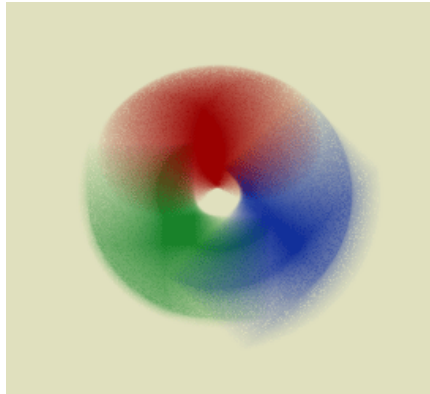


# MXFComponentSuite



White Paper

May 2004

© MOG Solutions

This information is subject to change without prior notice.

## Overview

This document presents the **MXFComponentSuite**, a suite of components built on top of the **MXF::SDK**. This SDK is being developed in a partnership between IRT and MOG Solutions.



IRT (Institut für Rundfunktechnik) is a private company acting as the central research and development establishment for the public broadcasting organizations in Germany (ARD, ZDF, and DLR), in Austria (ORF), and in Switzerland (SRG/SSR). The main focus of IRT's work is the improvement of existing and development of new digital systems and their standardization.

IRT is actively involved in standardization activities, including the development of the MXF SMPTE standard, therefore providing the needed expertise to verify **MXF::SDK** standard compliance. IRT is also a leading organization in equipment compliance testing, and is committed to use **MXF::SDK** in this activity.

The SDK provides a full object-oriented model of the MXF structure providing a detailed programming interface to every MXF mechanism. The C++ interface of the SDK is designed to be simple to use and comes with sample code and full documentation.

The **MXFComponentSuite** is a set of MXF-enabled components that abstract the application layer from the details and complexity of the MXF format, using simple API's. There will be many cases when the **MXFComponentSuite** will just be all that you need to make your products MXF aware, ready for the exciting new business opportunities that the IT world provides to broadcasting.

When should the **MXFComponentSuite** be used?

- Maybe C++ is not the best interface for you, perhaps you are developing in a different language or you don't have the resources with object-oriented programming skills to work with this kind of interface;
- Maybe your products don't really need that much of a detailed access to MXF mechanisms;

- Maybe you just need to wrap some Metadata into a file along with the Essence; or the reverse, to extract that information from an MXF file you receive;
- Maybe you just want to generate an MXF file from a regular EDL;
- Or maybe you just want to play the Essence stored in the file, or just get some technical metadata from that Essence (compression scheme, duration...).

In any of these cases, the **MXFComponentSuite** is ideal for you!

This suite of ActiveX Components can be easily integrated into your products using any of the most popular IDE's<sup>1</sup> in the market. Your developers may be using a variety of languages (Basic, Delphi, JAVA...) and don't even have to know what C++ is.

The MXF Components have a simplified programming interface designed to specifically target the most common MXF use cases, and in a way that allows your development team to quickly bring your products to the MXF arena.

You can also have an active role in the development of the **MXFComponentSuite**. MOG Solutions values the customer feedback, and your suggestions are always welcome to help us refine Component interfaces and even to define new Components.

---

<sup>1</sup> IDE - Integrated Development Environment; examples are Microsoft Visual Studio® (C++, Basic, C#), Borland C++ Builder®, Borland JBuilder®, Delphi® and Sun ONE®, among others.

## Product Roadmap

MXF is all about tight integration of Essence and Metadata. The easiest way to achieve this?

### MXFWrapper

With **MXFWrapper**, Essence and Metadata files go in and an MXF file comes out!

Metadata is represented in XML: the preferred and most flexible way to represent Metadata nowadays.



**Fig. 1 - Wrapping and unwrapping MXF files**

What about the reverse?

