

Processing Methods

The following processing methods are available:

- CallExternalFunction
 - Pause
-

CallExternalFunction

Call a DLL function.

Syntax

```
CallExternalFunction (DLLname, Functionname[ , Parameterlist])
```

Parameters

<i>DLLname</i>	A valid DLL name, including the path.
<i>Functionname</i>	Name of the function in the DLL.
<i>Parameterlist</i>	One or more parameters to be used for the function.

Description

The `CallExternalFunction` method calls the specified function in the DLL. A callback pointer and the values from the *Parameterlist* are passed to the function as explained below.

The function prototype for the functions in the external library is:

```
int functionname (PSAGJSCALLBACK pCallback, int argc, char** argv)
```

The first parameter (`PSAGJSCALLBACK`) is a pointer to the following structure:

```
typedef struct tagSAGJSCALLBACK
{
    CALLBACK_PROC SetJSVariable;
    CALLBACK_PROC GetJSVariable;
}SAGJSCALLBACK, *PSAGJSCALLBACK;
```

Two callback procedures are provided that enable you to communicate with the JavaScript that called the function: `GetJSVariable` and `SetJSVariable`. You use these functions to access variables in the JavaScript by name to retrieve or set the values of these variables. The procedures have the following syntax:

```
SetJSVariable (char* szVarName, char* szVarValue)
GetJSVariable (char* szVarName, char* szVarValue)
```

The second parameter (`int argc`) passes the number of the parameters that are in the *Parameterlist*.

The third parameter (`char** argv`) is a pointer to an array of strings that contains the values from the *Parameterlist*. Note that values are passed as strings.

Example

```
iFunctionRet= NSW.CallExternalFunction ('f:/mylib/mylib.dll', 'fnMylib', '10', '4', 'x')
```

When the function `fnMylib` is called, the second parameter (`int argc`) has the value 3 since there are three values in the *Parameterlist*. The third parameter (`char** argv`) points to a string array with the three strings "10", "4" and "x" (in that order) in the array.

The following is an example JavaScript program which calls the function `fnAdd2` in *mylib.dll*. The function `fnAdd2` uses the callback functions to retrieve the value of the script variable named `i`, to add 2 to it and to put the new value to the script variable.

```
i=1;
NSW.CallExternalFunction ('f:/mylib/mylib.dll', 'fnAdd2')
NSW.MessageBox ("i = " + i)
```

This is the function `fnAdd2` in the DLL:

```
int fnAdd2(PSAGJSCALLBACK pCallback, int argc, char** argv)
{
    char* szValue = new char(20);

    // Get the variable 'i' from JavaScript
    pCallback->GetJSVariable("i", szValue);

    // convert value to integer
    int iVal = atoi(szValue);
    iVal += 2;

    // convert value back to string
    itoa (iVal, szValue, 10);

    // Set the variable 'i' in JavaScript
    pCallback->SetJSVariable("i", szValue);

    return 1;
}
```

Return Values

Return value of the called DLL function. This must be an `int` value.

Pause

Pause processing of the script file.

Syntax

```
Pause(Number)
```

Parameters

Number The number of milliseconds that the script is to be paused.

Description

Pauses the execution of the script for the specified number of milliseconds.

Example

```
NSW.Pause(1000);
```

Return Values

None.