CreateXAEVT

Err:=CreateXAEVT(Class;id;addr;returnID;TransID;aevt)

This works exactly like CreateAEVT except it allows you to supply a return ID and/or transaction ID. The return ID will allow you to uniquely identify the reply to this AppleEvent if you allow the reply to be queued and install a handler for 'aevt','ansr'. You can allow the return ID to be generated automatically by passing -1 for the return ID. The transaction ID, if non-zero, can be used to specify that a series of AppleEvents are part of a single operation. (Not all applications support this feature.)

CopyDesc

Result:=CopyDesc(aevt)

Creates a duplicate copy of an AppleEvent, AERecord, or target ID. You can use CopyDesc(0) in an AppleEvent handler to obtain a handle to the actual event being processed. This is useful if you wish to write extensions to System 7 Pack™ which need to access an AppleEvent. When you finish using the result, you must dispose of it by calling DisposeDesc.

GetTransactionID
GetReturnID

Result:=GetTransactionID(aevt)
Result:=GetReturnID(aevt)

Obtains the transaction ID or return ID of an AppleEvent. You can pass 0 to use the AppleEvent currently being processed in an AppleEvent handler procedure. GetReturnID can be used in a handler for queued reply events (class 'aevt',id 'ansr') to determine which AppleEvent is being replied to.

GetAEInfo

Err:=GetAEInfo(Descriptor;Size;Keys;Types;Lengths)

Returns arrays of keywords, types, and lengths of each descriptor in an AppleEvent, AERecord, or List. 'Size', a LongInt variable, will be set to the number of descriptors. 'Keys' and 'Types' should be arrays of String(4) and 'Lengths' should be an array of long integers.

GetNthDesc

Err:=GetNthDesc(aevt,index,key,type,result)

Extracts a descriptor from an AppleEvent, AERecord, or List by position rather than by keyword. ‘index’ tells which descriptor to extract (the first one is 1). ‘key’ and ‘type’ will be set to the keyword and actual datatype of that descriptor. ‘result’ will be a handle to the descriptor. When you finish using it, you must dispose of the result by calling DisposeDesc.

GetNthItem

Err:=GetNthItem(aevt,index,key,type,textVar)

Extracts a value from an AppleEvent, AERecord, or List by position rather than by keyword and coerces the result to text. ‘index’ tells which descriptor to extract (the first one is 1). ‘key’ and ‘type’ will be set to the keyword and actual datatype of that descriptor. This will work for all numeric types and any other data which can be coerced to text. This command, along with GetAEInfo and GetNthDesc can be used to examine an entire AppleEvent and extract the contents of all nested lists.

Example:

```
--- DumpList
$err:=GetAEInfo ($1;$Size;Keys;Types;Lengths)
For ($i;1;$size)
   MESSAGE($2*"> ")
   $err:=GetNthDesc ($1;$i;$aKey;$aType;$aDesc)
   If (($aType="list")|($aType="reco"))
      MESSAGE($aKey+"     "+$aType+Char(13))
      DumpList ($aDesc;$2+1)
   Else
      $err:=GetNthItem ($1;$i;$aKey;$aType;$value)
      MESSAGE($aKey+"     "+$aType+"     "+$value+Char(13))
   End if
   $err:=DisposeDesc ($aDesc)
End for
```

ResetTimer

Err:=ResetTimer

Resets the AppleEvent manager’s timer to prevent timing out when responding to an AppleEvent. This command can be used in an AppleEvent handler in 4D 3.0 or later which needs to perform some lengthy processing before it returns a reply. Any non-zero result means that your handler was called with no reply requested or that you’re not in an AppleEvent handler procedure.

ObjX

ospec:=ObjX(class;container;value)

Creates an object specifier referring to a special object such as All, Any, First, Middle, Last.

Value should be one of these 4-character strings: "all ", "any ", "firs", "midd", or "last".

Coerce

Err:=Coerce(descriptor;type)

Allows you to coerce an AERecord to a specific type. For example, you can create an object specifier by coercing a record containing "want", "from", "form", and "seld" descriptors into type "obj ".

Comparison

ospec:=Comparison(class;container;object1;operator;value)

Creates a comparison object descriptor which lets you request something like "All rows where cell 1 contains 'Mike'". Class is the object class we're interested in. Container specifiers the container in which we should search. Object1 is an object specifier describing one operand in the comparison. This object specifier should use -1 as the container, which means "object being examined" and will be an object found within the previously specified container. Operator should be a 4-letter string describing a comparison, such as "cont" for contains, "bgwt" for begins with, "ends" for ends with, or "= ", "> ", ">= " etc. Value should be a text value to be compared against (if necessary, a number string will automatically be coerced to a numeric value).

Example:

```
$err:=MakeAddress ("FMPR";FileMaker)
If ($err=0)
 ` Show all records (rows) where field (cell) 1 contains "Mike"
 $err:=CreateAEVT ("misc";"mvis";FileMaker;aevt)
 $obj:=Comparison("crow";Obj("docu";0;1);Obj("ccel";-1;1);"cont";"Mike")
 $err:=PutObject (aevt;"----";$obj) ` automatically disposes $obj
 $err:=SendAppleEvent (aevt;reply;kAEWaitReply ;-1)
 $err:=DisposeDesc (aevt)
 $err:=DisposeDesc (reply)
 $err:=DisposeDesc (FileMaker)
End if
```