Enabling better media workflows

An overview of the Advanced Authoring Format

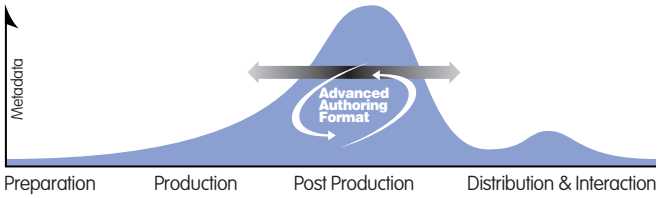
by the AAF Association

Advanced Authoring Format (AAF) is an industry-driven, open standard for multimedia authoring and post production, created by the AAF Association. It enables content creators to easily exchange digital media, video and audio, and metadata across platforms and between applications. It simplifies project management, saves time and preserves valuable metadata that was often lost in the past during media transfers.

AAF improves workflow in post production and authoring where individual systems and applications have become isolated by incompatibilities: so limiting their interaction. Typically, video and audio always have safe passage but detail about the material and edit decisions is often lost.

AAF allows the passage of full information. Not just the video, audio and text material—termed essence—but also the metadata with the decisions about how material has been manipulated (cuts, DVE, color correction etc.) and assembled. The metadata also passes on existing, original information such as timecode or edgecode, ownership, previous editing etc. that helps any later versioning.

Editing is about the assembly of material for passive viewing. Authoring also refers material assembly but the product is interactive—as seen in CD-ROMs, DVDs, Internet and interactive television. In either case it is where metadata is most prevalent and where the flexibility to alter information is vital. Other sectors of the production chain have different requirements, and other file formats exist to handle these efficiently. AAF interfaces with these but does not replace them: each does the best job in its own area.



Metadata peaks in the post production/authoring phase

AAF is not only compression-independent, it is also platform-independent: working on all the most widely used operating systems. It is broadly accepted and supported from key industry companies and organizations showing the high level of interest, importance and value attached to the AAF movement (see "Membership"). Mark Wildig, Smoke & Mirrors managing director, made this very clear:

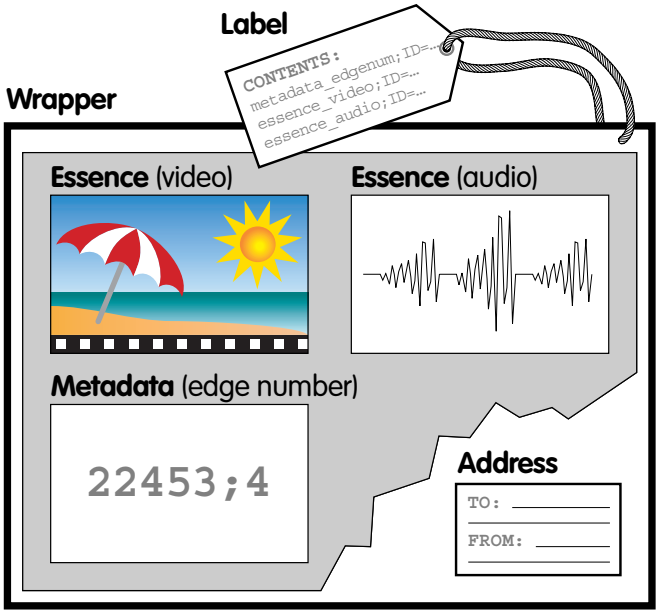
> *"It's so important for the future of our industry to have the seamless interchange of file formats—and I'd love to see AAF used as a delivery format for the TV stations. AAF will become a key technology for the post production industry. It enables the interchange and archiving of all the elements involved in the structure of complex layered material. So those key elements will always be available to be picked apart and worked on across different systems, and delivered to different media."*

The AAF Software Developer's Kit (SDK) Version 1.0 was officially released in April 2001, after members of the AAF Association performed extensive beta testing. It can be obtained from www.aafassociation.org, where you can either download the standard version of the SDK as open source, or register as an adopter to become an AAF developer.

Since that time, numerous companies have made contributions to the AAF technology. As a result, the Association is planning a maintenance release which will incorporate many of these contributions. In addition, sample files created by AAF Developer Jim Trainor are now available to AAF members.

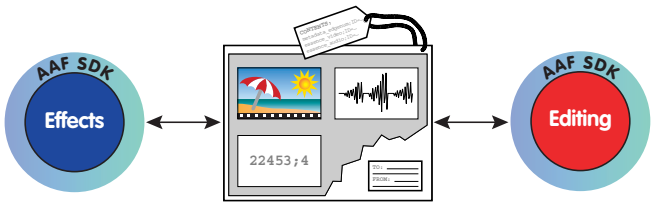
Quick Tour

The principle of AAF is quite straightforward and briefly explained here in this quick tour. Deeper detail follows. AAF takes program content, the essence and metadata, places it into a wrapper (a file format) adds an address and attaches a label on the outside giving a basic description of what's inside.