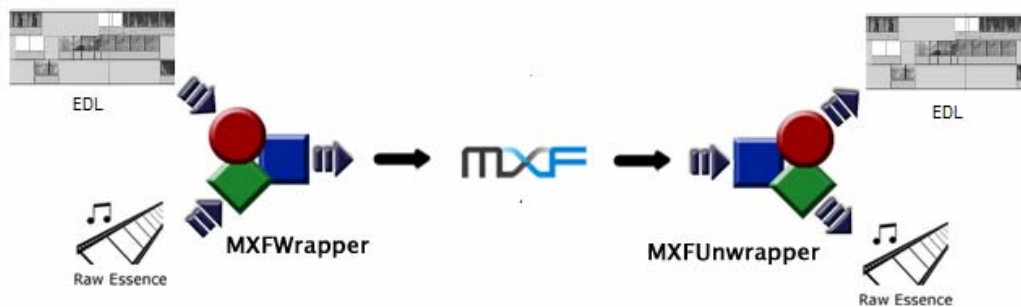
### MXFUnwrapper

Feed an MXF file to this Component and you get back the Essence enclosed in the MXF file along with all Metadata.

The Metadata will again be in XML format, ready to be used by your product or even integrated with other XML enabled products, like Media Asset Management systems, for example.

What about support for Editing Lists (EDLs)? The Essence may be organized according to a timeline inside the MXF file. The MXF Structural Metadata already captures this. However, some equipment might expect this kind of information in other formats, such as a standard EDL format like CMX or AAF. How can you easily generate an MXF file directly from these formats? Or the reverse, that is, how to get an EDL representing the contents of an MXF file?

**MXFWrapper** and **MXFUnwrapper** provide the solution. These components can optionally take and generate, respectively, different EDL formats. You can now bring legacy MXF unaware equipment into the MXF age!

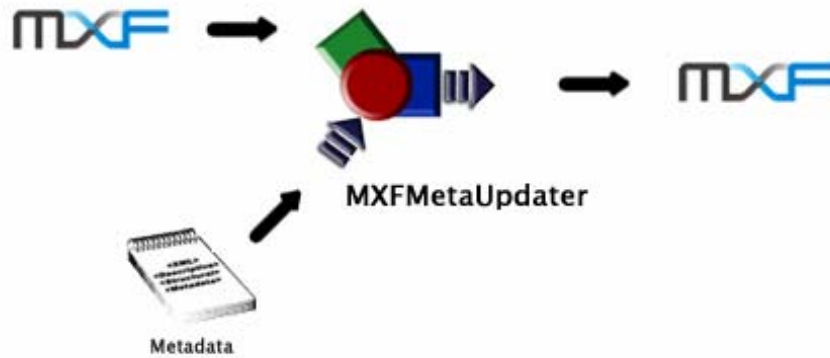


**Fig. 2 - Wrapping and unwrapping MXF files using EDLs**

Let's now assume you already have an MXF file and you want to update the Metadata already in the file. Do you need to "unwrap" and then "wrap" it again? This could take some time depending on the size of the file... Once again **MXFComponentSuite** has the answer:

## MXFMetaUpdater

You just have to specify the MXF file to be updated and the XML file with the new Metadata to be loaded.

