For more information, see *Developing Native Extensions for Adobe AIR*.
- The Flash Access Porting Kit for implementing a Flash Access adaptor interface for your device. See *Porting ADOBE® FLASH® ACCESS™ to Adobe AIR for TV*.

These development kits allow a platform or system developer to create platform-specific modules and native extensions without having the full source distribution of AIR for TV. These kits provide only the header and source files you need for a particular development task. For information on how to create an SDK for distribution, see *Optimizing and Integrating Adobe AIR for TV (PDF)*.

If you receive a DDK platform distribution from a silicon vendor, the tar file does not include driver source code specific to the silicon vendor. Obtain such source code from the silicon vendor and install it into the DDK distribution as in the following example:

```
# tar xf stagecraft-3.0.0.20110825-sdk-ddk-siliconvendora-debug.tgz
# tar xf stagecraft-3.0.0.20110825-src-thirdparty-private-siliconvendora.tgz
```

The first tar command extracts the DDK platform distribution. The next tar command extracts the driver source code that is specific to the silicon vendor.

Documentation distribution

The documentation for AIR for TV is available in a tar file. The tar file contains the PDF files for:

- *Getting Started with Adobe AIR for TV*
- *Integrating and Optimizing Adobe AIR for TV*

Third-party libraries

AIR for TV depends on some third-party libraries to build and execute. In some cases, the source distribution provides the third-party library. In other cases, get the third-party library from another source. For example, get the openssl library from the Internet, as discussed in "Install third-party software" on page 7.

The following list includes some of the libraries that AIR for TV uses. The list also gives the related chapter, if any, in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

- The cURL library. AIR for TV requires this library for networking access. See the chapter *Networking*.
- The openssl library. AIR for TV requires this library for HTTPS operations. See the chapter *Networking*.
- The FFMPEG library. AIR for TV uses this library in a sample overlay video implementation. See the chapter *The audio and video driver*.
- The FontConfig and FreeType libraries. You can use these libraries for device font support. See the chapter *The device text renderer*.
- The ICU library. AIR for TV can support the flash.globalizaton package with this library, depending on your platform requirements. See the chapter *The system driver*.
- The ALSA library. AIR for TV requires this library to build for the x86Desktop platform. See "Install third-party software" on page 7. AIR for TV provides an IAudioMixer implementation using this library. See the chapter *The audio mixer*.
- The X11 library. AIR for TV requires this library to build for the x86Desktop platform. See "Install third-party software" on page 7.
- The XML2 library. AIR for TV requires this library to build. See "Install third-party software" on page 7.
- The AIR SDK. AIR for TV requires this SDK to build its sample ActionScript extensions. Starting with AIR 2.5, the AIR SDK supports AIR for TV. Builds of your own native extensions also use the AIR SDK. See "Install third-party software" on page 7.
- The Open Source Flex SDK. AIR for TV requires this SDK to build its sample native extensions. Builds of your own native extensions also use this SDK. See "Install third-party software" on page 7.
- The Java runtime. The Flex SDK requires the Java runtime. See "Install third-party software" on page 7.

The LD_LIBRARY_PATH environment variable

If your build of AIR for TV includes only AIR for TV modules listed in the kerneltest program's output, setting the `LD_LIBRARY_PATH` environment variable is unnecessary.

However, if your build includes other libraries or executables, set the `LD_LIBRARY_PATH` environment variable before running AIR for TV or other executables. Set `LD_LIBRARY_PATH` to include the directory containing the library or executable.

You can install these other libraries or executables in the same bin directory as the AIR for TV modules. Even in this case, however, setting `LD_LIBRARY_PATH` is required before you can execute these libraries or executables.

