Wavelink Remote Control
Scripting Reference Guide

Version 4.0

av-rc-scripting-rg-20101004

Revised 4/10/2010

Wavelink Corporation
6985 South Union Park Avenue, Suite 335
Midvale, Utah 84047
Telephone: (801) 316-9000
Fax: (801) 316-9099
Email: customerservice@wavelink.com
Website: http:\\www.wavelink.com

Email: sales@wavelink.com

Table of Contents

Introduction

The Remote Control Scripting Reference Guide contains an introduction to script development in the Remote Control Script Editor. This document provides usage information, functions, and example code.

This introduction presents the following information:

- About Remote Control Scripting
- About This Document

About Remote Control Scripting

The Remote Control Script Editor allows you to create and execute JavaScript scripts that automate processes in the Remote Control Web Viewer. It is available when Remote Control is accessed through the Avalanche Web Console.

This section includes the following information:

- Overview of the Scripting Process
- Debugging Scripts
- Additional Information

Overview of the Scripting Process

The following steps outline the process of creating scripts using the Script Editor:

1. **Launch the Script Editor.** You can launch the Script Editor from the Remote Control Web Viewer.
2. **Create scripts using the Script Editor.** Use the Script Editor to manually build the script code.
3. **Execute the script from Remote Control.** While connected to a device, you can execute your script from the **Scripts** tab.

Debugging Scripts

When you run a script with the Remote Control Web Viewer, some debugging information may display underneath the Script Editor when Remote Control

encounters a scripting error. There is currently no log file created for script debugging.

Additional Information

For more information about the Remote Control Script Editor, see the *Wavelink Remote Control User Guide.*

About This Document

This document makes the following assumptions:

- You are familiar with Wavelink Avalanche and Remote Control.
- You know JavaScript.

This table describes how different information is formatted in this document:

Parameters	Parameters are represented in italic Courier New font. The descriptions of the parameters, when necessary, are listed at the right. Example: *Return Value*

Sample code	Sample code is displayed in Courier New font.
	
	Example:
	
	`var wnd = rc.FindWindow("", "Excel Mobile");`
	`if (wnd != 0)`

NOTE If you copy and paste the sample code, you may need to edit out returns.

Using the Script Editor

The Remote Control Script Editor allows you to create scripts and then execute them while you are connected to a device using Remote Control. The scripts must be written in JavaScript. This section provides information on using the Script Editor:

- Accessing the Script Editor
- Creating or Editing Scripts
- Running a Script
- Backing Up Scripts

Accessing the Script Editor

The Remote Control Script Editor is available through the Avalanche Web Console. You can only edit scripts while connected to a device using Avalanche Remote Control.

To access the Script Editor from the Avalanche Web Console:

1. From the **Inventory** tab, click on the name of a device with the Remote Control package installed.
2. In the **Tools** panel, click **Remote Control**.
3. The Remote Control Web Viewer appears.
4. Click on the **Scripts** tab.
5. Select a script from the list, or click **Add New**.

   The JavaScript editor appears. When you select a script from the list or create a new script, the editing panel appears to the right.

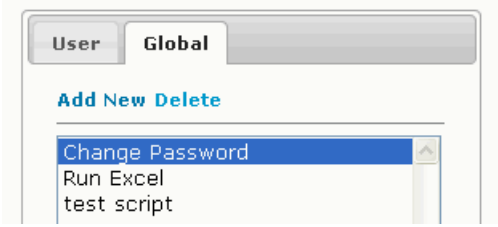
Creating or Editing Scripts

From the Script Editor, you can create a new script or edit an existing script. Scripts can be created as user-specific or global. If the script is user-specific, the only user who will have access to that script will be the Avalanche user who created it. If the script is global, it will be available to all users.

NOTE The script will be available for all devices, regardless of which device you are connected to while you are creating the script.

To create or edit a script:

1 Navigate to the **Scripts** tab of the Remote Control Web Viewer.



2 Select either the **User** or the **Global** tab depending on if the script is user-specific or global.

3 If you are creating a new script, click **Add New**. If you are editing an existing script, select the name of the script you want to edit.

4 In the editing panel, edit the script. Use the **Toggle editor** checkbox at the bottom to toggle whether the editor tools show or not. The tools provide you with a search function, a go-to-line function, undo/redo, font size, word wrap, syntax highlight, and smart display.

NOTE Each script is limited to around 32,000 characters.

5 Save the script regularly by clicking **Save Script**. Regular saving prevents lost information if you are disconnected from the mobile device.

Running a Script

After you have created a script, you can run it on a mobile device.

To run a script on a mobile device:

1 Navigate to the **Scripts** tab of the Remote Control Web Viewer.

2 Select the script from the list and click **Run Script**.

Messages about script performance will appear across the bottom of the editing panel.

Backing Up Scripts

If you want to back up your scripts or review them while not connected to a device, you can use Remote Control to save them as zipped text files.

To back up your scripts:

1. Navigate to the **Scripts** tab of the Remote Control Web Viewer.
2. Click **Download Scripts** (below the list of scripts).
3. A dialog box appears, giving you the option to open or save the .zip file. Enable the **Save file** option and click **OK**.

   Remote Control saves all the scripts in zipped text files that you can view in a text editor.

