

A Scriptable COM Object For Modifying User Settings

This document provides directions for using the **ScriptHlp** object to modify call handling mode for individual users or groups of users. With **ScriptHlp**, an administrator can standardize telephony practices by scheduling the assignment of call handling modes for groups of users. For example, a supervisor can set up a group that will receive after-hours calls at their home extensions. Other applications include setting call handling mode to voice mail after closing time, and locking down class of service for the evening.

Overview

ScriptHlp is a Microsoft COM (Component Object Model) object stored on your headquarters server (*Shoreline.ScriptHlp.CScriptHlp.1*).

The four methods belonging to the **ScriptHlp** object are described in the table below.

Method	Description
<code>initialize()</code>	Initializes the object script. Must be called before making any other calls.
<code>setUserCallHandlingMode(DN, CallHandlingModeName)</code>	Sets call handling mode for a specified user. DN (Dialable Number) identifies a user. CallHandlingModeName is a string representing one of five call handling modes: standard, in a meeting, out of office, extended absence, and custom.
<code>setUserUserGroup(DN, UserGroupName)</code>	Sets the user group for a specified user. DN (Dialable Number) identifies a user. UserGroupName is a string representing a user group. (See <i>Shoreware Director, Users.. UserGroups</i> for a list of user group names.)
<code>sendUserHome(DN)</code>	Returns the user to his or her home port. DN (Dialable Number) identifies a user.

Calling the Object

The object can be called from any of Microsoft's scriptable languages, including JavaScript and VB Script.

Error Log File

The ScriptHlp object logs errors to the file *\Shoreline Data\Logs\ScriptH-mm-dd-yy.log*.

Sample Script

The following sample script, *TestScriptHlp.js*, demonstrates the use of the **ScriptHlp obj** to set user groups and call handling modes.

Run *TestScriptHlp.js* with the command *TestScriptHelp.js <input file>*.

```
var g_scriptHlpr;

if (WScript.Arguments.length < 1)
{
    WScript.echo("invalid arguments");
    WScript.Quit();
}

var filename = WScript.Arguments(0);
WScript.echo("Attempting to open file: " + filename);

var fso = new ActiveXObject("Scripting.FileSystemObject");

if (!fso.FileExists(filename) )
{
    WScript.echo("File does not exist !");
    WScript.Quit();
}

var file = fso.OpenTextFile(filename,1,true);
var line;
var array;

// initialize the connection
CSIS_Connect()
```



```

while (!file.AtEndOfStream)
{
    line = file.ReadLine();

    array = line.split(",");
    if (array.length != 3)
    {
        WScript.echo("Wrong number of arguments in: " + line);
        continue;
    }
    array[0] = Trim(array[0]);
    array[1] = Trim(array[1]);
    array[2] = Trim(array[2]);

    WScript.echo ("Ext: " + array[0] + " - Setting " + array[1] + " to: " + array[2]);

    if (array[1] == "CH")
    {
        try
        {
            CSIS_SetUserCHM(array[0], array[2]);
            WScript.echo ("Successfully Set Call Handling Mode for Ext: " + array[0]);
        }
        catch (e)
        {
            WScript.echo ("Error Setting Call Handling Mode for Ext: " + array[0] + " : " + e.error);
        }
    }
    else
    if (array[1] == "UG")
    {
        try
        {
            CSIS_SetUserUserGroup(array[0], array[2]);
            WScript.echo ("Successfully Set User Group for Ext: " + array[0]);
        }
        catch (e)
        {
            WScript.echo ("Error Setting User Group for Ext: " + array[0] + " : " + e.error);
        }
    }
    else