Make utility command-line options

The make utility writes verbose output by default. This output allows you to observe progress as AIR for TV compiles, builds, and links each module in the platform. For more information about creating platform-specific modules and the corresponding makefiles, see the chapter *Coding, Building, and Testing* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Once a platform builds reliably, you can reduce diagnostic output by passing the quiet option to the make utility:

```
make quiet
```

By default, the make utility builds all modules, performing incremental builds as needed. However, you can also force the make utility to rebuild every component of a platform without reusing any object files generated by previous build. To rebuild everything, pass the clean option to remove the object files, then run the make utility again to build all targets:

```
## Remove old build output files
make clean
## Build everything
make
```

By default, the make utility builds all modules of the platform that the `SC_PLATFORM` environment variable specifies. You can build an individual module by passing the name of the module to the make utility. This value is the name of a module-specific makefile (found in the installDir/products/stagecraft/build/linux/modules/ directory) without the .mk filename extension. For example, the following command makes the IStreamPlayer module:

```
## These commands assume you installed the source distribution
## in your home directory.
cd ~/installDir/products/stagecraft/build/linux
## build IStreamPlayer
make IStreamPlayer
```

You can also use the clean option on individual modules. For example:

```
## Clean the stream player module
make clean-IStreamPlayer
```

Note: *Always run the make utility from within the installDir/products/stagecraft/build/linux/ directory, even when building individual modules.*

You can also use the rebuild option to force rebuilding. This option first removes all output files (like the option clean), and then rebuilds all the modules. For example:

```
## rebuild everything ('all' is the default target)
make rebuild
## rebuild the stream player module
make rebuild-IStreamPlayer
```

You can pass a list of modules to build. The modules can appear in any order on the command line; the make utility automatically builds them and any dependent objects in the correct dependency order.

```
cd ~/installDir/products/stagecraft/build/linux
make libjpeg IStreamPlayer
```

You can combine options:

```
make quiet rebuild-IAEKernel IShell
```

You can also build the test applications cppunittest and audiotest, along with the AIR for TV modules. Use the following command:

```
make test
```

For more information about cppunittest and audiotest, see the following references in *Optimizing and Integrating Adobe AIR for TV (PDF)*:

- *Executing unit tests* in the chapter *Coding, building, and testing*
- *Testing your audio mixer* in the chapter *The audio mixer*

Chapter 3: Working with the stagecraft binary executable

A device runs AIR for TV by running a binary executable file called *stagecraft*. The AIR for TV make utility builds the stagecraft binary executable. The source file for the stagecraft binary executable is in:

```
installDir/products/stagecraft/source/executables/stagecraft/stagecraft_main.cpp
```

To run stagecraft from a C++ application, create a process and launch the stagecraft binary executable in it.

Running the stagecraft binary from the command line

After you have built AIR for TV and determined that all its modules linked correctly, you can run it. When you run it, you specify the AIR application to run.

You can specify one of the following:

- An installed AIR application.

  Typically in a production environment, you specify an installed AIR application. You install an AIR application from its AIR file. An AIR application developer packages the application into an AIR file for distribution and installation. For more information, see "Running an installed AIR application" on page 18.

- An ad-hoc AIR application

  You can test an AIR application that has not yet been packaged into an AIR file. This testing is useful when developing an application. It is also useful when testing the integration and optimization of AIR for TV on your platform. For more information, see "Running an ad-hoc AIR application" on page 17.

Before running either type of AIR application, make sure that its AIR application descriptor file follows the rules specified in "The AIR application descriptor file" on page 17.

Installing an AIR application

An AIR application is distributed in an AIR file. An AIR file is an archive file that contains the application files. Application files include:

- "The AIR application descriptor file" on page 17
- SWF files
- Resource files, such as image files

To install an AIR application, do the following:

- Create a directory under /opt/adobe/stagecraft/apps. For example:

  ```
  /opt/adobe/stagecraft/apps/HelloWorld
  ```

  For more information about this directory, see the chapter *FileSystem usage* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

- Copy the AIR file, such as HelloWorld.air, to the new application directory.

- Extract the files archived in the AIR file to the application directory. Preserve the directory structure that is in the AIR file. Since an AIR file is an archive file, you can use most extraction tools for this step.