Overview of Actions

The following tables display an overview of the actions in the Remote Control Script Editor. The actions have been divided into the following sections according to the type of action:

- Mouse
- Window
- Keyboard
- Registry
- Current Device Information
- Current Device
- Notification
- Execution
- Communication
- Server

In addition to the Example sections for each action, there is a chapter containing sample scripts to help with the scripting process. See *Sample Scripts* on page 77 for additional scripting examples.

Mouse

This section contains the following actions related to the mouse:

Action	Description
MouseMove	Moves the mouse to the specified coordinates.
MousePress	Moves the mouse to the specified coordinates and simulates a button press.
MouseRelease	Moves the mouse to the specified coordinates and simulates a button release.
MouseClick	Moves the mouse to the specified coordinates and simulates a button click (press and release).

MouseMove

Moves the mouse to the specified coordinates.

Syntax

```
void MouseMove(int x, int y)
```

Parameters

x	X coordinate
y	Y coordinate

Example

```
// Move mouse to left top coordinate
rc.MouseMove(0,0);
```

MousePress

Moves the mouse to the specified coordinates and simulates a button press.

Syntax

```
void MousePress(int button, int x, int y)
```

Parameters

button	rc.MOUSE_LEFT, rc.MOUSE_RIGHT
x	X coordinate
y	Y coordinate

Example

```
// Move mouse to left top coordinate and press the left button
rc.MousePress(rc.MOUSE_LEFT, 0,0);
```

MouseRelease

Moves the mouse to the specified coordinates and simulates a button release.

Syntax

```
void MouseRelease(int button, int x, int y)
```

Parameters

button	rc.MOUSE_LEFT, rc.MOUSE_RIGHT
x	X coordinate
y	Y coordinate

Example

```
// Move mouse to left top coordinate and release the left button
rc.MouseRelease(rc.MOUSE_LEFT, 0,0);
```

MouseClick

Moves the mouse to the specified coordinates and simulates a button click (press and release).

Syntax

```
void MouseClick(int button, int x, int y)
```

Parameters

button	rc.MOUSE_LEFT, rc.MOUSE_RIGHT
x	X coordinate
y	Y coordinate

Example

```
// Move mouse to left top coordinate and click the left button
rc.MouseClick(rc.MOUSE_LEFT, 0,0);
```

Window

This section contains the following actions related to windows:

Action	Description
SendMessage	Sends the specified message to a window or windows. The SendMessage function calls the window procedure for the specified window and does not return until the window procedure has processed the message.
PostMessage	Places (posts) a message in the message queue associated with the thread that created the specified window and returns without waiting for the thread to process the message.
FindWindow	Retrieves a handle to the top-level window whose class name and window name match the specified strings. This function does not search child windows. This function does not perform a case-sensitive search.
GetClassName	Retrieves the name of the class to which the specified window belongs.
WaitForWindow	Waiting for a specified period of time to give the window a chance to open.
GetForegroundWindow	Retrieves a handle to the foreground window (the window with which the user is currently working).
SetForegroundWindow	Sets the specified window to the foreground.
GetWindowText	Retrieves the text of the specified window's title bar (if it has one). If the specified window is a control, the text of the control is copied.

SendMessage

Sends the specified message to a window or windows. The SendMessage function calls the window procedure for the specified window and does not return until the window procedure has processed the message.

Syntax

```
int  SendMessage (int hWnd, int msg, int wParam, int lParam)
```

Parameters

hWnd	A handle to the window whose window procedure will receive the message. If this parameter is `rc.HWND_BROADCAST (0xffff)`, the message is sent to all top-level windows in the system, including disabled or invisible unowned windows, overlapped windows, and pop-up windows; however, the message is not sent to child windows. To post a message to a thread's message queue and return immediately, use the PostMessage function.
msg	The message to be sent.
wParam	Additional message-specific information.
lParam	Additional message-specific information.

Returns

The return value specifies the result of the message processing; it depends on the message sent.

Example

```
// Find the Excel Mobile window handle
var wnd = rc.FindWindow("", "Excel Mobile");
if (wnd != 0)
{
      // We found it, send it a close message
      rc.SendMessage(wnd, rc.wm.WM_CLOSE, 0, 0);
}
```

PostMessage

Places (posts) a message in the message queue associated with the thread that created the specified window and returns without waiting for the thread to process the message.

Syntax

```
void PostMessage (int hWnd, int msg, int wParam, int lParam)
```

Parameters

hWnd	A handle to the window whose window procedure is to receive the message. The following values have special meanings.
msg	The message to be sent.
wParam	Additional message-specific information.
lParam	Additional message-specific information.

Example

```
// Find the Excel Mobile window handle
var wnd = rc.FindWindow("", "Excel Mobile");
if (wnd != 0)
{
      // We found it, send it a close message
      rc.PostMessage(wnd, rc.wm.WM_CLOSE, 0, 0);
}
```

FindWindow

Retrieves a handle to the top-level window whose class name and window name match the specified strings. This function does not search child windows. This function does not perform a case-sensitive search.

Syntax

```
hwnd FindWindow (String className, String windowName)
```

Parameters

className	The window class name. If className is "", it finds any window whose title matches the windowName parameter.
windowName	The window name (the window's title). If this parameter is "", all window names match.

Returns

If the function succeeds, the return value is a handle to the window that has the specified class name and window name.

If the function fails, the return value is 0.

Example

```
// Find the Excel Mobile window handle
var wnd = rc.FindWindow("", "Excel Mobile");
if (wnd != 0)
{
      //Do something with it.
}
```

GetClassName

Retrieves the name of the class to which the specified window belongs.