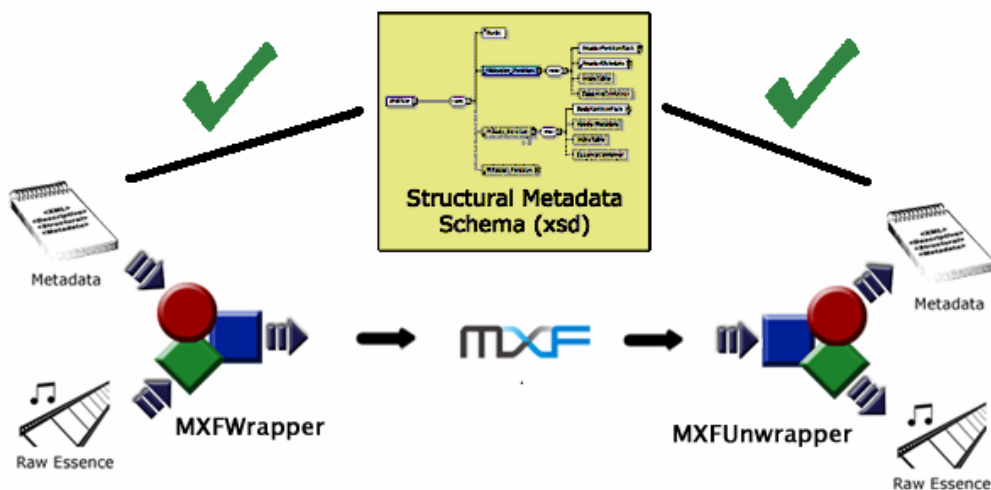


**Fig. 3 - MXFMetaUpdater**

**MXFMetaUpdater** uses mechanisms defined in the MXF standard to update the Metadata in an extremely efficient way.

XML is so flexible that one could even argue it is too flexible. You could write the same information in XML in an infinite number of ways. How will you ever guess in which of these ways do the Components accept and output the metadata? The answer is: you don't need to guess!

The **MXFComponentSuite** uses the latest in schema technology: XML Schema.



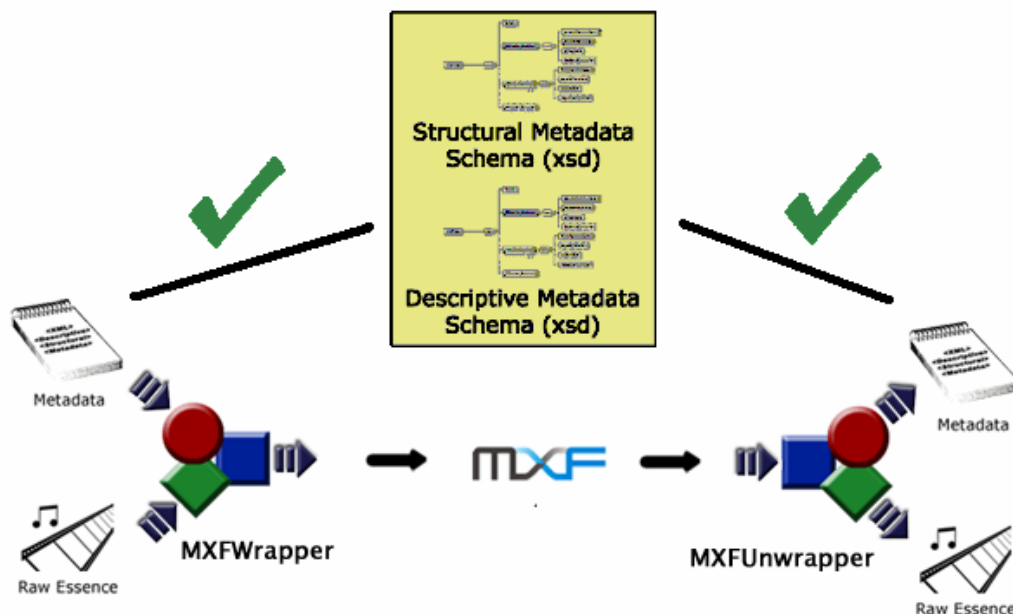
**Fig. 4 – Schema validation**

The exact schema used in the Suite's XML documents is defined using XML Schema, adding document structure information and also type safety to your metadata handling.

How about Descriptive Metadata? Of course there is a data model being developed (DMS-1), but MXF is data model agnostic which means you can plug in any Data Model you want into the MXF Metadata Structure!

With **MXFComponentSuite**, you just need to specify the XML Schema document that formally describes your specific data model and feed it to the **MXFWrapper** and **MXFUnwrapper** Components. That's it!

You can now feed with your Data Model specific Descriptive Metadata to **MXFWrapper** or read it from existing MXF files with **MXFUnwrapper**.



**Fig. 5 – Descriptive metadata**

But formally defining a Data Model can be quite demanding, especially when you try to get the most out of the huge amount of XML Schema features, and the **MXFComponentSuite** definitely does!

Maybe you want your costumers to have the flexibility to define their own Data Models. In that case, just plug into your products the

## **MXF DMS Editor**

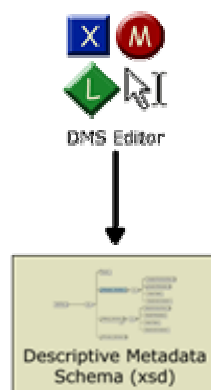
This component, when integrated into your products, will provide a graphical user interface that allows your customers to define their own data models, bringing your products to a level of flexibility that will for sure meet even their most demanding requirements.