Note: *Before installing an AIR application, validate the application's signature.*

The AIR application descriptor file

Every AIR application requires an application descriptor file. An AIR application developer creates this XML file using, for example, Adobe Flash Professional CS4 or CS5, or using Adobe Flash Builder 4.5.

However, starting with AIR 2.5, the application descriptor file has differences compared to previous AIR releases. Adobe® Flash® Professional CS5.5 supports these changes when you use it to publish an AIR application. Similarly, Adobe Flash Builder 4.5 supports these AIR 2.5 changes.

However, if you are not using Flash Builder 4.5 or the Adobe Flash Professional CS5.5, modify the application descriptor file as follows:

- Change the published version of AIR to at least version 2.5. Do so by changing the `<application>` XML element as follows:

  ```
  <application xmlns="http://ns.adobe.com/air/application/2.5">
  ```

- Replace the `<version>` XML element with a `<versionNumber>` XML element to specify the version of the application. For example:

  ```
  <versionNumber>1.0.0</versionNumber>
  ```

  Note: *The `<versionNumber>` element requires the form <0-999>.<0-999>.<0-999>.*

- If the application descriptor file contains a `<supportedProfiles>` tag, include either `TV` or `extendedTV` in the value. AIR for TV can also run the application if the file contains no `<supportedProfiles>` tag, because no such tag assumes that the application supports all profiles.

For more information about the AIR application descriptor file, see Setting AIR application properties in *Building Adobe AIR applications*.

Note: *For an AIR application to use native extensions for Adobe AIR, run AIR for TV with the command-line option --profile extendedTV. See "The AIR profile" on page 21.*

Running an ad-hoc AIR application

An ad-hoc AIR application has not been packaged into an AIR file, and therefore has not been installed from the AIR file. Instead, it is a collection of SWF files and resource files such as image files. Like an installed AIR application, it requires an application descriptor file. However, the files of an ad-hoc AIR application are not necessarily in the same directory layout that installed AIR applications use.

To execute AIR for TV so it runs an ad-hoc AIR application, do the following:

1. Modify the application descriptor file as described in "The AIR application descriptor file" on page 17.
2. Run the stagecraft executable as follows (from in the bin directory in this example):

   ```
   ./stagecraft --airbase <AIR application base path> <application.xml>
   ```

   The *`<AIR application base path>`* parameter is a full or relative path to the base directory containing the AIR application SWF files and resource files.

   The *`<application.xml>`* parameter is the full or relative filename of the application descriptor file.

   Note: *Relative paths are relative to the working directory.*

For example, the following command specifies that:

- the SWF file and resource files are in $HOME/apps/HelloWorld.
- the application descriptor file is in $HOME/apps.

```
## Run an ad-hoc AIR application
./stagecraft --airbase $HOME/apps/HelloWorld $HOME/apps/HelloWorld-app.xml
```

The stagecraft binary runs the requested application. The stagecraft binary exits when the AIR application exits.

Running an installed AIR application

An AIR application is installed on a device if its files are in a directory structure that is identical to the application's package structure. To install an AIR application on a device, see "Installing an AIR application" on page 16.

To execute AIR for TV so it runs an installed AIR application, do the following:

1. Modify the application descriptor file as described in "The AIR application descriptor file" on page 17.
2. Run the stagecraft executable as follows:

   ```
   ./stagecraft <AIR application installation path>
   ```

   The *<AIR application installation path>* parameter is a full or relative path to the installation directory of the AIR application.

   Note: *Relative paths are relative to the working directory.*

   For example, the following command runs a Hello World AIR application:

   ```
   ## Run an installed AIR application
   ./stagecraft /opt/adobe/stagecraft/apps/HelloWorld
   ```

The stagecraft binary runs the requested application. The stagecraft binary exits when the AIR application exits.

Running multiple processes

AIR for TV is composed of modules. One module is the IFlashRuntimeLib module, which contains the AIR runtime. The AIR runtime controls the rendering of the AIR application's SWF content. The AIR runtime runs in its own window, controlled by a *StageWindow instance*. Each process running AIR for TV has only one StageWindow instance at a time.