Syntax

```
String GetClassName (int hwnd)
```

Parameters

hwnd	A handle to the window and, indirectly, the class to which the window belongs.

Returns

If the function succeeds, the return value is the window class name.

If the function fails, the return value is "".

Example

```
var wnd = rc.FindWindow("Dialog", "Configure");
if (wnd != 0)
{
      // Should display "Dialog"
      rc.MessageBox(rc.GetClassName(wnd));
}
```

WaitForWindow

Waits for a specified period of time to give the window a chance to open.

Syntax

```
int WaitForWindow(String windowName, int secs)
int WaitForWindow(String windowClass, String windowName, int secs)
```

Parameters

className	The window class name. If className is "", it finds any window whose title matches the windowName parameter.
windowName	The window name (the window's title). If this parameter is "", all window names match.
Secs	The number of seconds to wait for the window to be created.

Returns

If the function succeeds, the return value is the window handle.

If the function fails, the return value is 0.

Example

```
// Start your program
rc.Run("\\Program Files\\YourApp\\YourApp.exe")
// Check for your main window, this will wait 5 seconds
var wnd = rc.WaitForWindow("YourApp", "Your Appp Window", 5);

isWindowThere(wnd);

function isWindowThere(wnd)
{
      if (wnd == 0)
      {
            rc.MessageBox("Sorry, something is wrong here!");
            return false;
```

```
        }
        else
        {
// Set one of your controls text
rc.SetDlgItemText(wnd, 1019, "From Our Script");
return true;
}
}
```

GetForegroundWindow

Retrieves a handle to the foreground window (the window with which the user is currently working).

Syntax

```
int GetForegroundWindow ()
```

Returns

The return value is a handle to the foreground window. The foreground window can be null in certain circumstances, such as when a window is losing activation.

Example

```
var wndExcel = rc.FindWindow("", "Excel Mobile");
if (wndExcel != 0)
{
      var hwnd = rc.GetForegroundWindow();
      if (wndExcel == hwnd)
      rc.MessageBox("Excel was on top");
      else
            rc.MessageBox("Excel was not on top");
}
else
{
rc.MessageBox("Excel was not running");
}
```

SetForegroundWindow

Sets the specified window to the foreground.

Syntax

```
int SetForegroundWindow (int hwnd)
```

Parameters

hwnd	A handle to the window that should be activated and brought to the foreground.

Returns

If the window was brought to the foreground, the return value is true.

If the window was not brought to the foreground, the return value is false.

Example

```
// Find the Excel Mobile window handle
var wnd = rc.FindWindow("", "Excel Mobile");
if (wnd != 0)
{
      // Force to the foreground
      rc.SetForegroundWindow (wnd);
}
```

GetWindowText

Retrieves the text of the specified window's title bar (if it has one). If the specified window is a control, the text of the control is copied.

Syntax

```
String GetWindowText (int hwnd)
```

Parameters

hwnd A handle to a window.

Returns

Text of the associated window.

Example

```
// Find the Excel Mobile window handle
var wnd = rc.FindWindow("", "Excel Mobile");
if (wnd != 0)
{
rc.MessageBox(rc.GetWindowText(wnd));
}
else
{
      rc.MessageBox("Sorry, Excel does not seem to be running.");
}
```

Keyboard

This section contains the following actions related to the keyboard:

Action	Description
KeyDown	Sends a simulated key down event.
KeyUp	Sends a simulated key up event.
SendKey	Sends a simulated key down and up event. This is the equivalent of sending the keydown and keyup events.
SendKeys	Sends a string of keyboard events.
Paste	Sends a string of data and pastes it onto the device.

KeyDown

Sends a simulated key down event.

Syntax

```
void KeyDown(char key)
void KeyDown(int key)
```

Parameters

key Either the character or hexadecimal value of the key. Possible values include:

VK_LBUTTON

VK_RBUTTON

VK_CANCEL

VK_MBUTTON // NOT contiguous with L & RBUTTON

VK_XBUTTON1 // NOT contiguous with L & RBUTTON

VK_XBUTTON2 // NOT contiguous with L & RBUTTON

VK_BACK

VK_TAB

VK_CLEAR

VK_RETURN

VK_SHIFT

VK_CONTROL

VK_MENU

VK_PAUSE

VK_CAPITAL

VK_KANA

VK_HANGEUL

VK_HANGUL

VK_JUNJA

VK_FINAL

VK_HANJA

VK_KANJI

VK_ESCAPE

VK_CONVERT

VK_NONCONVERT

VK_ACCEPT

VK_MODECHANGE

VK_SPACE

VK_PRIOR

VK_NEXT

VK_END

VK_HOME

VK_LEFT

VK_UP

VK_RIGHT

VK_DOWN

VK_SELECT

VK_PRINT

VK_EXECUTE

VK_SNAPSHOT

VK_INSERT

VK_DELETE

VK_HELP

VK_0 through VK_9

VK_A through VK_ Z

VK_LWIN

VK_RWIN

VK_APPS

VK_SLEEP

VK_NUMPAD0 through VK_NUMPAD9

VK_MULTIPLY

VK_ADD

VK_SEPARATOR

VK_SUBTRACT

VK_DECIMAL

VK_DIVIDE

VK_F1 through VK_F24

VK_NUMLOCK

VK_SCROLL

VK_OEM_NEC_EQUAL // '=' key on numpad

* Fujitsu/OASYS kbd definitions

VK_OEM_FJ_JISHO // 'Dictionary' key

VK_OEM_FJ_MASSHOU // 'Unregister word' key

VK_OEM_FJ_TOUROKU // 'Register word' key

VK_OEM_FJ_LOYA // 'Left OYAYUBI' key

VK_OEM_FJ_ROYA // 'Right OYAYUBI' key

NOTE VK_L* & VK_R* - left and right Alt, Ctrl and Shift virtual keys used only as parameters to GetAsyncKeyState() and GetKeyState(). No other API or message will distinguish left and right keys in this way.

