

User Interface Methods

The following user interface methods are available:

- Cancel
 - DeleteUserVariable
 - GetClientArgument
 - GetUserConnectionValue
 - GetUserInput
 - GetUserVariableValue
 - MessageBox
 - OutputMsg
 - SetClientArgument
 - SetUserConnectionValue
 - SetUserVariableValue
-

Cancel

Cancel processing of the script file immediately.

Syntax

```
Cancel()
```

Parameters

None.

Description

The Cancel method stops processing and terminates the script. This can also be called from a nested function.

Example

```
NSW.Cancel();
```

Return Values

None.

DeleteUserVariable

Delete a user-defined variable.

Syntax

```
DeleteUserVariable (Variable)
```

Parameters

Variable A user-defined variable (string).

Description

The DeleteUserVariable method removes a user-defined variable that has been set with the SetUserVariableValue method.

Example

```
bRet = NSW.DeleteUserVariable ("myvar");
```

Return Values

If the method is successful, it returns true.

GetClientArgument

Return values defined by a client-side user exit (API).

Syntax

```
GetClientArgument(Name)
```

Parameters

Name Name of a script argument defined in the client.

Description

The GetClientArgument method returns the values defined by the user exit (API) method SetScriptArgument.

String comparison is case-sensitive.

Example

```
szArgValue = NSW.GetClientArgument( "param1" );
```

Return Values

The defined value for this argument.

If the argument is not set, an empty string is returned and errno is set to PROC_ARGUMENT_NOT_FOUND.

GetUserConnectionValue

Return the value of a user-specific connection variable.

Syntax

```
 GetUserConnectionValue (Variable)
```

Parameters

Variable A user-specific connection variable. See the table below.

Description

The `GetUserConnectionValue` method returns the value of a user-specific connection variable.

One of the following user connection variables has to be specified for the *Variable* parameter:

Variable	Description
NSW_CLIENT_TCP_IP_ADDR	IP address of the client.
NSW_CLIENT_HOST_NAME	Host name of the client.
NSW_USER_NAME	User name.

Thus, if you want to know the name of the current user, you have to specify `NSW_USER_NAME` for the *Variable* parameter.

Example

```
szValue = NSW.GetUserConnectionValue (NSW_CLIENT_TCP_IP_ADDR);
```

Return Values

Value of the variable.

If the variable was not found, an empty string is returned and `errno` is set to `NSW_USERVALUE_NOTFOUND_ERROR`.

GetUserInput

Display a dialog box in which the user can enter text.

Syntax

```
 GetUserInput(Prompt1, Prompt2, Flags, InitialValue[, Length])
```

Parameters

<i>Prompt1</i>	A string that is displayed in the dialog box.
<i>Prompt2</i>	A string that is displayed in a second line in the dialog box.
<i>Flags</i>	Can be a combination of the following: REQUIRED_DATA, LENGTH_LIMIT, STRING_DATA, NUMERIC_DATA and PASSWORD_CHAR.
<i>InitialValue</i>	A default text that is displayed in an input field of the dialog box.
<i>Length</i>	Maximum length of the input field. If LENGTH_LIMIT is set, <i>Length</i> must also be valid.

Description

The `GetUserInput` method displays a dialog box in the viewer of the client workstation. It returns the data entered by the user and an identifier for the button pressed by the user.

Example

```
strPassword = NSW.GetUserInput("Password", "Enter your password", REQUIRED_DATA | LENGTH_LIMIT | PASSWORD_CHAR, 12);
strFromUser = NSW.GetUserInput("Enter a number", "", NUMERIC_DATA);
```

Return Values

The data entered by the user.

The property `NSW(errno)` will be set to 0, if the user chooses the **OK** button. It will be 1, if the user chooses the **Cancel** button.

GetUserVariableValue

Return the value of a user-defined variable.

Syntax

```
 GetUserVariableValue ( Variable )
```

Parameters

Variable A user-defined variable (string).

Description

The `GetUserVariableValue` method returns the value of a user-defined variable that has been set with the `SetUserVariableValue` method.

Example

```
 szValue = NSW.GetUserVariableValue ( "myvar" );
```

Return Values

Value of the variable.

If the variable was not found, an empty string is returned and `errno` is set to `NSW_USERVALUE_NOTFOUND_ERROR`.

MessageBox

Pause processing of the script file and display a message box.

Syntax

```
MessageBox ( Text[ , Flags] )
```

Parameters

Text String to be shown.

Flags A combination of the flags for a Windows message box (for example, MB_OK or MB_YESNO).

Description

The MethodBox method displays a message box in the viewer on the client workstation. Processing of the script file is resumed when the user chooses a command button in the message box.

Example

```
NSW.MessageBox( "Error opening session" );
ReturnCode = NSW.MessageBox( "Good morning" , MB_OK );
```

Return Values

ReturnCode is set to the Windows identifier of the button that has been chosen (for example, IDOK or IDCANCEL).

OutputMsg

Send non-blocking output to the viewer.

Syntax

```
OutputMsg(String)
```

Parameters

String String to be shown on the client side.

Description

The OutputMsg method sends a message text to the viewer.

For the GUI viewers, this output appears in the status line.

For the Terminal Viewer, this output appears in the output window.

Example

```
NSW.OutputMsg( "Now reading database" );
```

Return Values

None.

SetClientArgument

Define arguments to be processed by a client-side user exit (API).

Syntax

```
SetClientArgument(Name)
SetClientArgument(Name, Value)
```

Parameters

Name Name of a script argument.

Value Value for this argument.

Description

The SetClientArgument method defines the arguments to be returned to the client and to be processed by the user exit (API) method GetScriptArgument.

If only the *Name* is given, the argument is deleted from the list of client arguments.

Example

```
bArgDel = NSW.SetClientArgument( "param1" );
bArgSet = NSW.SetClientArgument( "param1" , "retval" );
```

Return Values

If the method is successful, it returns true.

If the argument is not set or if it has been deleted, false is returned and errno is set to PROC_ARGUMENT_NOT_FOUND.

SetUserConnectionValue

Set the value of a user-specific connection variable.

Syntax

```
SetUserConnectionValue (Variable, Value)
```

Parameters

Variable A defined user-specific variable.

Value Value to which the variable is to be set.

Description

The SetUserConnectionValue method sets the value of a user-specific variable.

The following user connection variable has to be specified for the *Variable* parameter:

Variable	Description
NSW_USER_PASSWORD	User password.

Example

```
bRet = NSW.SetUserConnectionValue (NSW_USER_PASSWORD, "abc");
```

Return Values

If the method is successful, it returns true.

If the variable was not found, false is returned and `errno` is set to `NSW_USERVALUE_NOTFOUND_ERROR`.

If you try to set a variable which cannot be modified (such as `NSW_CLIENT_HOST_NAME` which is only allowed with the `GetUserConnectionValue` method), `errno` is set to `NSW_USERVALUE_NOTSETTABLE_ERROR`.

SetUserVariableValue

Set the value of a user-defined variable.

Syntax

```
SetUserVariableValue (Variable, Value)
```

Parameters

Variable A user-defined variable (string).

Value Value to which the variable is to be set.

Description

The SetUserVariableValue method defines a variable and sets its value.

For example, you can define a variable with the name "myvar" and set its value to "abc". This variable is then permanently stored for the current user and can be retrieved (for example, after a restart of the server) with GetUserVariableValue("myvar").

Example

```
bRet = NSW.SetUserVariableValue ( "myvar" , "abc" );
```

Return Values

If the method is successful, it returns true.