You can run multiple AIR applications concurrently by running multiple process of AIR for TV. However, each AIR for TV process must run a different AIR application than other concurrently running AIR for TV processes.

If your platform uses the DirectFB library, use the DirectFB multi-application core to allow multiple processes. For more information, see *DirectFB* in the chapter *The graphics driver* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

One way to run multiple processes is to run at least one process in the background. For example:

1. On the Linux command line, go to the directory containing stagecraft binary executable and enter the following command to run the process in the background:

   ```
   ./stagecraft --outputrect x,y,width,height  /opt/adobe/stagecraft/apps/HelloWorld &
   ```

   Enter values for *x,y,width,height* to provide the position and size of the window of the StageWindow instance.

2 Start a second AIR for TV process by running the same command, but specify a different AIR application. Since the first process is running in the background, you can enter the second command on the command line. In the second command, specify a different position for the window of the StageWindow instance. For example:

```
./stagecraft --outputrect x2,y2,width2,height2/opt/adobe/stagecraft/apps/SomeOtherApp
```

For more information about the `--outputrect` option, see "Display options" on page 21.

Note: *X11 windowing systems do not observe window placement specified by the* `--outputrect` *option.*

Using the stagecraft binary command-line options

You can use command-line options to change the default behavior of the stagecraft binary and its interface to AIR for TV. If you do not specify an option, stagecraft passes default values to AIR for TV.

Note: *All options that affect playback of an AIR application must appear on the command line before the AIR installation path or* `--airbase` *parameter.*

Displaying all stagecraft options on the command line

Execute stagecraft with no arguments to display usage information (including options and default values) on the Linux command line:

```
## cd to bin/ directory of the platform you built.
## This example is for x86Desktop debug builds
cd ~/installDir/build/stagecraft/linux/x86Desktop/debug/bin
./stagecraft
```

The output from the command is the following:

```
Adobe (R) AIR (R) for TV
(C) 2008-2011. Adobe Systems Incorporated. All rights reserved.
Patent and legal notices: http://www.adobe.com/go/digitalhome_thirdpartynotice
Using: AIR_3.0

usage: stagecraft [--options] air-install-dir-path
    stagecraft [--options] --airbase air-base-dir-path application.xml
    Valid options are:
    --airbase <air-base-dir-path>
    --aircmdline "command-line args for AIR"
    --profile [tv|extendedTV] (default is tv)
    --extensionsDir <extension directoy>
    --contentdims w,h
    --outputdims w,h
    --outputrect x,y,w,h
    --bgalpha <0-255>
    --gl [on|off] (overrides default OpenGL mode)
    --keymap filename
    --sslclientcerttable <sslclientcerttable-string>
    --sslcertsdir <ssl certificates directory>
    --tracefps <samplePeriod[MS|S]> (e.g. 5000MS or 5S)
    --debugmode
    --graphicsmemthreshold <num[M]>
    --astrace
    --noastrace
    --nospeedlimit
```

```
    --fontdirs comma separated Dir names
    --proxy <dottedIP-String> <port>
    --proxyauthentication <username> <password>
    --pause
    --dcts
    --spdump
    --modulemap <moduleName:filename>
    --instrument
    --mousestream <dottedIP-String>:<port>
    --mousefeedback <seconds between screenshots>
    --locale <locale for language and formatting>
default values are:
    --profile tv
    --sslcertsdir /opt/adobe/stagecraft/data/config/ssl/certs/
    --contentdims <swf authored stage dims>
    --outputdims <swf authored stage dims>
    --bgalpha 255
    --graphicsmemthreshold 30M
    --extensionsdir /opt/adobe/stagecraft/extensions/
    --mousefeedback 2
    --locale en_US.UTF-8
    --astrace
    --debugmode
Other options for the stagecraft executable:
    --notrackmem: disables memory leak detection tracking
    --tracefault: enables stacktrace and registry dump on crash
    --useshell: enables the IShell
    --noclearscreen: prevents the screen from being initially cleared
```

Note: *This output occurs when running in debug mode. The options* `--notrackmem` *and* `--tracefault` *are available only when running in debug mode. The options* `--mousestream` *and* `--mousefeedback` *are only for Adobe internal use.*

Running the AIR application

You can run an installed AIR application with the following command line usage:

```
stagecraft [--options] air-install-dir-path
```

For information, see "Running an installed AIR application" on page 18.