VK_LSHIFT

VK_RSHIFT

VK_LCONTROL

VK_RCONTROL

VK_LMENU

VK_RMENU

VK_BROWSER_BACK

VK_BROWSER_FORWARD

VK_BROWSER_REFRESH

VK_BROWSER_STOP

VK_BROWSER_SEARCH

VK_BROWSER_FAVORITES

VK_BROWSER_HOME

VK_VOLUME_MUTE

VK_VOLUME_DOWN

VK_VOLUME_UP

VK_MEDIA_NEXT_TRACK

VK_MEDIA_PREV_TRACK

VK_MEDIA_STOP

VK_MEDIA_PLAY_PAUSE

VK_LAUNCH_MAIL

VK_LAUNCH_MEDIA_SELECT

VK_LAUNCH_APP1

VK_LAUNCH_APP2

VK_OEM_1 // ';:' for US

VK_OEM_PLUS // '+' any country

VK_OEM_COMMA // ',' any country

VK_OEM_MINUS // '-' any country

VK_OEM_PERIOD // '.' any country

VK_OEM_2 // '/?' for US

VK_OEM_3 // '`~' for US

VK_OEM_4 // '[{' for US

VK_OEM_5 // '\|' for US

VK_OEM_6 // ']}' for US

VK_OEM_7 // '"' for US

VK_OEM_8

* Various extended or enhanced keyboards

VK_OEM_AX // 'AX' key on Japanese AX kbd

VK_OEM_102 // "<>" or "\|" on RT 102-key kbd.

VK_ICO_HELP // Help key on ICO

VK_ICO_00 // 00 key on ICO

VK_PROCESSKEY

VK_ICO_CLEAR

VK_PACKET

* Nokia/Ericsson definitions

VK_OEM_RESET

VK_OEM_JUMP

VK_OEM_PA1

VK_OEM_PA2

VK_OEM_PA3

VK_OEM_WSCTRL

VK_OEM_CUSEL

VK_OEM_ATTN

VK_OEM_FINISH

VK_OEM_COPY

VK_OEM_AUTO

VK_OEM_ENLW

VK_OEM_BACKTAB

VK_ATTN

VK_CRSEL

VK_EXSEL

VK_EREOF

VK_PLAY

VK_ZOOM

VK_NONAME

VK_PA1

VK_OEM_CLEAR

Example

```
// Send the a key
rc.KeyDown('a');
rc.KeyDown(rc.vk.VK_A);
```

KeyUp

Sends a simulated key up event.

Syntax

```
void KeyUp(char key)
void KeyUp(int key)
```

Parameters

key	Either the character or integer value of the key

Example

```
// Send the a key
rc.KeyUp('a');
rc.KeyUp(rc.vk.VK_A);
```

SendKey

Sends a simulated key down and up event. This is the equivalent of sending the keydown and keyup events.

Syntax

```
void SendKey(char key)
void SendKey(int  key)
```

Parameters

key	Either the character or integer value of the key

Example

```
// Send the a key
rc.SendKey('a');
rc.SendKey(rc.vk.VK_A);
```

SendKeys

Sends a string of keyboard events.

Syntax

```
void SendKeys(String data)
```

Parameters

data	The string of data to be sent

Example

```
// Send the a key
rc.SendKeys("Wavelink RC");
```

Paste

Sends a string of data and pastes it onto the device.

Syntax

```
void Paste(String data)
```

Parameters

data	The string of data to be pasted.

Example

```
// Send the a key
rc.Paste("Wavelink RC");
```

Registry

This section contains the following actions related to the registry:

Action	Description
OpenRegKey	Opens the specified registry key.
CreateRegKey	Creates the specified registry key.
QueryRegKey	Queries the specified registry key.
RegQueryValue	Queries the given key for its value.
DeleteRegValue	Deletes a registry value.
DeleteRegKey	Deletes a registry key.
RenameRegKey	Renames a registry key.
CloseRegKey	Closes the registry key.
RegSetValue	Sets a registry value.

OpenRegKey

Opens the specified registry key.

Syntax

```
int OpenRegKey(int key, String name)
```

Parameters

Key A handle to an open registry key or a pre-defined value. Predefined values are:

rc.reg.HKEY_CLASSES_ROOT

rc.reg.HKEY_CURRENT_CONFIG

rc.reg.HKEY_CURRENT_USER

rc.reg.HKEY_LOCAL_MACHINE

rc.reg.HKEY_USERS

Returns

The handle to the registry key if successful or 0 if the function fails.

Example

```
var hKey = rc.OpenRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\YourApp");
if (hKey != 0)
{
      rc.MessageBox("Registry Key opened");
      rc.CloseRegKey(hKey);
}
else
{
      rc.MessageBox("Registry key not found", "Error", 10,
      rc.mb.MB_ICONSTOP);
}
```

CreateRegKey

Create the specified registry key.