The output of this component is a XML Schema document that can be readily fed into the **MXFWrapper** and the **MXFUnwrapper**, therefore leveraging the easy integration of the MXF Components into your product development.

Of course using **DMS Editor** assumes your client already has a pretty good idea of the kind of Metadata he wants to include in the MXF files. That is not always the case though.

More and more, organizations are becoming aware of the need to carefully analyse their workflow and streamline their internal processes. In the definition of an architecture that meets their requirements, Metadata plays a crucial role, and the definition of a Data Model can be quite complex.

In MOG Solutions you will find a valuable partner, with an experienced team, that can help you advise your clients on the best practises towards Essence and Metadata integration throughout their workflow, with MXF as the base framework.



**Fig. 6 – DMS Editor**

What about the more technical Metadata related to the Essence? How does it fit in this Suite? After all, when you feed Essence into the

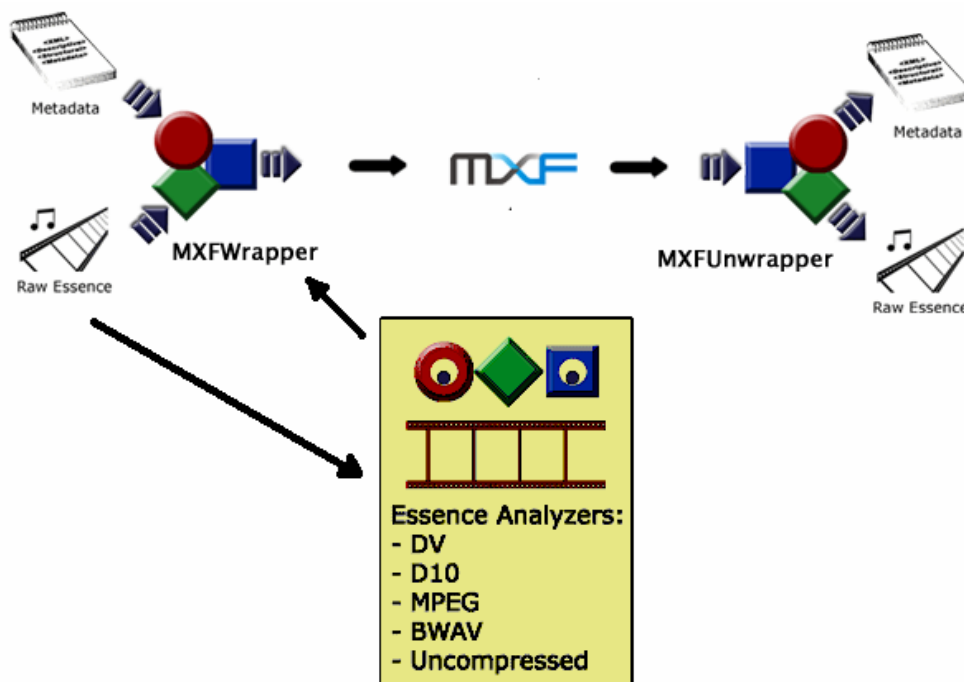
**MXFWrapper**, ultimately, technical Metadata about the Essence must also end up in the MXF file.

And the reverse operation is also required. When you receive an MXF file, you want to read technical Metadata about the Essence without having to extract and parse it.

For this purpose, MXFComponentSuite equips you with a set of

## Essence Analysers

Each Essence Analyser is able to analyse a specific type of Essence (Uncompressed, DV, D10, MPEG PS, MPEG 1/2 Video, MPEG 1/2 Audio, BWAV...) and outputs an XML document with all the technical Metadata.



**Fig. 7 – Essence analysers**

Once again, the output of any **Essence Analyser** can be readily fed into **MXFWrapper** to add Essence technical Metadata to your MXF files.

What if your product just needs to access some technical information from raw Essence files to display on the user's desktop? Maybe you don't want to spend your resources analysing all the bits in a variety of Essence formats to figure out compression scheme parameters, aspect ratios, pixel layout...

If that's your case, just plug in your choice of Essence Analysers since these can be used standalone, just as any other Component in the Suite.