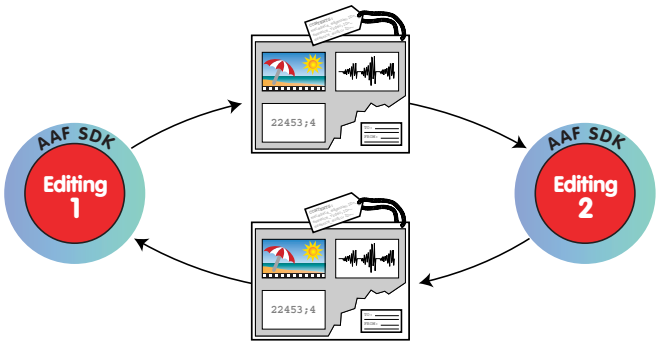
The AAF file holds essence and metadata in a wrapper with an address and label

Then another AAF-compatible system or application can look at the label and display information about what's inside. It can open the wrapper and, if it has the right application, work with its contents. Then information about any work done by the application, say to re-size the video, is added to the metadata which continues as a total record about its related essence. Any modified or new video and any other new essence along with modified metadata are inserted into the existing file.



AAF-compliant applications are able to exchange files

This procedure is very analogous to e-mails and their attachments. You can read the normal text and see the file name of the attachment. If you have the required application (say Acrobat for pdf files) you can open the attached file. If you don't have the application you are still able to forward the file—NOT removing any information but the record of your forwarding is added—plus any comments.



In a "round trip", any modifications made to a file are recorded in its metadata

There are many variants to workflow and much attention paid to this in AAF. For example, a system may send a file to an application and receive the result straight back. Known as a Roundtrip, information about alterations made to the file during the trip must be included in the metadata.

The AAF file is designed for easy updating which is a great benefit when authoring or editing. It allows new editing information to be added without having to re-write the whole file. At the end of the authoring/post phase, the file holds the finished program. This may not exist as a continuous piece but as sections of original essence and instructions (in the metadata) for how they should be manipulated and assembled. For onward applications, like distribution and transmission, the file is 'flattened' (see "Program Delivery"), resolving the data into a continuous program and greatly simplifying the metadata. Unflattened files may be useful as archives for any future reworks.

GXF and MXF file formats receive special mention. GXF was developed by the Grass Valley Group, also a member of the AAF, and is the existing file transfer format currently used by the Profile XP and Vibrint editing systems. MXF is being developed by Pro-MPEG together with the AAF Association. The data model in MXF is in fact a subset of AAF—its data structure and names of data elements are a subset of AAF's. This maximizes interoperability between AAF and MXF; an AAF application can open an MXF file thinking it is an AAF file. It just works!

Open

AAF has to address the needs of the authoring and post production industries without restriction. It must be open, not a proprietary, closed format and have no sigma or tie-ins attached. This has several implications.

- The AAF software developers kit (SDK) is available as a download via the website (www.aafassociation.org) to aid speedy AAF-based developments. Although many commercial companies are involved, the AAF Association is a nonprofit organization. Manufacturers may download the SDK for free and modify the source code if required. They may make products (called "larger works") and charge money for the result. These licensing terms are perpetual.
- AAF is not competing with anyone. It is open and anyone can join. Working with other formats and helping to support the user workflow is always an aim.
- AAF handles uncompressed video, audio and data. It is also compression agnostic; not knowing or caring about compression, it is equally well able to handle compressed material. It is format independent—the size or composition of the pictures is of no concern. However, it carries history information describing the details of compression and picture size.
- AAF is designed not only to facilitate interchange in post but also the workflow outside. It interfaces well with MXF, GXF and other formats to support the flow of essence and metadata from content creation through to playout and interactive services.
- It works cross-platform with any of the commonly used operating systems: Windows®, Mac OS®, IRIX® and Linux.

Editing and Authoring

Editing and authoring generally involve:

1) opening one or more source content files
2) manipulating or editing the content
3) saving the results
4) moving to a different application or platform to always use the best tools/people for the job.

Editing and authoring require the use of many applications. Being able to move from one vendor to another and retain metadata along the way is very powerful. It is vital to record and carry forward not only editing and scripting decisions, but also a record of the steps along the way. These may include information on the sources, equipment configuration, intermediate data and any alternatives that may be selected later.

Generally vendors' applications save the resulting 'compositional metadata' file in their own proprietary format (compositional metadata is metadata such as EDL information that tells how the various video and audio elements were combined). This 'closed' approach makes the use of multiple applications and later repurposing of content difficult, as the compositional metadata is not transferable. This blocks the workflow and metadata has to be repeatedly re-entered every time when moving between applications—costing time and money as well as adding room for error.

For interchange between open systems, applications have to know how to read the file wrapper header, how to unwrap the contents, and then how to interpret that content. This creates considerable challenges when sending a file from one vendor's system to another.

Here, the whole point of AAF becomes clear: that editors and authors should be able to use many different applications on the same composition. The content and the decisions made in one application should be visible in another. AAF's unified interchange model enables such interoperability, offering distinct advantages over traditional authoring models:

- Editing/authoring requires a wide range of applications to combine and modify essence. Although applications may have very different domains (e.g. audio editing and 3D graphics animation) all applications should work together to produce the final presentation.
- Applications can extract and display valuable information about the content in an AAF file (from its wrapper 'label') even if they do not understand the data format. This allows the user to better coordinate the authoring process.

