OE System Object Reference

class System

The System class has a variety of system oriented methods. It is a class without any instance methods, only class methods. To use any of these methods you must first use the import method to import this class, as follows:

```
import('System');
```

Instance Methods

NONE

Class Methods

System Information

`String` **`currentDir`**;

This method returns the name of the current directory.

`String` **`systemDir`**;

This method returns the name of the system directory.

`String` **`windowsDir`**;

This method returns the name of the MS Windows directory.

`String` **`computerName`**;

This method returns the name of the computer.

`String` **`userName`**;

This method returns the current user name.

`Integer` **`screenWidth`**;

This method returns the width of the screen in pixels.

`Integer` **`screenHeight`**;

This method returns the height of the screen in pixels.

`Integer` **`screenClientWidth`**;

This method returns the width of the client area of the screen in pixels.

`Integer` **`screenClientHeight`**;

This method returns the height of the client area of the screen in pixels.

`Integer` **`screenCaptionAreaHeight`**;

This method returns the height of the caption area in pixels.

`Integer` **`getLastError`**;

This method returns the error code of the last system call.

Environment

`String` **`getEnv`**(`String envName`);

This method returns a string value containing value of a system environment variable.

`Integer` **`setEnv`**(`String envName, String envValue`);

This method sets the value of the specified system environment variable. The return value is 0 unless there is an error, then the system error code is returned.

> **Note:** This sets the environment variable for the current process only. It does not permanently set this environment variable.

`String` **`expandEnvStrings`**(`String envValue`);

This method expands the environment strings and replaces them with values defined for the current user.

Folder/Directories

`Integer` **`createFolder`**(`String folderName`);

This method creates a new directory (or folder) on a disk volume.

`Integer` **`removeFolder`**(`String folderName`);

This method deletes a directory (or folder) from a disk volume. The directory must be empty.

Files

`Boolean` **`fileExists`**(`String fileName`);

This method returns `true` if the file exists and `false` otherwise.

`String` **`extractFileExtension`**(`String fileName`);

This method returns the extension part of the full path name.

`String` **`extractFilename`**(`String fileName`);

This method returns the file part of the full path name.

`String` **`extractPathname`**(`String fileName`);

This method returns the part of the full path name.

`String` **`extractFullPathname`**`(String fileName);`

This method returns the full path name for a file name.

`String` **`replaceFileExtension`**`(String fileName, String newExtension);`

This method returns the replaces the current extension with the one specified.

Standard Dialogs

`String` **`dlgOpenFile`**`(String title,String initFolder,String initFileName);`

This method displays a standard open file dialog box and returns the selected file name. It will return `null` if the user presses the cancel button.

`String` **`dlgSaveFile`**`(String title,String initFolder,String initFileName);`

This method displays a standard save file dialog box and returns the selected file name. It will return `null` if the user presses the cancel button.

`String` **`dlgBrowseForFolder`**`(String title,String initFolder);`

This method displays a standard select folder dialog box and returns the selected folder name. It will return `null` if the user presses the cancel button.

Miscellaneous

`Integer` **`runProgram`**`(String cmdLine[,Integer waitTime]);`

This method runs an external program. The optional waitTime parameter lets you wait the specified number of milliseconds before continuing. If this value is 0 (or not specified), it will return immediately.

`Integer` **`clock`**`;`

This method returns the processor time in seconds times 1000. You can get the compute time by converting it to a real number (Number) and dividing by 1000.

```
var c=Number.make(System.clock)/1000.0;
```

This can be useful for timing various processes. For example,

```
var startTime=System.clock;
// ...
// Do some long process here
// ...
var endTime=System.clock;
var duration=Number.make(endTime-startTime)/1000.0;
writeln('Elapsed time in seconds='+duration.toString);
```

`Integer` **`randomNumber`**`;`

This method returns a pseudo random number.

`Integer` **`randomNumberSeed`**;

This method sets the starting point for generating a series of pseudo random numbers. To reinitialize, use 1 as the seed.

`Integer` **`beep`**[`Integer frequency[, Integer duration`]];

This method generates an audio beep tone. The optional parameters are the frequency in hertz and the duration in milliseconds. The default values are 750 for the frequency and 300 for the duration.

`String` **`computeUUID`**;

This method computes and returns a UUID value.