Syntax

```
int CreateRegKey(int key, String name)
```

Parameters

Key	A handle to a newly created registry key or a pre-defined value. Predefined values are: rc.reg.HKEY_CLASSES_ROOT rc.reg.HKEY_CURRENT_CONFIG rc.reg.HKEY_CURRENT_USER rc.reg.HKEY_LOCAL_MACHINE rc.reg.HKEY_USERS

Returns

The handle to the registry key if successful or 0 if the function fails.

Example

```
var hKey = rc.CreateRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\YourApp");
if (hKey != 0)
{
      rc.MessageBox("Registry Key created");
      rc.CloseRegKey(hKey);
}
else
{
      rc.MessageBox("Registry key not created", "Error", 10,
      rc.mb.MB_ICONSTOP);
}
```

QueryRegKey

Queries the specified registry key.

Syntax

```
response QueryRegKey(int key)
```

Parameters

Key	A handle to a newly created registry key or a pre-defined value. Predefined values are: rc.reg.HKEY_CLASSES_ROOT rc.reg.HKEY_CURRENT_CONFIG rc.reg.HKEY_CURRENT_USER rc.reg.HKEY_LOCAL_MACHINE rc.reg.HKEY_USERS

Returns

Registry response object. This object needs to be tested to determine if it is valid. This object contains the following methods:

`boolean isValid();`	Determines if the object is valid or not.
`int getSubKeys();`	Number of keys under the key being queried.
`int getSubKeysLen();`	Longest subkey name length.
`int getValues();`	Number of values associated with the key being queried.
`int getMaxValueNameLen();`	Longest value name length.
`int getMaxValueLen();`	Longest value length.

Example

```
var hKey = rc.OpenRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\Wavelink\\Avalanche");
if (hKey != 0)
```

```
{
	var resp = rc.QueryRegKey(hKey);
	if (resp.isValid())
	{
		var msg = new String().concat(
			"Subkeys: ", resp.getSubKeys(), "\n",
			"Values: ", resp.getValues(), "\n",
			"Max Value Name Len: ",
			resp.getMaxValueNameLen(), "\n",
			"Max Value Len: ", resp.getMaxValueLen()
			);

		rc.MessageBox(msg);
	}
	else
	{
		rc.MessageBox("Something is not right here!");
	}

	rc.CloseRegKey(hKey);
}
```

RegQueryValue

Queries the given key for its value.

Syntax

```
response RegQueryValue(int key, String value)
```

Parameters

key A handle to a newly created registry key or a pre-defined value. Predefined values are:

rc.reg.HKEY_CLASSES_ROOT

rc.reg.HKEY_CURRENT_CONFIG

rc.reg.HKEY_CURRENT_USER

rc.reg.HKEY_LOCAL_MACHINE

rc.reg.HKEY_USERS

Returns

Registry value response object. This object needs to be tested to determine if it is valid. This object contains the following relevant methods:

boolean getValid()	Determines if the object is valid
String value.getTypeString()	String representation of the type of data contained in this object. Valid values are: REG_SZ REG_EXPAND_SZ REG_BINARY REG_DWORD REG_DWORD_BIG_ENDIAN REG_LINK REG_MULTI_SZ REG_QWORD UNKNOWN
int getDWORDData()	Data value
String getName()	Value name
int getType()	Data Type. Valid values are:
String getStringData()	Data value: rc.reg.REG_SZ rc.reg.REG_EXPAND_SZ rc.reg.REG_BINARY rc.reg.REG_DWORD rc.reg.REG_DWORD_BIG_ENDIAN rc.reg.REG_LINK rc.reg.REG_MULTI_SZ rc.reg.REG_QWORD

Example

```
var hKey = rc.OpenRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\Wavelink\\Avalanche");
if (hKey != 0)
{
      var resp = rc.RegQueryValue(hKey, "AmcServerIpAddress");
      if (resp.getValid())
      {
            var msg = new String().concat(
                  "AMC Server : ", resp.value.getDisplayValue(),
                  "\n",
                  "Data Type: ", resp.value.getTypeString(), "\n",
                  "\n"
                  );

            rc.MessageBox(msg);
      }
      else
      {
            rc.MessageBox("Something is not right here!");
      }

      rc.CloseRegKey(hKey);
}
```

DeleteRegValue

Deletes a registry value.

Syntax

```
boolean DeleteRegValue(int key, String name)
```

Parameters

key	A handle to a newly created registry key or a pre-defined value. Predefined values are: rc.reg.HKEY_CLASSES_ROOT rc.reg.HKEY_CURRENT_CONFIG rc.reg.HKEY_CURRENT_USER rc.reg.HKEY_LOCAL_MACHINE rc.reg.HKEY_USERS
name	Value name

Returns

Success returns true, failure returns false.