There are many other issues related to completely transparent interoperability. The significant benefit of AAF is assurance that compositions output by AAF-compliant applications will be accessible by the right tool for the job, without risk of being 'stranded' by proprietary file format restrictions.

Applications that can use AAF for interchange include:

- Post production systems, including digitization, off-line editing, graphics, compositing, and rendering systems
- Image manipulation applications, including palletizing tools
- Audio production/engineering systems, including multitrack mixers and samplers
- 3D rendering systems
- Multimedia content creation systems, including scripting, cataloging, titling, logging and content repackaging applications
- Image and sound recording equipment, including cameras and camcorders, scanners, telecines, sound dubbers, disk recorders, and data recorders
- Content Management Systems, Digital Asset Management Systems and other content and asset tracking business applications
- Television studio systems, including picture and sound editors, servers, effects processors, archiving and broadcast automation systems

Interchange

AAF allows interchange of a broad range of essence formats and metadata. Even so, applications may have interchange restrictions due to other considerations. So, there are different kinds of interchange possible:

- Interchange of a limited set of essence
- Interchange of a broad set of essence with some related metadata
- Interchange of essence and a rich set of metadata, including compositions but having limited support for some essence types
- Full interchange of all essence types and all metadata described in the AAF specification, preserving any additional private information stored in the AAF file

AAF incorporates existing multimedia data types such as video, audio, still image, text, and graphics. Applications can store application-specific data in an AAF file, and can use AAF as the application's native file format. The AAF SDK has codecs built in for some commonly used formats, such as CDCI and RGBA images, WAV and AIFC audio and provides an extension framework for any new or proprietary formats. This allows the SDK to view the essence content of a file. It is important to note that AAF can encapsulate *any* essence data type, but seeing or hearing it via the SDK may require adding a codec. In any case, AAF can be used to transfer *any* essence stream without restriction.

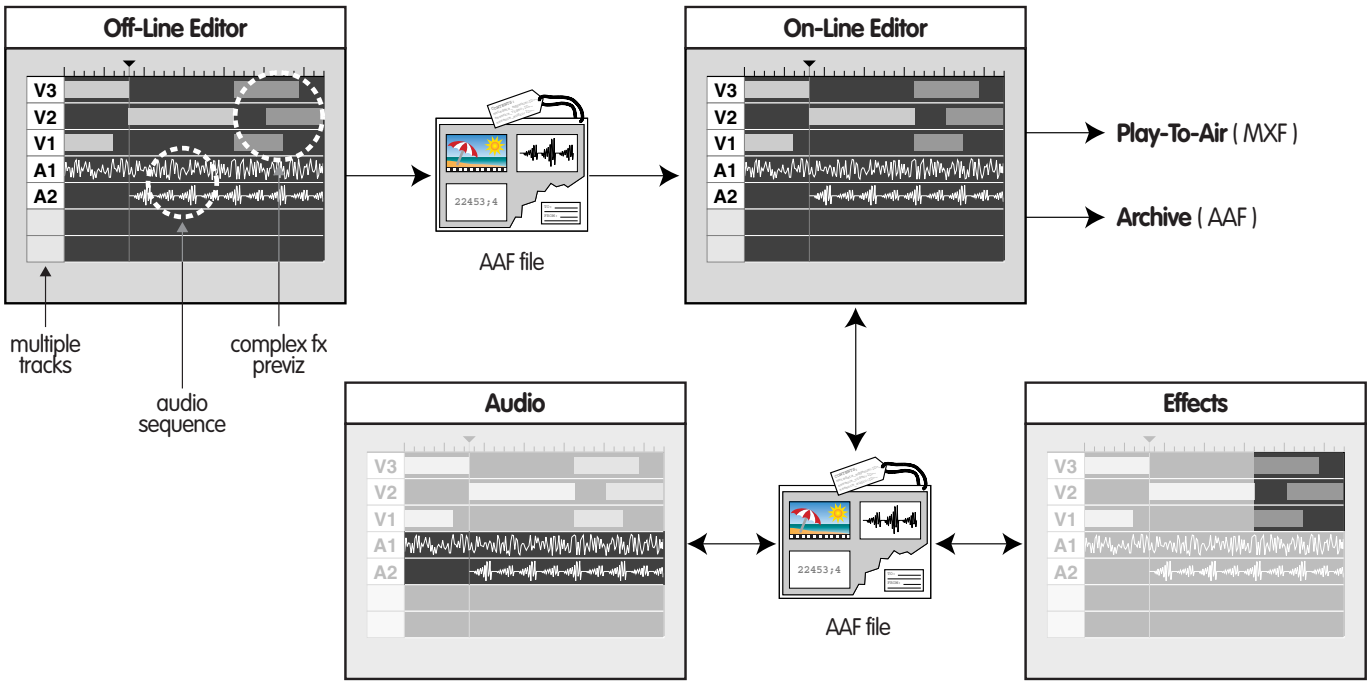
Compositional Information

AAF allows users to move this information along with essence, formatting and storing the information in a standardized way.

