Using convert_constants

Beginning with 4D v11, plugin developers were discouraged from using the Macintosh resource manager, so Active4D v6 does not. As a result, Active4D v6 cannot read custom 4D constants, because they are kept in 4DK# resources.

The **convert_constants** method allows you to create an Active4D library that contains all of the custom 4D constants in a given resource file. By importing the library in Active4D, you will be able to reference the custom constants as you did before.

1. Import the convert_constants.c4d method into a structure.
2. Write a method that calls **convert_constants**. The first parameter is the name of the library you want to create (without the ".a4l" extension). If you want to convert constants from a particular resource file, open the resource file using **Open resource file** and pass the document reference as the second parameter to **convert_constants**.
3. **convert_constants** will prompt you for a destination folder in which to place the library. After selecting the folder, it will create the library and then ask you if you want to view the created library.

Example

Let's say we converted a database and migrated our custom constants to a "User constants.bundle" plugin. Our custom constants have two themes:

"foo"
    foo name   "foobar"
    foo age   27

"bar"
    bar name   "barfoo"
    bar age   31

After importing the **convert_constants** method into a database, we would convert them into a library by writing a method like this:

```
$res:=Open resource file("")

If (OK=1)
    convert_constants("constants";$res)
    CLOSE RESOURCE FILE($res)
```

```
End if
```

After running, you will end up with a text file called “constants.a4l” in the destination folder you selected. The source of the library will be as follows:

```
library "constants"

// foo
define(foo name; "foobar")
define(foo age; 27)

// bar
define(bar name; "barfoo")
define(bar age; 31)

end library
```

To use this library, put it in the Active4D folder (or in any folder listed in “lib dirs” in Active4D.ini), and put this code in the **On Application Start** event handler in Active4D.a4l:

```
import("constants")
```

After doing this all of your existing custom constant references will work as they did before.