BringToFront

Err:=BringToFront(Signature)

Brings the application specified by a 4-character signature to the front. The application must already be running. This only requests that the application be brought to the front. It won't actually happen until 4D® gives up CPU time by calling WaitNextEvent. In pre-3.0 versions of 4D®, you may have to call ProcessAEVT to force it to happen.

FindAppName

Err:=FindAppName(Signature;Name)

Given a 4-character signature, this will return the name of the application associated with that ID. It will return the name without a full path.

FindCreator

Err:=FindCreator(File Name;Creator ID)

Given a full or partial pathname of a file, will return the creator ID. This is useful to figure out which application to launch when any document is selected.

Example:

```
Set Channel(10;"")
if (OK = 1)

Set Channel(11)

$err := FindCreator(Document;$creator)

if ($err = 0)


$err := Launch($creator;Document)

end if
end if
```

IsRunning

Result:=IsRunning(Creator ID)

Returns 1 if the application is running, 0 otherwise.

Long

Result:=Long(TypeString)

Converts a 4-character type string to a long integer value. This is useful when a type or application signature needs to be passed to one of the low-level functions.

ProcessList

Err:=ProcessList(Visible;StringArray)

Builds an array of the signatures of all running processes. If 'visible' is non-zero, only the visible applications will be listed. Otherwise, the list will also include invisible processes, such as background-only applications and File Sharing Extension. The array will be allocated if necessary and will contain a 4-character signature string for each running (or visible) application. In a compiled database, you must pre-define the array as fixed 4-character strings, although the number of elements will change dynamically.

Example:

```
If (Before)
   $err:=ProcessList (0;ProcList)
   If ($err=0)
      ARRAY STRING(64;ProcNames;Size of array(ProcList))
      For ($i;1;Size of array(ProcList))
         $err:=FindAppName (ProcList$i;$pn)
         ProcNames$i:=$pn
      End for
   End if
End if
```

S7Version

Version:=S7Version(Serialization String)

Returns the serialization string and version # of System 7 Pack™ as a long integer value, which will be negative for a demo version. The current version is 3.5, which will be returned as 350 (or -350 for the demo version).

SetTimeout

SetTimeout(Ticks)

Sets the time-out value in ticks (60ths of a second) to use when sending an AppleEvent. A good value to use is "600", which is 10 seconds. To specify the default value, which is about a minute, pass -1. If the value is too short, you may get an error code of -1712 (time out) when sending an AppleEvent. If the value is too long, the system will appear to "hang" while waiting for a

response if none is returned.

System7

Result:=System7

Returns 1 if you're running a compatible system which supports AppleEvents. This should be one of the first calls in your program, so you can provide a graceful exit if necessary. Other System 7 Pack™ functions will not crash if you're not running system 7, but will simply return an error code of -2.

IsVersion3

Result:=IsVersion3

Returns non-zero if you’re running 4D® v3.0 or later. NOTE: This replaces S7P 3.3’s “AE Process ID”. Since 4D 3.0.3 and later handle AppleEvents in a system process, we no longer provide direct access to it.