Media Derivation

AAF can describe the process by which several separate pieces of video and audio were combined to produce another. AAF can track the origin of all of the original components (sometimes called History Information, or derivation chain). For example, when an AAF file contains audio and video originated from film, it may contain descriptive information about the film source, including edgecode, and in- and out-points from an intermediate videotape. This is very useful when repurposing the material for a new cut or production.

Derivation information can also describe the creation of computer-generated material. If a composition was generated from a 3D animation and still images, the AAF file can contain the information needed to go back to the original sources and make changes without having to regenerate the entire composition.



AAF enhances workflow by allowing specific parts of a composition to be sent to an appropriate application for treatment and then gathered back into the whole.

Flexibility and Efficiency

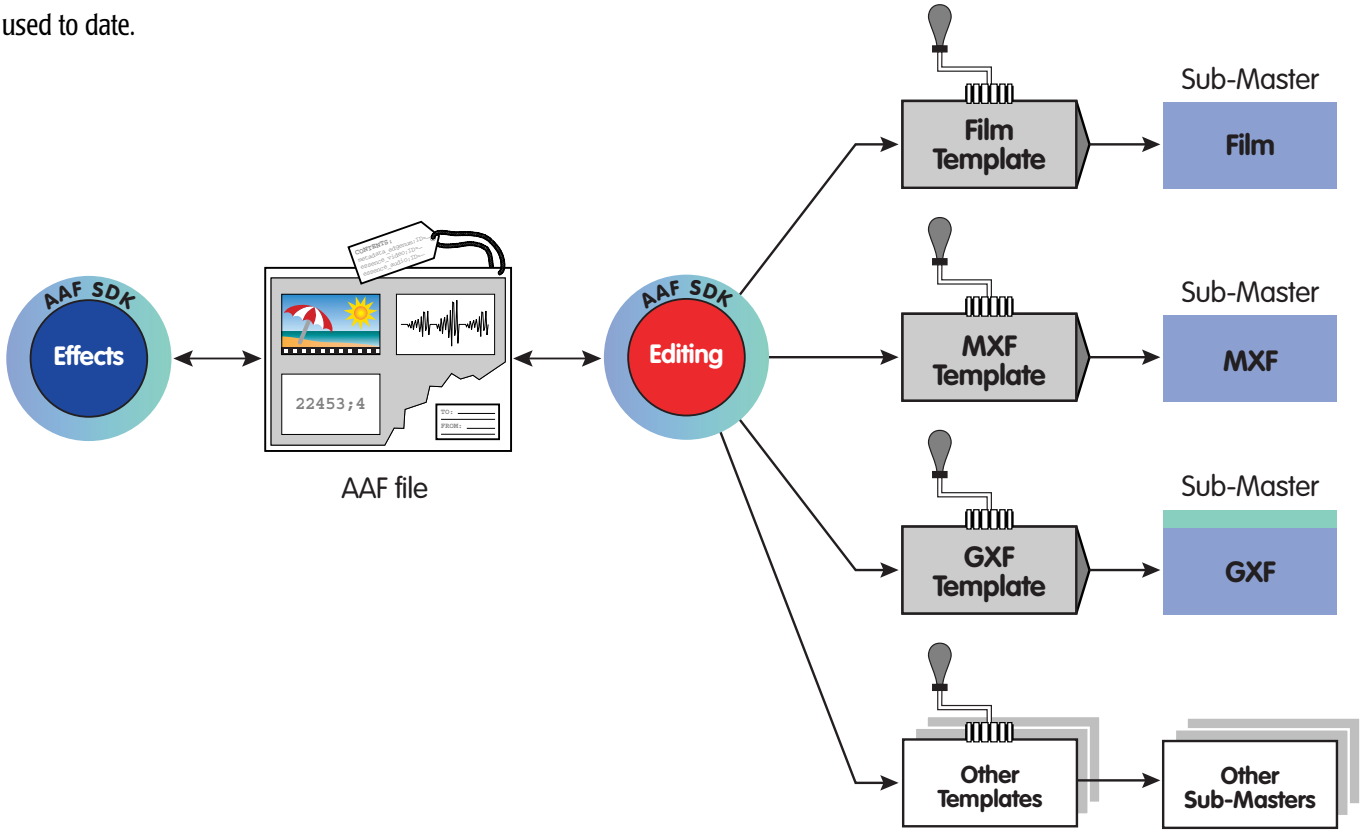
AAF is not designed as a streaming media format but as a file interchange format. It can also be used for native capture and playback of essence with flexible storage for large data objects. For example, it allows sections of data to be broken into pieces for efficient storage and offers references to externally stored essence. AAF also allows in-place editing; where the metadata changes but the entire file is not rewritten (See "AAF File Format").

Templates

It is AAF's templates that provide filters to control which parts of the essence and metadata are passed on to other applications. For example, there are typically different requirements between distribution and an archive repository. The former needs the conformed (flattened) essence with only basic metadata (e.g. program title), the latter may require any or all the history of material origination, ownership and detail of editing processes used to date.