Example

```
var testdata = "Test Value";
var hKey = rc.CreateRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\ScriptTest");
if (hKey != 0)
{
      rc.RegSetValue(hKey, rc.reg.REG_SZ, "value", testdata );
      var resp = rc.RegQueryValue(hKey, "value");
      if (resp.getValid())
      {
            if (resp.value.getStringData().equals(testdata ))
            {
```

```
                        rc.MessageBox("Value created successfully");
                }
                else
                {
                        rc.MessageBox("Oops... The value is incorrect");
                }
        }
        else
        {
                rc.MessageBox("Failed to create the value");
        }
        rc.Sleep(4000);

        if (rc.DeleteRegValue(hKey, "value"))
        {
                rc.MessageBox("Value deleted successfully");
        }
        else
        {
                rc.MessageBox("Oops... Failed to delete the value");
        }
        rc.CloseRegKey(hKey);
}
else
{
        rc.MessageBox("Sorry, cannot create the registry key");
}
```

DeleteRegKey

Deletes a registry key.

Syntax

```
boolean DeleteRegKey (int key, String name)
```

Parameters

key	A handle to a newly created registry key or a pre-defined value. Predefined values are: rc.reg.HKEY_CLASSES_ROOT rc.reg.HKEY_CURRENT_CONFIG rc.reg.HKEY_CURRENT_USER rc.reg.HKEY_LOCAL_MACHINE rc.reg.HKEY_USERS
name	Key name

Returns

Success returns true, failure returns false.

Example

```
var test_reg_key = "ScriptTest";
var hKey = createTestKey(test_reg_key );
if (hKey != 0)
{
      rc.CloseRegKey(hKey);
}
else
{
      rc.MessageBox("Failed to create reg key");
}
```

```
      if (rc.DeleteRegKey(rc.reg.HKEY_LOCAL_MACHINE, test_reg_key
      ))
{
      rc.MessageBox("Successfully deleted reg key");
}
else
{
      rc.MessageBox("Oops... Something went wrong!");
}
function createTestKey(key)
{
      return rc.CreateRegKey(rc.reg.HKEY_LOCAL_MACHINE, key);
}
```

RenameRegKey

Renames a registry key.

Syntax

```
boolean RenameRegKey(int key, String oldName, String newName)
```

Parameters

key	A handle to a newly created registry key or a pre-defined value. Predefined values are: rc.reg.HKEY_CLASSES_ROOT rc.reg.HKEY_CURRENT_CONFIG rc.reg.HKEY_CURRENT_USER rc.reg.HKEY_LOCAL_MACHINE rc.reg.HKEY_USERS
oldName	Old key name
newName	New key name

Returns

Success returns true, failure returns false.

Example

```
var test_key_old = "ScriptTestOld";
var test_key_new = "ScriptTestNew";
rc.DeleteRegKey(rc.reg.HKEY_LOCAL_MACHINE, test_key_old );
rc.DeleteRegKey(rc.reg.HKEY_LOCAL_MACHINE, test_key_new );
var hKey = rc.CreateRegKey(rc.reg.HKEY_LOCAL_MACHINE,
test_key_old);
if (hKey != 0)
{
      rc.CloseRegKey(hKey);
}
```

```
else
{
      rc.MessageBox("Failed to create reg key");
}
if (rc.RenameRegKey(rc.reg.HKEY_LOCAL_MACHINE, test_key_old,
test_key_new ))
{
      rc.MessageBox("Successfully renamed reg key");
}
else
{
      rc.MessageBox("Oops... Something went wrong!");
}
```

CloseRegKey

Closes the registry key.

Syntax

```
boolean RenameRegKey(int key, String oldName, String newName)
```

Parameters

key A handle to a registry key.

Returns

Success returns true, failure returns false.

Example

See other registry examples.

RegSetValue

Sets a registry value.

Syntax

```
result RegSetValue(int key, int type, String name, String data)
Result RegSetValue(int key, int type, String name, int data)
```

Parameters

key	A handle to a newly created registry key or a pre-defined value. Predefined values are: rc.reg.HKEY_CLASSES_ROOT rc.reg.HKEY_CURRENT_CONFIG rc.reg.HKEY_CURRENT_USER rc.reg.HKEY_LOCAL_MACHINE rc.reg.HKEY_USERS
type	Valid options: rc.reg.REG_SZ rc.reg.DWORD
name	Value Name
data	Value data

Returns

Success returns true, failure returns false.

Example

See other registry examples.

Current Device Information

This section contains the following actions related to current device information:

Action	Description
GetCurrentDevice	This class retrieves information about the currently connected device.
GetDeviceInformation	Retrieves device information about the currently connected device.
GetSystemInformation	Retrieves system information about the currently connected device.

GetCurrentDevice

This class retrieves information about the currently connected device.

Syntax

```
device GetCurrentDevice()
```

Methods Associated with This Class

Device object. This object contains the following methods:

getCommandVersion()	Returns device command processor version (int).
int getLocationId()	Avalanche Location ID
String getLocationName()	Avalanche location description
int getHardwareId()	Avalanche hardware ID
int getAmcDeviceId()	Avalanche device ID
boolean isConnected()	Connection status
String getManufacturer()	Device manufacturer
String getModel()	Device model
String getClientId()	RC Client ID
String getDescription()	Device description
String getIpAddress()	Device IP address
String getName()	Device name
String getOemInfo()	Device OEM information
String getDateLastSeen()	String representation of the last time device was seen
long getDateLastSeen()	Timestamp for when the device was last seen
String getAgentAddress()	Avalanche mobile device server address
boolean isOldClient()	Indicates device has old client loaded
String getAvaTerminalId()	Avalanche terminal ID
String getCarrierId()	Carrier identifier
Carrier getCarrier()	Carrier object
String getPhoneNumber()	Device phone number

Example

```
var mu = rc.GetCurrentDevice();
var msg = new String().concat("Device Info:\n",
      "\n",
      "Manufacturer: ", mu.getManufacturer(), "\n",
      "Device: ", mu.getModel(), "\n",
      "Oem Info: ", mu.getOemInfo(), "\n",
      "IP Address: ", mu.getIpAddress(), "\n",
      "Cmd Ver: ", mu.getCommandVersion(), "\n",
      "Carrier: ", mu.getCarrierId(), "\n",
      "Phone #: ", mu.getPhoneNumber(), "\n",
      "AMC Address: ", mu.getAgentAddress(), "\n",
      ""
      );

rc.MessageBox(msg, "Current Device Information", 5);
```

GetDeviceInformation

Retrieves device information about the currently connected device.