```



```

if (array[1] == "SH")
{
    try
    {
        CSIS_SendUserHome(array[0]);
        WScript.echo ("Successfully Sent User Home for Ext: " + array[0]);
    }
    catch (e)
    {
        WScript.echo ("Error Sending User Home for Ext: " + array[0] + " : " + e.error);
    }
}
else
{
    WScript.echo("CH or UG not seen in: " + line);
}
}
WScript.Quit();

function CSIS_Connect()
{
    var bRet = 0;
    try
    {
        g_scriptHlpr = new ActiveXObject("Shoreline.ScriptHlp.CScriptHlp.1");
        bRet = g_scriptHlpr.initialize();
    }
    catch (e)
    {
        WScript.echo(e.description);
        bRet = e.error;
    }
    return bRet;
}

function CSIS_Disconnect(hostname, token)
{
    // no implementation
}

function CSIS()
{
}

function CSIS_SetUserCHM(lDN, CallHandlingModeName)
{
    return g_scriptHlpr.setUserCallHandlingMode(lDN, CallHandlingModeName);
}

```



```

function CSIS_SetUserUserGroup(lDN, UserGroupName)
{
    return g_scriptHlpr.setUserUserGroup(lDN, UserGroupName);
}

function CSIS_SendUserHome(lDN)
{
    return g_scriptHlpr.sendUserHome(lDN);
}

this.Connect = CSIS_Connect;
this.SetUserCHM = CSIS_SetUserCHM;
this.SetUserUserGroup = CSIS_SetUserUserGroup;

/*
=====
LTrim(string) : Returns a copy of a string without leading spaces.
=====
*/
function LTrim(str)
/*
    PURPOSE: Remove leading blanks from our string.
    IN: str - the string we want to LTrim
*/
{
    var whitespace = new String(" \t\n\r");

    var s = new String(str);

    if (whitespace.indexOf(s.charAt(0)) != -1) {
        // We have a string with leading blank(s)...

        var j=0, i = s.length;

        // Iterate from the far left of string until we
        // don't have any more whitespace...
        while (j < i && whitespace.indexOf(s.charAt(j)) != -1)
            j++;

        // Get the substring from the first non-whitespace
        // character to the end of the string...
        s = s.substring(j, i);
    }
    return s;
}

```



```

/*
=====
RTrim(string) : Returns a copy of a string without trailing spaces.
=====
*/
function RTrim(str)
/*
    PURPOSE: Remove trailing blanks from our string.
    IN: str - the string we want to RTrim

*/
{
    // We don't want to trip JUST spaces, but also tabs,
    // line feeds, etc. Add anything else you want to
    // "trim" here in Whitespace
    var whitespace = new String(" \t\n\r");

    var s = new String(str);

    if (whitespace.indexOf(s.charAt(s.length-1)) != -1) {
        // We have a string with trailing blank(s)...

        var i = s.length - 1;          // Get length of string

        // Iterate from the far right of string until we
        // don't have any more whitespace...
        while (i >= 0 && whitespace.indexOf(s.charAt(i)) != -1)
            i--;

        // Get the substring from the front of the string to
        // where the last non-whitespace character is...
        s = s.substring(0, i+1);
    }

    return s;
}

/*
=====
Trim(string) : Returns a copy of a string without leading or trailing spaces
=====
*/
function Trim(str)
/*
    PURPOSE: Remove trailing and leading blanks from our string.
    IN: str - the string we want to Trim

    RETVAL: A Trimmed string!
*/
{
    return RTrim(LTrim(str));
}

```



Sample Input File

Reproduced below is a sample input file you can use with the command TestScriptHelp.js *<input file>*.

The format of each line is:

<dn>,*<cmd>*,*<setting>*

where *<cmd>* is either CH (call handling mode) or UG (user group).

```
1500,CH,In a meeting
```

```
1500,UG,House Telephones
```

```
1501,CH,Out of Office
```

```
1501,UG,Anonymous Telephones
```

```
1500,SH,<filler>
```





960 Stewart Drive Sunnyvale, CA 94085 Phone 408-331-3300 10877-80SHORE Fax 408-331-3333 www.shoretel.com