Extensibility

AAF defines mechanisms for extending the storage of metadata and essence so it can include new types as they come into common use. Effects extensibility allows tool vendors to develop a rich library of new and engaging effects, or processes, for use with AAF files. When an effect or codec is referenced in an AAF file, binary plug-in modules allow AAF-compliant applications to determine if that facility is available and, if not, to find it and load it on demand.



AAF files can be distilled down to a distribution or other formats using templates which, to varying degrees, filter out superfluous metadata and 'flatten' essence. AAF templates allow extensibility as the introduction of a new format to the AAF specification is accomplished by simply writing and publishing a corresponding template.

AAF File Format

AAF is a wrapper for essence and metadata providing a single model for the interchange of data. Essence types may include video, audio, still images, graphics, text, MIDI files and animation. Metadata types may include compositional information, event triggers, timecode/edgecode and references to original ownership and past processes. AAF files preserve their original and existing metadata, as well as adding new authoring/editing information. AAF supports efficient playback and incremental updates and is scaleable for both high-end professional and consumer-level applications. The AAF file format includes:

- Information about the original sources so that edited essence can be traced back to originals
- References to external essence files, including files located on remote networks
- An extensible video and audio effects architecture with a rich set of built-in base-level effects
- Open, cross-platform support with the SDK working with any of the commonly used operating systems, Windows®, Mac OS®, IRIX® and Linux.

Full authoring information

AAF has very rich metadata capabilities. It can describe complex edits, compositing, effects and other functions used in post production. It can contain a finished program but, more usually, it holds all the source elements to render a finished program.

External references

A very powerful element of AAF is its ability to carry external references. For example, an AAF file contains three audio clips, two video clips and the "EDL" in metadata form, instructing what to do with the clips to create the edited result. In addition, this AAF file might contain a reference to a closed-caption file resident in a separate system. So, the AAF file can contain all the information about a post production project, regardless of how large the project is or where project elements are stored.

Easily editable

AAF's use of Structured Storage supports its full feature capability. This means that the alterations and additions involved in the authoring/editing process are easily and quickly added to the file without having to entirely rewrite it (similar to FAT works, Structured Storage files can be expanded without having to be contiguous).

Compression agnostic

AAF is compression agnostic. It can be used with a number of compression formats as well as supporting the uncompressed operation which is much favored in post production.

Note: AAF is targeted at the specific needs of authoring and post. It is less suitable for store-and-forward or broadcast playout, where other formats such as GXF and MXF are better suited to these applications (see "Program Delivery").

Essence Issues

Today, editing and authoring often involve the use of several types of content together while managing interactions and relationships between them. The content types are generally motion picture film/video, audio, still images, animation, 3D geometry and text. The sheer number of content file formats, each with its own strength (preferred compression codec, optimized file size, preferred color resolution, etc.), requires many format-to-format conversions to produce an end product.

AAF's strengths are in the interchange of essence and metadata and working cross-platform. This allows the users' creative energies to be focused on their prime tasks rather than on interchange issues. It also means software development can focus on improvements to the authoring application's feature set.

Program Delivery

In contrast to authoring systems, delivery systems are used to transport *finished* programs. Technically, delivery has at least two major considerations: the target playback hardware (TV, audio equipment, PC) and the distribution vehicle (film, broadcast TV, DVD, network, etc.).

The content created using AAF in authoring will be delivered by many different vehicles using data formats such as baseband video, MPEG-2 Transport Stream, QuickTime 4, Advanced Streaming Format (ASF), General eXchange Format (GXF), and the Material eXchange Format (MXF). These do not need AAF's rich set of metadata which was useful during authoring. So, by conforming and saving the content without the authoring metadata—stripping out the metadata to 'flatten' the file—AAF optimizes completed compositions for delivery.

About MXF

The Media Exchange Format, MXF, is supported by the many members of the Pro-MPEG Forum (www.pro-mpeg.org) and is aimed at the exchange of program material between file servers. It is also a format for tape streamers and digital archives. It usually contains one complete sequence but this may comprise a sequence of clips and program segments. There are six operational patterns: Simple, Compiled, Compound, Uncompiled Simple, Uncompiled Compound and Metadata-only.

As MXF is derived from the AAF data model it integrates closely with AAF files as well as stream formats. Bridging file and streaming transfers MXF helps move material between AAF file-based post production and streaming program replay using standard networks. This setup extends the reliable essence and metadata pathways of both formats to reach from content creation to playout.

The MXF body carries the content. It can include MPEG, DV and uncompressed video and contains an interleaved sequence of picture frames, each with audio and data essence plus frame-based metadata. MXF has been submitted to the SMPTE as a proposed standard and is fully SMPTE KLV compliant.

About GXF

The Grass Valley Group (www.grassvalleygroup.com) designed the General eXchange Format (GXF) to support on-air and news operations and other related broadcast applications. It is based on an extensible content (essence) encoding concept similar to, but not identical to, the SMPTE's KLV scheme (a method for wrapping data for transport over networks) used by AAF. GXF is used in many broadcast facilities for applications including the transfer of material on data networks and archiving onto data tape and other storage devices.