You can run an ad-hoc AIR application with the following command line usage:

```
stagecraft [--options] --airbase air-base-dir-path application.xml
```

For information, see "Running an ad-hoc AIR application" on page 17.

Passing parameters to the AIR application

Use the following option to pass parameters to the AIR application:

```
--aircmdline "command-line args for AIR"
```

List the `--aircmdline` option only once. To pass multiple parameters to the AIR application, enclose them in quotes.

For example, to run an AIR application with the options `"-mode fast -color red rocketship.dat"`, use the following command:

```
./stagecraft --aircmdline "-mode fast -color red rocketship.dat"
    /opt/adobe/stagecraft/apps/myAIRapp
```

AIR for TV runs the AIR application installed in the directory myAIRapp. The AIR application receives five command-line parameters: `-mode`, `fast`, `-color`, `red`, and `rocketship.dat`.

Note: *An AIR application gets its command-line parameters using ActionScript by listening for the event* `InvokeEvent`*:*

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvoke);
function onInvoke(event:InvokeEvent):void
{
    for (var i:Number = 0; i < event.arguments.length; i++)
    {
        trace("arg " + i + ": " + event.arguments[i]);
    }
}
```

For more information, see InvokeEvent in the *ActionScript 3.0 Reference for the Adobe Flash Platform*.

The AIR profile

Use the following option to set the AIR profile that the AIR application uses:

```
--profile [tv|extendedTV] (default is tv)
```

The AIR profile defines a set of supported ActionScript APIs and capabilities available to the AIR application. AIR for TV runs applications that use either the `tv` or `extendedTV` profile.

The `extendedTV` profile provides one additional capability to AIR applications compared with the `TV` profile. Specifically, the `extendedTV` profile allows an AIR application to use any ActionScript extensions that have been installed on the device as device-bundled extensions

Note: *An AIR application can use its extensions only if you specify the option* `--profile extendedTV`*.*

For more information, see Application profiles in *Building Adobe AIR Applications*. Also see *Developing Native Extensions for Adobe AIR*.

Display options

The following options modify default values that the AIR runtime uses to interpret and render SWF content:

- `--contentdims` *`widthPixels,heightPixels`*

  Initial width and height (expressed in pixels) of the render plane. The render plane is the bitmap on which the AIR runtime renders each frame of Flash animation. By default, the dimensions of the render plane are equal to the Stage size specified when authoring the SWF file.

  Depending on the SWF content, specifying a smaller render plane size can reduce system processor usage. For more information, see *Plane dimensions and scaling* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

  For hardware-based scaling, use the `--outputdims` and `--outputrect` options instead of the `--contentdims` option. Hardware scaling is a scaling of the final bitmap: it is faster than the vector-based scaling that the AIR runtime performs, but produces poorer quality.

  Note: *ActionScript in the SWF content can use the* `stageHeight` *and* `stagewidth` *properties of the flash.display.Stage object to get and set the dimensions of the render plane.*

- `--outputdims` *`widthPixels,heightPixels`*

  Width and height (expressed in pixels) of the output plane. The output plane is the bitmap used to display each completed frame of animation on the display device. On platforms that provide a windowing system, each output plane appears in its own window.

By default, the output plane is the same size as the render plane. You can use the `--outputdims` option to scale output. For more information, see *Plane dimensions and scaling* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Note: *ActionScript in the SWF content can use the `screenResolutionX` and `screenResolutionY` properties of the flash.system.Capabilities class to get the dimensions in pixels of the display device. An output plane's dimensions can cover either the entire display of the display device, or a subrectangle of the display.*

- `--outputrect` *`xPixels,yPixels,widthPixels,heightPixels`*

  Size and location of the output plane. The *`widthPixels,heightPixels`* values specify the width and height of the output plane in pixels, just as the `--outputdims` option does. By default, the output plane is the same size as the render plane. The *`xPixels,yPixels`* values specify the position of the upper-left corner of the window that displays the output plane.

  Specifically, the *`xPixels,yPixels`* values specify the coordinates of the upper-left corner of the window in relation to the upper-left corner of the display device. Passing `0,0` as this coordinate pair places the window in the upper-left corner of the display device. Passing `20,10` places the window 20 pixels to the right and 10 pixels down from the upper-left corner of the display device.

  However, your graphics driver implementation determines whether it can position the window containing the output plane according to `xPixels` and `yPixels`. In graphics drivers provided with the source implementation, X11 windowing systems do not observe the values of `xPixels` and `yPixels`. Instead, they tile multiple windows. Depending on your platform, the graphics driver can alter or ignore the `xPixels,yPixels` values you specify.

- `--bgalpha` *`int 0 to 255`*

  The alpha channel (transparency) value of the background on which the AIR runtime renders the AIR application. Valid values are any integer 0 - 255. A value of 0 represents a fully transparent background and a value of 255 represents an opaque background. The default value is 255.

  Use this option to blend AIR application content with other planes of graphics or video on your embedded system that provide a hardware-based compositor. That is, this value allows an underlying hardware-based graphics or video display to be visible through the AIR application.

  For example, consider an AIR application that provides a user interface for viewing broadcast television. Use this option to overlay the user interface semi-transparently over the broadcast content. Setting the `--bgalpha` option to a semi-transparent value allows the broadcast content beneath the AIR application to "show through" the controls that the AIR application presents.

  Note: *An AIR application that uses overlay video -- rendering and displaying video in the hardware -- always obscures anything layered beneath it.*

- `--nospeedlimit`

  Requests that the AIR runtime run the SWF content at the maximum frame rate possible. It also requests that the AIR runtime does not sleep in its main loop, and that it renders frames as fast as it can. Using this option can cause SWF content to play much faster than expected. Use this option to determine the maximum frame rate that a specific platform can deliver.

AIR module

The following options affect which AIR module to load to run the SWF content of the StageWindow instance:

`--gl [on|off]`

The option `--gl on` means that the AIR module that loads supports OpenGL ES 2.0. The option `--gl off` means that the AIR module that loads does *not* support OpenGL ES 2.0. Not supporting OpenGL ES 2.0 is the default.

Use `--gl on` if your platform supports Stage 3D hardware-accelerated graphics.

Key mapping

`--keymap` *`filename`*

The `--keymap` option specifies the *`filename`* of the key map XML file.

About key maps

A key map is an XML file that maps key codes used in AIR for TV to key codes used in the ActionScript environment. The key codes used in AIR for TV are defined in `include/ae/stagecraft/StagecraftKeyDefs.h`. Your platform implementation for handling user input uses these key codes for dispatching user input events into AIR for TV. For example, when a user presses the Select key on a remote control device, your platform implementation dispatches the key code `AEKEY_ENTER` to AIR for TV.

AIR for TV then uses a default mapping to determine which ActionScript key code to pass to the SWF content. For example, `AEKEY_ENTER` maps by default to the ActionScript key code `Key.ENTER`. This default key mapping is in StagecraftKeyDefs.h in the section commented with "Standard Flash key symbols in AEKEY space".

A key map file allows you to replace the default mapping without rebuilding AIR for TV. A key map file is useful, for example, when distributing SWF content authored for a desktop to a consumer electronics device. Typically, consumer electronics devices do not have a full keyboard for user input. For example, consider a desktop game in which the SWF content interprets pressing the spacebar to mean "shoot". The key map file maps the Select key on the device's remote control to the spacebar.

Key map XML file format

The following is an example of the XML format of a key map file:

```
<?xml version="1.0" encoding="UTF-8"?>
<keymap>
    <entry>
        <input type="aekey">AEKEY_INFO</input>
        <output type="char">i</output>
    </entry>
    <entry>
        <input type="char">'x'</input>
        <output type="flashkey">Key.ESCAPE</output>
    </entry>
    <entry>
        <input type="char">&#x26;</input>
        <output type="char">&#65;</output>
    </entry>
</keymap>
```

The XML format has one root element named `keymap`. The `keymap` element must contain one ore more `entry` elements. Each `entry` element must contain an `input` and an `output` element. Each `input` and `output` element must have a `type` attribute. The `type` attribute identifies the type of value that the element contains. The `type` of an `input` element must be `aekey` or `char`. The type of an output element must be `aekey`, `char`, or `flashkey`.

The first entry in the XML example maps a `AEKEY_INFO` key event from AIR for TV to the ASCII character 'i' in the ActionScript environment. For this example, the remote control is configured to send an `AEKEY_INFO` key value to AIR for TV when the user presses the "info" button. Then, this mapping directs AIR for TV to behave as if the user pressed the 'i' key on a computer keyboard. Therefore, AIR for TV delivers the 'i' key event to the SWF content. The second entry maps from the ASCII 'x' character to the ActionScript Key constant `Key.ESCAPE`. For this example, the user input device has a physical keyboard but has no Esc key. This mapping causes the 'x' key to behave like the Esc key to the SWF content. The last entry demonstrates using XML numeric character values for `char` values in either the input or the output elements.

Determining which key map to use

AIR for TV uses the following steps to determine the key mapping:

1. AIR for TV uses the key map file that you specify on the stagecraft command line.
2. If the stagecraft command line does not specify a key map file, AIR for TV uses the key map file specified in a parameter in an interface call from the host application.
3. If no keymap.xml file is specified, AIR for TV uses its default key mapping.

Font directories

AIR for TV provides a set of default device fonts in /opt/adobe/stagecraft/fonts. Use the following command-line option only if the provided default fonts do not meet your needs. This command-line option sets the directories where font files that your platform provides are installed:

```
--fontdirs comma separated Dir names
```

List the full path names of the directories containing the font files. Separate the path names with commas. Do not include spaces.

These directories are used for searching for font files by your device text renderer implementation. For more information, see *The device text renderer* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Native extensions directory

Use the following option to set the directory where native extensions are installed:

```
--extensionsdir <extension directory>
```

Provide a full path name.

The extensions directory defaults to /opt/adobe/stagecraft/extensions.

For more information, see *Developing Native Extensions for Adobe AIR*.

HTTP proxy information

AIR for TV supports HTTP requests through a proxy server. You can specify proxy information for the StageWindow instance. When you specify a valid proxy host IP address to the StageWindow instance, all HTTP requests in the SWF content are sent to the proxy server. Use the following command-line options to specify the proxy information:

- `--proxy host port`

  For `host`, specify the proxy server IP address. Specify a string value in dotted decimal notation: xxx.xxx.xxx.xxx where xxx is 0 – 255.

  For `port`, specify the integer value of the proxy server port.

- `--proxyauthentication` *`username password`*

  Specify the user name and password for the proxy server.

For more information, see *HTTP requests through a proxy server* in the chapter *Networking* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

SSL certificates

`--sslclientcerttable` *`sslclientcerttable-string`*

The AIR runtime can have a table of Secure Sockets Layer (SSL) client certificates. The `--sslclientcerttable` option allows you to associate a target host name with a certificate and private key. The client side uses the certificate and key in SSL mutual authentication with that target host.

For information about the format of the *`sslclientcerttable-string`*, see *HTTPS mutual authentication* in the chapter *Networking* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

You can also override the default directory for storing SSL certificates. Use the following command-line option:

`--sslcertsdir` *`<ssl certificates directory>`*

If the certificates are encrypted, store the certificate bundle (the ca-bundle.crt file) in:

`<ssl certificates directory>``/encrypted`

If the certificates are unencrypted, store the certificate bundle (the ca-bundle.crt file) in:

`<ssl certificates directory>``/unencrypted`

The directory you specify in *`<ssl certificates directory>`* cannot be the `/home` directory or any subdirectory of the `/home` directory.

For more information, see *Certificate authority (CA) certificates* in the chapter *Networking* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Locale setting

AIR for TV supports the flash.globalization package. This package is a set of ActionScript classes that provide language-specific, country-specific, and region-specific support for numbers, currencies, dates, and times. Use the following option to set the locale for the AIR for TV application:

`--locale <locale for language and formatting>`

If you do not specify this option, AIR for TV uses the locale `en_US.UTF-8`.

For more information, see *Locale support* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Generating performance statistics

You can use the `--tracefps` option to generate performance statistics. For more information, see *Measuring performance* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

Other options for debugging and performance tuning

Other options to the stagecraft binary control debugging and performance-tuning functionality.

- `--astrace`

  Prints ActionScript `trace()` output to the command line. This functionality occurs by default in debug builds. Pass `--noastrace` to disable this functionality.

- `--noastrace`

  Disables command-line output of ActionScript `trace()` statements. This setting is the default in release builds.

- `--debugmode`

  Prints error messages and ActionScript stack traces when an unhandled exception occurs in ActionScript execution. The option is always on when you use a debug build. In a release build, the option is off by default. Use `--debugmode` to turn on the option in a release build. Setting `--debugmode` automatically sets `--astrace`.

  When the `--debugmode` option is off and an ActionScript error occurs, you get a numeric error code. No stack trace is available.

  Note: *Using `--debugmode` impacts the performance of AIR for TV.*

- `--notrackmem`

  Disables detection of memory leaks.

- `--tracefault`

  Enables a stack trace and registry dump when a crash occurs.

- `--graphicsmemthreshold <num[M]>`

  Not supported. Do not use.

- `--showblit`

  Enables a transparent overlay color.

  During development in debug mode, you can use a transparent overlay color to tint a blitted or filled region that the hardware accelerators render. The overlay color allows you to visually determine which objects your I2D implementation is rendering, and which objects the AIR runtime is rendering.

  For more information, see *FillRect() method* in the chapter *The graphics driver* in*Optimizing and Integrating Adobe AIR for TV (PDF)*.

- `--useshell`

  Enables the IShell, which provides an interactive shell implementation that you can use to pass commands to AIR for TV while it is running. For more information, see the `IShell.h` and `IShellCommand.h` files in the source distribution directory include/ae/core/shell.

- `--pause`

  Suspends executing the SWF content until you enter `go` in the stagecraft interactive shell. Using `--pause` automatically starts the interactive shell.

- `--noclearscreen`

  Disables clearing of the display during initialization. The default behavior is to clear the display before executing the SWF content.

- `--dcts`

Enables running DCTS.

For more information on using DCTS, see *Certification testing* in the introductory chapter of *Optimizing and Integrating Adobe AIR for TV (PDF)*.

- `--spdump`

  Enables command-line output of StreamPlayer buffer levels. For more information about buffers in StreamPlayer implementations, see *The audio and video driver* in *Optimizing and Integrating Adobe AIR for TV (PDF)*.

- `--modulemap` *`moduleName:filename`*

  Substitutes another library file for the specified module. For example, if you want to use a graphics driver library named MyTmpGraphicsDriverLib.so, use the following command-line argument:

  ```
  --modulemap IGraphicsDriver:MyTmpGraphicsDriverLib.so
  ```

- `--instrument`

  Prints performance times in milliseconds about starting and shutting down AIR for TV. Specifically:

  - Time from starting AIR for TV until AIR for TV has loaded the AIR application.
  - Time from starting AIR for TV until AIR for TV has started playing the AIR application.
  - Time from starting AIR for TV until AIR for TV has drawn the first screen of the AIR application.
  - Time to shut down AIR for TV from when the AIR application exits.

Using double-byte filenames on the command line

You can use double- byte filenames and directory names on the command line. For example, you can use Chinese, Japanese, or Korean characters.