Syntax

```
device GetDeviceInformation ()
```

Returns

Device information object. This object contains the following methods:

int getDeviceBpp();	Number of bits per pixel
int getDeviceNumColors();	Number of colors
int getDevicePlanes();	Number of bit planes
int getWidth();	Device width
int getHeight();	Device height
int getOrientation();	Screen orientation
int getVersion();	Device processor version
int getPlatformType();	0 - CE 1 - Windows
String getDeviceName();	Device name
String getPassword();	Device password
String getOemInfo();	OEM information
String getClientId();	Device ID
String getComputerName();	Device name
String getAgentAddress();	Avalanche mobile device server address
String getAvaTerminalId();	Avalanche terminal ID
String getCarrierId();	Carrier name
String getPhoneNumber();	Device phone number

Example

```
var di = rc.GetDeviceInformation();
```

```
var msg = new String().concat("Device Info:\n",
      "\n",
      "BPP: ", di.getDeviceBpp(), "\n",
      "# Colors: ", di.getDeviceNumColors(), "\n",
      "Platform: ", di.getPlatformType() == 0 ? "CE" : "Windows",
      "\n",
      "Name: ", di.getDeviceName(), "\n",
      ""
      );

rc.MessageBox(msg, "Current Device Information", 5);
```

GetSystemInformation

Retrieves system information about the currently connected device.

Syntax

```
device GetSystemInformation ()
```

Returns

System information object. This object contains the following methods:

int getMajorVersion()	The major version number of the operating system. For example, for Windows CE version 2.10, the major version number is 2.
int getMinorVersion()	The minor version number of the operating system. For example, for Windows CE version 2.10, the minor version number is 1.
int getBuild();	The build number of the operating system or a 0.
int getTotalPhys();	The total number of bytes of physical memory
int getAvailPhys();	The number of bytes of physical memory available
int getPageSize();	The page size and the granularity of page protection and commitment
int getAvailPages();	Available pages
int getStorageSize();	The size, in bytes, of the object store
int getFreeSize();	The amount of free space, in bytes, in the object store
int getCpuType();	The type of processor in the system
byte getBatteryLevel();	Battery level percent
byte getBackupBatteryLevel();	Backup battery level percent
byte getChargingStatus();	0 - not charging 1 - charging
byte getCpuCount();	Number of CPUs
byte getMemoryUtilization();	Memory utilization

Example

```
var si = rc.GetSystemInformation();
```

```
var msg = new String().concat("SystemInfo:\n",
      "\n",
      "OS Ver: ", si.getMajorVersion(), ".", si.getMinorVersion(),
      "\n",
      "Memory: ", si.getAvailPhys(), "/", si.getTotalPhys(), "\n",
      "CPU: ", getCPUDescr(si.getCpuType()), "\n",
      "Battery: ", si.getBatteryLevel(), "\n",
      "Backup Battery: ", si.getBackupBatteryLevel(), "\n",
      "Charging Status: ", si.getChargingStatus(), "\n",
      ""
      );

rc.MessageBox(msg, "Current Device Information", 5);
msg

function getCPUDescr(cpu)
{

      if (cpu ==2577)
            return "STRONGARM";
      if (cpu ==1824)
            return  "ARM720";
      if (cpu ==2080)
            return "ARM820";
      if (cpu ==23360)
            return  "ARM920";
      if (cpu ==70001)
            return  "ARM_7TDMI";
      if (cpu == 386)
            return  "INTEL_386";
      if (cpu ==486)
            return  "INTEL_486";
      if (cpu ==586)
            return  "INTEL_586";
```

```
        if (cpu ==686)
              return  "INTEL_686";
        if (cpu ==4000)
              return  "MIPS_R4000";
        if (cpu ==21064)
              return  "ALPHA_21064";
        if (cpu ==403)
              return  "PPC_403";
        if (cpu ==601)
              return  "PPC_601";
        if (cpu ==603)
              return  "PPC_603";
        if (cpu ==604)
              return "PPC_604";
        if (cpu ==620)
              return  "PPC_620";


        return new String().concat("UNKNOWN(", cpu, ")");
    }
```

Current Device

This section contains the following actions related to the current device:

Action	Description
GetProcesses	Retrieves system information about the currently connected device.
GetDir	Retrieves a handle for the specified director.
Reboot	Performs a warm boot of the device.
Run	Runs the specified executable.
PlaySound	Plays the specified sound file or system sound.
KillProcess	Kills the specified process.

GetProcesses

Retrieves system information about the currently connected device.