GVG is examining requirements for the transfer of a limited set of metadata from the authoring system to the distribution environment. Some metadata may be transferred from the authoring process but additional new information, which relates specifically to the finished program, would be inserted when the finished program is published. Additionally GVG is looking at requirements for a driver that would allow authors to publish GXF streams from AAF compliant applications.

GXF supports cuts-only video edits along with audio fade in/out but not the rich effects or editing of complex packages. It fits very well with on-air applications with support for MPEG (elementary streams), DVCPRO, JPEG video, uncompressed AC3 and Dolby E audio. At least seven vendors support GXF today.

Developers – Getting Started

Downloading the SDK

The AAF Software Development Kit is available for download from Source Forge (**http://www.sourceforge.com**). Developers must register at Source Forge to access the AAF project site, and be familiar with CVS.

Please refer to the Source Forge site documentation for information regarding CVS. An all platforms CVS client is available at **http://www.cvshome.org**.

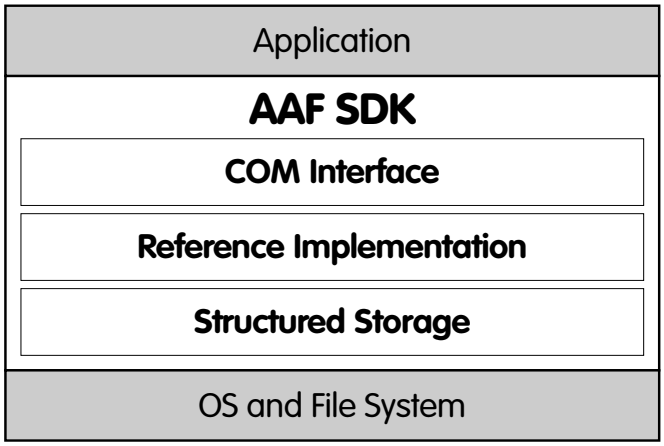
One you have a Source Forge login, and a CVS client, the following command will download the most recent version of the SDK:

```
cvs -z3 -d:pserver:anonymous@cvs.aaf.
sourceforge.net:/cvsroot/aaf co AAF
```

Follow the build instructions in **AAF\doc\AAFProjectFAQ.html**.

Software Architecture

The SDK is composed of three primary components: a COM interface, a reference implementation of the AAF object model, and Microsoft's structured storage as the persistent object store. This is illustrated below:



Application developers need only be concerned with the COM api. The api is documented in the source code (see the `AAF.idl` file) and also at: **http://aaf.sourceforge.net/docs/com-api/contentsf.html**. Developers should also review the AAF Developers' Guide, and the AAF Specification. Both are available at **http://www.aafassociation.org**. These documents, the specification in particular, describe the AAF object model. The COM interface is the only means which developers access the objects described in the specification.

The SDK includes a minimal implementation of COM for use on non-Microsoft platforms. Structured Storage binaries are also included for Irix, Linux, Macintosh, and, of course, Windows 2000 (NT). There is no Structured Storage source code in the SDK.

First Program

The following is a walk through of a very simple AAF program. Several important details are omitted for the sake of clarity. Notable, there is not error checking, and the COM interfaces are not correctly released.

First, we need several include files. `AAF.h` declares the COM interface, `AAFStoredObjectIDs.h` declares unique identifiers for AAF stored objects. This file is required to create new object instances.

```
#include <AAF.h>
#include <AAFStoredObjectIDs.h>
```

Before accessing any COM interfaces, the AFF COM module must be loaded:

```
int main( int argc, char** argv )
{
  // Load the default COM implementation
  // library. Passing null will cause the
  // default module to be loaded.
  AAFLoad( "AAFCOAPI.dll" );
```

Next, a file needs to be opened or created. Here we create a new file. All new files require a product identification:

```
  // A product identification structure is
  // required to create a new file. This
  // includes a product UUID. Need a UUID?
  // Try Microsoft's UUIDGEN tool. Here
  // a newly generated UUID to use:
  // 89aa595e-51ec-4e65-8ce8-37818def78f3

  aafProductIdentification_t ident =
  {  L"AAF Association",  // Company Name
    L"First File",        // Product Name,
    L"2.71828182818",     // Product Version
    {0x89aa595e, 0x51ec, 0x8ce8,
    {0x8c, 0xe8, 0x37, 0x81, 0x8d, 0xef,
     0x78, 0xf3}}         // Product UUID
  };

  // Create a new file.
  IAAFFile* file;
  AAFFileOpenNewModify( L"./my_first_aaf_file.aaf",
                        0, &ident, &file );
```

`AAFFileOpenNewModify` returns a pointer to a COM interface. All objects in an AAF file are contained by the header, hence, the first object we get an interface for is the header. The dictionary is also required for this example, so we get an interface for that object as well:

```
  // We will need the header.
  IAAFHeader* header;
  file->GetHeader( &header );

  // The dictionary is required to create a
  // new object.
  IAAFDictionary* dictionary;
  header->GetDictionary( &dictionary );
```