So now you can generate MXF files with Essence and Metadata and you can extract these from existent files. But what if you just want to play the file? After all, an MXF file includes timeline information and you might just want your clients to be able to play it on their desktop.

Microsoft Direct Show<sup>®</sup> technology provides all mechanisms to enable you to do this EXCEPT for MXF decoding. No problem. MXFComponentSuite includes the

## MXF DirectShow<sup>®</sup> Filter

You just register this filter on Direct Show<sup>®</sup> and your product can now directly play MXF files on the desktop with no additional effort from your development team!



Fig. 8 – MXFDirectShow Filter

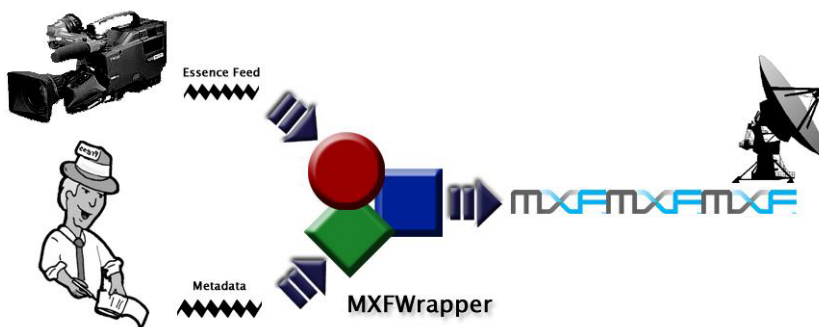
## Streaming

The **MXFComponentSuite** model presented so far is file based, that is, you specify file names to the Components as inputs and these produce files as output (except for the **MXFDirectShow Filter**).

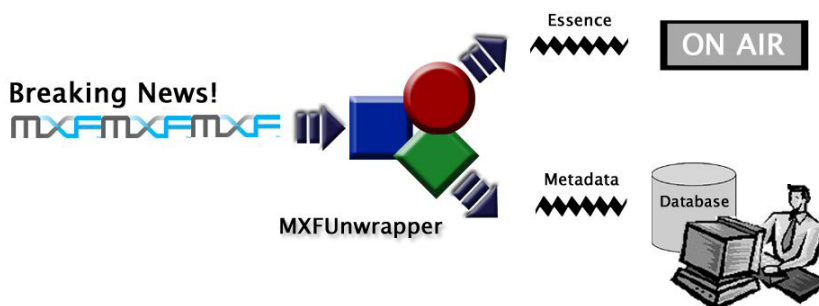
However, some use cases require a more dynamic approach to MXF. In a streaming environment, you want to “unwrap” an MXF file as it arrives, rather than having to wait for the all file to arrive to start processing it.

Once again the **MXFComponentSuite** holds the solution for this. For some of the Components in the Suite, both ActiveX and C++ library versions are shipped. While the ActiveX controls follow a file based model, the C++ version of the **MXFWrapper** and the **MXFUnwrapper** allow you to use either a file based model or a streaming model.

This way, you can easily use these simple C++ interfaces to do all your MXF “wrapping”...



... and “unwrapping”...



... on-the-fly!

## Data Sheets

The following data sheets describe the inputs and outputs of the MXF Components in their ActiveX version. The C++ library versions offer the same functionality but add to it the streaming interfaces mentioned before.

For a detailed description of the C++ interfaces please refer to the API documentation shipped with the **MXFComponentSuite**.

### MXFWrapper

Generates and MXF file from raw Essence and Metadata, which may include Descriptive Metadata according to any Data Model.

