Scripting with iView Multimedia

by Yan Calotychos, ©iView Multimedia Ltd. 1999

Introduction

As of version 3.5, iView Multimedia enables users to extend its functionality via AppleScript.
iView Multimedia offers a rich AppleScript vocabulary, primarily designed to allow export and import media files, as well as annotations, media info and digital camera info, to and from the Finder and third party data base and spreadsheet applications.
The iView Multimedia vocabulary does NOT intend to replicate functionality provided through the interface of the application, nor is it intended to be used as a remote control to drive the software.

AppleScript, though elegant and syntactically natural, it is a programming language, and as such, unless it is used consistently (and frequently), it can be frustrating in order to accomplish a task that could otherwise, be issued with a single click via the application's interface.

However, there are times when a solution to a problem may require a highly complex and densely populated dialog to accomplish. For example assume you had 200 media files in a catalog and you wanted to rename them incrementally, but also based on type. This kind of problem would require a "rename" dialog with several options and settings that would at best be intimidating and very confusing.
This is when AppleScript becomes useful.
iView Multimedia uses AppleScript to extend it's functionality, to handle specific situations that are either too repetitive to do via the interface or too complex to do via a settings dialog.

Thinking about writing your own scripts?

Well, as mentioned earlier, AppleScript is a computer language, and as such you'll need to learn its intricacies in order to use it.
Examining dictionaries via AppleScript Editor helps.
Studying samples provided with most applications that support scripting also helps.
However be warned!
Unlike other computer languages, AppleScript is rather 'free form'. A program line that works fine with one application may not work on another, as both vocabulary and syntax used are application (and implementation) dependent. In other words the use of an AppleScript verb (such as 'close') in an application depends very much on the nature of the application and the way the application interprets it.

The 'Object' model

The object model is the most widely adapted way to support AppleScript, as it provides a 'natural' way to present and describe the scriptable functions of the application.
The user can start thinking in terms of objects (a window or a media object), their properties (name, position, etc.), and one or more actions (events), that can be applied to, in order to get or set that object's properties.
Scripts lines typically contain a verb (event), a noun (object).

```
  [close] [window] [whose name contains "caltalog"]
```

And typically some clause that specifies the object on which the verb (event) is applied to. This event is generally associated with a property of the object (such as its name, visibility, position, etc.).

Objects in iView Multimedia

iView Multimedia is modeled on 4 objects: the application object; the window object; the media object, and the mark object.
A single "application" object can contain one or more "windows" and a set of seven "marks". Each "window" can contain one or more "media objects".
Each object has a set of properties, specific to iView Multimedia.

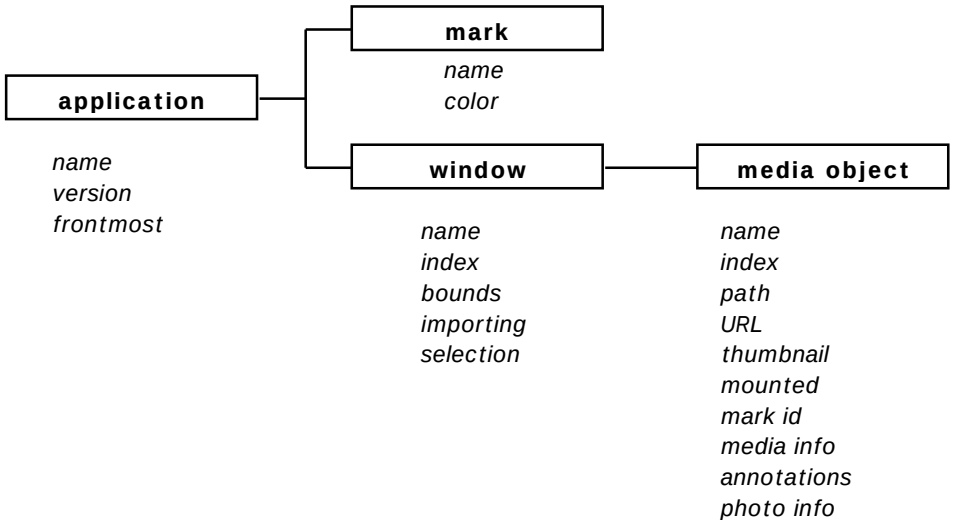

Figure 1. iView Multimedia Objects and Object Properties

The 'application object' is typically launched via the Finder, and supports a few universal events such as 'open', 'quit', 'print' etc.
The 'mark object' is associated with a unique set of colors that can be used to 'color' & 'label' media in an open catalog. Both the 'name' and the 'color' properties of a mark object can be retrieved and modified via apple events.
Note that under the current version there is only a single set of 7 marks available by the application.
The 'window object' is associated with a iView Multimedia catalog. An important property of the window is 'importing' which indicates the state for a particular window. An AppleScript can wait until the importing is complete (its value is false) in order to perform an action such as closing, printing or saving the window contents in a catalog file.
Finally the "media object" (referred to as object for short), carries all properties for an entry in your catalogs. These include the basic elements such as name and location of the original file in your disks, and more complex elements such as the annotation, media and photo info records that include a list of sub-properties that define the object in greater detail (see the applications dictionary).

Where do I go from here?

Through a small and easy to learn set of Apple Events, iView Multimedia enables you to share data between iView Multimedia and other AppleScript aware applications,

including database, D.T.P. and spreadsheet applications.
This enables you to use dedicated software for archiving, printing and data editing. AppleScript also enables you to create complex scripts to edit your annotations within iView Multimedia.
For more information please study the sample scripts provided in this package. You may find that most code is commented. You can start removing comments line by line and try each line with an open iView Multimedia catalog.

Supported Apple Events for each object type.

Event	App	Win	Obj	Mrk	Remarks
count	√	√	√	√	Count objects within a parent object.
exists	√	√	√	√	Verify if an object exists.
data size	√	√	√	√	Data size of a property of an object (in bytes).
get	√	√	√	√	Get the value of a property of an object.
set	.	√	√	√	Set the value of a property of an object.
save	.	√	.	.	Save a window in a catalog file.
close	.	√	.	.	Close a window.
select	.	√	.	.	Bring a window to front.
make	.	√	.	.	Make a new window.
delete	.	.	√	.	Delete one or more objects.
do menu	√	.	.	.	Trigger selected menu items
open	√	.	.	.	Open catalogs or files/folders in window
print	√	.	.	.	Print catalogs
quit	√	.	.	.	Quit application