Next, we want to create an instance of a new MasterMob object. We can use the dictionary to do this. We must provide the unique ID of the object we wish to create, and the interface we wish to create. All objects must be initialized after being created, and before being used:

```
  // Now, use the dictionary to create a
  // MasterMob and initialize it.  You must
  // always initialize a new created object.
  IAAFMasterMob* masterMob;
  dictionary->CreateInstance(
     AUID_AAFMasterMob, // The object ID
     IID_IAAFMasterMob, // The interface ID,
     reinterpret_cast<IUnknown**>(&masterMob)
);
  masterMob->Initialize();
```

We want to name the mob. Look at the documentation for `IAAFMasterMob`. You will not find a "`SetName`" method. However, you will find that COM objects that implement `IAAFMasterMob` also implement `IAAFMob`. We must use the standard COM "`QueryInterface()`" method to access the `IAAFMob` interface:

```
  // Name this mob "My First Mob"
  // But wait!  We IAAFMasterMob has no
  // method to set the mob name.  To do that
  // we require an IAAFMob interface.
  // Objects that implement IAAFMasterMob
  // also implement IAAFMob.
  IAAFMob* mob;
  masterMob->QueryInterface( IID_IAAFMob,
 reinterpret_cast<void**>(&mob) );
  mob->SetName( L"First Mob" );
```

The new MasterMob object will not be saved until we add it to the header, and save the file:

```
  // Add mob the file (via the header)
  header->AddMob( mob );

  // Save and close the file.
  file->Save();
  file->Close();
```

Finally, it is good practice to unload the COM module when finished:

```
  // Unload the COM module.
  AAFUnload();

  return 0;
}
```

That's it. That will create an AAF file with a single MasterMob named "First Mob". Here is a dump of the file to illustrate this:

```
Object    Header
Prop      ByteOrder
Value       IAAFTypeDefInt: (Int16) 18761 0x4949
Prop      LastModified
Value       IAAFTypeDefRecord: 2 members
Prop      Content
Value       IAAFTypeDefStrongObjRef:  to object of
            class ContentStorage
Object       ContentStorage
Prop         Mobs
Value          IAAFTypeDefSet:
Value            IAAFTypeDefStrongObjRef:  to
                 object of class MasterMob
Object            MasterMob
Prop              MobID
Value               IAAFTypeDefRecord: 6 members
Prop              Name
Value               IAAFTypeDefString: First Mob
Prop              Slots
Value               IAAFTypeDefVariableArray: 0
                    elements of type
                    IAAFTypeDefStrongObjRef
Prop              LastModified
Value               IAAFTypeDefRecord: 2 members
Prop              CreationTime
Value               IAAFTypeDefRecord: 2 members
Prop      Dictionary
Prop      Version
Prop      IdentificationList
```

This dump shows the objects in the file, the properties of each object, and the value of each property (Dictionary, Version, and IdentificationList excluded). Our new Master Mob is the first, and only, Mob object of the Mobs set contained by the ContentStorage object.

Example Programs

The SDK includes a number of example programs. These are a set of examples that came into existence during the development of the SDK itself. To complement these examples, the AAF Association is producing a set of new example programs. This new code aims to provide a set of examples that are mildly representative of the type of processing that must be performed by real applications. They are built upon a small library of reusable C++ classes that are implemented using modern C++ coding practices. They also hide many of the distracting details of accessing AAF objects via multiple COM interfaces.

The new examples walk a developer through the process of creating a file with metadata only, metadata plus audio and video essence, and finally an example that builds a composition from these objects. Code fragments would not be meaningful in the narrow context of this document, however, it is informative to examine a dump of the full composition created by the example code.

The dump shows all the objects types (and their containment relationships) required to build a composition. The CompositionMob ties it all together to create a multi-track audio/video composition with a transition effect. Understanding how to create, or simply read, even a simple file such as this requires careful study of the AAF Object Specification, study of example code, and plenty of experimentation and practice with the SDK.

The examples programs also include the dump utility used to generate the output presented here. The dump utility itself is a good example of how to implement processing operations that must pass over all objects in an AAF file.

Still, these examples only scratch the surface of what can be accomplished with the SDK. Expect to see more.

This new example code is not available on SourceForge, but is available to AAF members. Please contact the AAF Organization to access the code.