#### Type

Active X Component

Automation Interface	<b>Yes</b>
Graphic User Interface	<b>No</b>

#### Inputs

MXFSchema.xsd	<p>MXF Structural Metadata schema in XML Schema format. This document is supplied by MOG Solutions as part of the <b>MXFComponentSuite</b>.</p> <p>This document is used by the Component to validate the Metadata fed to it in XML format, to interpret it and tag it with the correct keys to be written to the MXF file in KLV format.</p> <p><b>The component also allows as input a binary version of the schema for improved performance.</b></p>
DMS-x.xsd documents	<p>XML Schema documents that formally describe the structure (scheme) of the Descriptive Metadata.</p> <p>These documents are used by the Component to validate and interpret the Descriptive Metadata fed to it.</p> <ul style="list-style-type: none"> <li>• One of these documents may be the DMS-1.xsd document that formally specifies DMS-1 Descriptive Metadata Schema as defined in the MXF Specification.</li> <li>• Other customized Descriptive Metadata Schemes (DMS's) may be fed, including proprietary DMS's, as long as these make proper use of Descriptive Metadata hooks as defined in the MXF Specification.</li> </ul>

	<ul style="list-style-type: none"> <li>Any DMS-x.xsd document must follow the same design principles followed in MXFSchema.xsd. Please refer to [1] for information on schema design guidelines.</li> </ul> <p><b>The component also allows as input a binary version of the schema for improved performance.</b></p>
metadata.xml	<p>XML Document with the actual Metadata to be included in the MXF file</p> <ul style="list-style-type: none"> <li>The document shall include the MXF Structural Metadata that describes the structure of the file. This XML document shall comply with the schema defined by the MXFSchema.xsd input document.</li> <li>The document may optionally include Descriptive Metadata <ul style="list-style-type: none"> <li>Descriptive Metadata shall comply with the schema defined by the DMS-x.xsd document(s) fed as input.</li> <li>Descriptive Metadata shall be hooked to the Structural Metadata as defined in the MXF specification and in the MXFSchema.xsd. Please refer to [1] for information on this.</li> </ul> </li> </ul>
Output File Name	The name of the MXF file to output.
Essence File Name(s)	The name(s) of the Essence File(s) referenced in the Structural Metadata fed to the Component and to be included in the file.

## Outputs

Output.mxf	The resulting MXF file
------------	------------------------

## Notes

The C++ version of this Component provides streaming interfaces for on-the-fly wrapping of Essence and Metadata. Please refer to the C++ API documentation shipped with the Component.

## MXFMetaUpdater

Updates an MXF file with the specified Metadata.

## Type

Active X Component

Automation Interface	<b>Yes</b>
Graphic User Interface	<b>No</b>

## Inputs

Input.mxf	The input MXF file
MXFSchema.xsd	<p>MXF Structural Metadata schema in XML Schema format. This document is supplied by MOG Solutions as part of the <b>MXFComponentSuite</b>.</p> <p>This document is used by the Component to validate the Metadata fed to it in XML format, to interpret it and tag it with the correct keys to be written to the MXF file in KLV format.</p> <p><b>The component also allows as input a binary version of the schema for improved performance.</b></p>
DMS-x.xsd documents	<p>XML Schema documents that formally describe the structure (scheme) of the Descriptive Metadata.</p> <p>These documents are used by the Component to validate and interpret the Descriptive Metadata fed to it.</p> <ul style="list-style-type: none"> <li>• One of these documents may be the DMS-1.xsd document that formally specifies DMS-1 Descriptive Metadata Schema as defined in the MXF Specification.</li> <li>• Other customized Descriptive Metadata Schemes (DMS's) may be fed, including proprietary DMS's, as long as these make proper use of Descriptive Metadata hooks as defined in the MXF Specification.</li> <li>• Any DMS-x.xsd document must follow the same design principles followed in MXFSchema.xsd. Please refer to [1] for information on schema design guidelines.</li> </ul> <p><b>The component also allows as input a binary version of the schema for improved performance.</b></p>
metadata.xml	<p>XML Document with the new Metadata to be included in the MXF file</p> <ul style="list-style-type: none"> <li>• The document shall include the MXF Structural Metadata that describes the structure of the file. This XML document shall comply with the schema defined by the MXFSchema.xsd input document.</li> <li>• The document may optionally include Descriptive Metadata <ul style="list-style-type: none"> <li>○ Descriptive Metadata shall comply with the schema defined by the DMS-x.xsd document(s) fed as input.</li> <li>○ Descriptive Metadata shall be hooked to the Structural Metadata as defined in the MXF specification and in the MXFSchema.xsd. Please refer to [1] for information on this.</li> </ul> </li> </ul>

## Outputs

The Component does not produce any new file. It updates the one specified as input.

## MXFUnwrapper

Extracts Essence and related Metadata from an input MXF file

## Type

Active X Component

Automation Interface	<b>Yes</b>
Graphic User Interface	<b>No</b>

## Inputs

Input.mxf	The input MXF file
MXFSchema.xsd	<p>MXF Structural Metadata schema as an XML Schema document. This document is supplied by MOG Solutions as part of the <b>MXFComponentSuite</b>.</p> <p>This document is used by the Component to interpret the Metadata fed to it embedded in the MXF file in KLV format.</p> <p><b>The component also allows as input a binary version of the schema for improved performance.</b></p>
DMS-x.xsd document(s)	<p>XML Schema documents that formally describe the Descriptive Metadata data models to be expected by the Component.</p> <p>These documents are used by the Component to interpret the Descriptive Metadata fed to it embedded in the file in KLV format.</p> <p>Any Descriptive Metadata embedded in the file for which no DMS-x.xsd is provided shall not be decoded and, therefore, shall not be extracted by the Component.</p> <p><b>The component also allows as input a binary version of the schema for improved performance.</b></p>
Target Essence Directory	The location to which Essence files should be written

## Outputs

metadata.xml	<p>An XML document with all Structural Metadata in the file, plus any Descriptive Metadata the Component was able to decode, based on the provided DMS-x.xsd document(s).</p> <p>The Metadata shall identify any Essence files extracted from the MXF file.</p>
Essence Files	Essence is extracted from the MXF file and stored as files in the specified Target Essence Directory

## Notes

The C++ version of this Component provides streaming interfaces for on-the-fly unwrapping of Essence and Metadata. Please refer to the C++ API documentation shipped with the Component.

## MXF DMS Editor

This Component allows you to graphically create your own Descriptive Metadata Schemes (DMS's), or extend existent ones. Its output is an XML Schema document that can be readily fed into other Components of the Suite.

## Type

Active X Component

Automation Interface	<b>Yes</b>
Graphic User Interface	<b>Yes</b>

## Inputs

MXFSchema.xsd	<p>MXF Structural Metadata schema as an XML Schema document. This document is supplied by MOG Solutions as part of the <b>MXFComponentSuite</b>.</p> <p>The Structural Metadata includes the hooks defined in the MXF specification on which to plug new DMS's. It also formally defines the base data types listed in the MXF specification that can be used on DMS definitions.</p>
DMS-x.xsd	Optional XML Schema documents that formally describe existent DMS's.

Document(s)	These inputs are needed if you wish to extend existing DMS's, like DMS-1 for example.
-------------	---------------------------------------------------------------------------------------

### Outputs

DMS-new.xsd	The new or extended DMS in XML Schema format, according to the schema design guidelines as defined in [1]
-------------	-----------------------------------------------------------------------------------------------------------

## Essence Analyzers

An Essence Analyzer parses a raw Essence file and produces an XML document with technical Metadata about that Essence. The document's format complies with the schema design guidelines as specified in [1].

For each Essence type there is a **Essence Analyzer** Component, but they all share the same interface.

### Type

Active X Component

Automation Interface	<b>Yes</b>
Graphic User Interface	<b>No</b>

### Inputs

Essence File Name	The name of the Essence File to be analysed
-------------------	---------------------------------------------

### Outputs

metadata.xml	The technical Metadata about the analysed Essence in XML format.
--------------	------------------------------------------------------------------

## MXF DirectShow® Filter

A Direct Show® filter that decodes MXF and plays the output timeline of an MXF file specified in MXF using the MaterialPackage mechanism.

### *Type*

Direct Show® Filter

### *Inputs*

input.mxf	Receives in its input an MXF file
-----------	-----------------------------------

### *Outputs*

Raw Essence	Delivers at the output a continuous Essence stream representing the output timeline of the file (Material Package)
-------------	--------------------------------------------------------------------------------------------------------------------

### *Notes*

With this filter registered in DirectShow®, you can actually play your MXF files in a DirectShow® based player (like Microsoft Windows Media Player®) just by double clicking the file in the Windows® Explorer.

## References

- [1] MOG Solutions 2003, "XML Schema for MXF Metadata", 2003