Example Composition Dump

```
Object    Header
Object       ContentStorage
Object            SourceMob
Object                TimelineMobSlot
Object                    SourceClip
Object               CDCIDescriptor
Object            MasterMob
Object                TimelineMobSlot
Object                    SourceClip
Object                TaggedValue
Object                TaggedValue
Object                KLVData
Object                KLVData
.
. (three more MasterMobs)
.

Object            SourceMob
Object                TimelineMobSlot
Object                    SourceClip
Object               WAVEDescriptor
Object            SourceMob
Object                TimelineMobSlot
Object                    SourceClip
Object               CDCIDescriptor
.
. (one more of each of the above SourceMobs)
.
Object            CompositionMob
Object                TimelineMobSlot
Object                    Sequence
Object                        SourceClip
Object                        Transition
Object                           OperationGroup
Object                        SourceClip
Object                TimelineMobSlot
Object                    Sequence
Object                        SourceClip
Object                        Transition
Object                           OperationGroup
Object                        SourceClip
Object            EssenceData
Object            EssenceData
Object            EssenceData
Object            EssenceData
```

Membership

Principal Members

- Avid
- British Broadcasting Corporation
- Cable News Network
- Discreet
- Fox News Corp.
- Liberty Livewire
- Microsoft
- National Imagery and Mapping Agency
- Panasonic
- Pinnacle
- Quantel
- Sony
- Turner Entertainment Networks

General Members

- AIST—Animated Image Systems Technology GmbH
- Ascential Software
- EMC
- Encoda Systems
- Grass Valley Group
- Leitch
- Matrox
- NL Technology, LLC
- Omneon
- Pandora
- Philips
- Post Impressions
- Smoke & Mirrors
- Snell & Wilcox
- Sonic Foundry
- tecmath
- Warner Bros.

Associate Members

- Bulldog
- eMotion
- Front Porch Digital
- Dutch Broadcasting Services Corporation (NOB)

Supporters

- ☐ International Digital Cinema Festival

The AAF Association

Incorporated in January 2000, the AAF Association, Inc. is a broadly-based trade association promoting the development and adoption of AAF technology throughout the media industry. With membership embracing many major players in the industry, it intends to help deliver the full benefits of digital media to content creators including those working in film, television, Internet and post production. Membership in the AAF Association is open at various levels to any interested parties. To learn more, please contact us at: info@aafassociation.org

Comments from some members of the AAF Association

"As post production moves away from its reliance on D1 tape and towards networked file systems containing digitised multi format video and film, there is a great need for common interchange and archiving formats supported by all manufacturers. The AAF provides the common framework which we hope all manufacturers will adopt to allow for the interchange of media within the post production process."

John Bennett
Head of Software development,
MovingPictureCompany, London

"AAF provides a way for us to move metadata from one system to another without re-keying. The beauty of AAF is that the appropriate metadata is joined at the hip with the media though its workflow and life cycle.

"Our future purchase decisions will be based on AAF compatibility. In other words, we are more likely to buy products that include AAF. It is worth the effort of manufacturers to devote resources to this worthwhile initiative."

Suzanne Donino
Senior Vice President
Turner Broadcasting Systems, Inc.

"As a leader in the communications industry, Liberty Livewire is a highly integrated media company, offering a wide range of services in the production, post production, audio, and broadcast market segments. Livewire seeks to integrate these services by optimizing disparate methodologies and streamlining the interchange of metadata and program content.

"Many of our current business processes require onerous reformatting or re-keying of data as content is manufactured and distributed. File and metadata interchange among production and post production systems is difficult, time-consuming, and unduly expensive. This inefficiency costs Liberty and its clients both time and money.

"As a founding member of the AAF Association, Liberty Livewire supports the work of the AAF in establishing a common file exchange format for the production and post production environments. We look forward to the time when AAF is supported in products, and encourage all manufacturers and users to embrace the AAF open source interchange format. We anticipate that AAF will enhance the creative experience by simplifying interchange, enabling editors, designers, and others to do their best work."

Gavin Schutz
Executive Vice-President,
Chief Technology Officer, Liberty Livewire

"The BBC is committed to improving the links between different stages of the programme-making process. With solid industrial backing and the AAF Association to guide its development, the AAF format is on course to deliver the dream of seamless transfer of content between systems."

Andrew Oliphant
BBC Research and Development

"CNN/Turner Broadcasting needs standards to describe complex media and enable interchange between different systems. We look to the AAF Association to move these efforts forward. Without this work, vendors are faced with developing custom integration that is costly, slow, and not extensible."

Gordon Castle
VP Research and Development, CNN

"The Fox Group, in all aspects of its enveavors, seeks to produce content using the latest technology and techniques, which are, at the same time, cost-effective. To that end, we feel that a common media file format, both for storage, transmission and archiving, is highly desirable and would meet that goal.

"AAF and MXF hold the promise of a common file interchange format, tailored to both file transfer and file streaming. Rich in metadata, these formats will allow Fox to produce highly sophisticated projects using multiple platforms without the inefficiency of error prone "cross conversions". Further, we will be able to more easily mine our archive in the future for use in other venues and markets as well as for re-use within new projects.

"We encourage the work undertaken by AAF and the Pro-MPEG forum in making these formats open source and available to multiple vendors."

Andy Setos
Executive Vice President,
News Technology Group, The Fox Group

"The AAF Association is delivering on its promise of enabling interoperability for the post-production environment. The release of AAF SDK v1.0 in April 2001 gives manufacturers the practical means of incorporating the format in their products, and we encourage them to do so. In the year ahead, we look forward to working with manufacturers on implementing and using AAF-based systems and with the AAF Association on enhancing the AAF technology yet further."

Phil Tudor
BBC Research & Development