Prototype

```
processes GetProcesses ()
```

Returns

Process information vector. This object contains the following methods:

int getPid()

Identifier of the process.

int getHwnd()

Top level window handle.

int getThreads()

Number of execution threads started by the process.

int getPpid()

Identifier of the process that created the process being examined.

int getBasePriority()

Base priority of any threads created by this process.

int getMemoryBase()

Load address of the executable file.

String getName()

The executable file for the process.

String getWindow()

Window text.

Example

```
var proc = rc.GetProcesses().toArray()

var msg = "Whats running<br />";
for (i in proc)
{
      var procName = proc[i].getName();
      msg = msg.concat(procName, "<br />")
}
msg.concat("");
```

GetDir

Retrieves a handle for the specified directory.

Syntax

```
fileInfo GetDir (directory)
fileInfo GetDir(directory, options)
```

Parameters

directory	Directory to get the contents of
options	Options to control what is returned. Predefined values are:
	GETDIR_OPTS_INCLUDE_FILE
	GETDIR_OPTS_INCLUDE_DIRECTORY

Returns

A vector of `fileInfo` objects.

```
fileInfo object;
```

Attributes:

`FILE_ATTRIBUTE_READONLY`	Indicates that the file or directory is read-only. Applications can read the file, but cannot write to it or delete it. In the case of a directory, applications cannot delete it.
`FILE_ATTRIBUTE_HIDDEN`	Indicates that the file or directory is hidden. It is not included in an ordinary directory listing.
`FILE_ATTRIBUTE_SYSTEM`	Indicates that the file or directory is part of the OS or is used exclusively by the OS.
`FILE_ATTRIBUTE_DIRECTORY`	Indicates that the handle identifies a directory.
`FILE_ATTRIBUTE_ARCHIVE`	Indicates that the file or directory is an archive file or directory. Applications use this attribute to mark files for backup or removal.
`FILE_ATTRIBUTE_INROM`	Indicates that this file is an OS file stored in ROM. This type of file is read-only; It cannot be modified.

FILE_ATTRIBUTE_NORMAL	Indicates that the file or directory has no other attributes set. This attribute is valid only if used alone.
FILE_ATTRIBUTE_TEMPORARY	Indicates that the file is being used for temporary storage. File systems attempt to keep all of the data in memory for quicker access, rather than flushing it back to mass storage. The application should delete a temporary file as soon as it is no longer needed.
FILE_ATTRIBUTE_SPARSE_FILE	Indicates that the file is a sparse file.
FILE_ATTRIBUTE_REPARSE_POINT	Indicates that the file has an associated reparse point.
FILE_ATTRIBUTE_COMPRESSED	Indicates that the file or directory is compressed. For a file, this means that all of the data in the file is compressed. For a directory, this means that compression is the default for newly created files and subdirectories.
FILE_ATTRIBUTE_ROMMODULE	Indicates that the file is an OS file stored in ROM and run directly from ROM, rather than being first copied to RAM.

Fuctions:

String getName()	The file/directory name
long getSize()	The file size
int getAttributes()	The file attributes
FileTime getModified()	File modification date/time
boolean isDir()	If this object represents a directory
boolean isFile()	If this object represents a file
boolean isHidden()	If this object represents a hidden directory or file
boolean isReadOnly()	If this object represents a read only file
String getExt()	The file extension

Example

See *Track Files and Directories* on page 79 for an example.

Reboot

Performs a warm boot of the device.

Prototype

```
Reboot ()
```

Example

```
rc.Reboot();
```

Run

Runs the specified executable.

Prototype

```
Run (String exeName)
```

Parameters

exeName Executable to run

Example

```
rc.Run("pxl")
var wnd = rc.WaitForWindow("Excel Mobile", 5)
if (wnd > 0)
{
      rc.PlaySound("Alarm4.wav")
      rc.Sleep(2000);
      rc.PostMessage(wnd, rc.wm.WM_CLOSE, 0, 0)
}
"Done"
```

PlaySound

Plays the specified sound file or system sound.

Prototype

```
PlaySound(String sound)
```

Parameters

sound	Sound file to play or system sound such as: SystemAsterisk SystemExclamation SystemExit SystemHand SystemQuestion SystemStart Etc.

Example

```
rc.PlaySound("Alarm4.wav")
rc.PlaySound("SystemExclamation")
```

KillProcess

Kills the specified process.

Prototype

```
KillProcess(int pid)
```

Parameters

pid Process ID for the process to kill

Example

```
var proc = rc.GetProcesses().toArray()
var found = false;
for (i in proc)
{
      var procName = proc[i].getName();
      if (procName == "BubbleBreaker.exe")
      {
            rc.KillProcess(proc[i].getPid());
            found = true;
            break;
      }
}

if (found == true)
      rc.MessageBox("BubbleBreaker killed");
else
      rc.MessageBox("BubbleBreaker not found");
```

Notification

This section contains the following action related to notifications:

Action	Description
MessageBox	Displays a notification message box on the device.
SetDlgItemText	Sets the title or text of a control in a dialog box.

MessageBox

Displays a notification message box on the device.

Prototype

```
void MessageBox(String msg)
void MessageBox(String msg, String title, int seconds, int flags)
```

Parameters

msg The message to be displayed

title Title of message box

seconds Number of seconds to display message box

Flags Specifies a set of bit flags that determine the contents and behavior of the dialog box. This parameter can be a combination of flags from the following groups of flags.

Specify one of the following flags to display an icon in the message box:

rc.mb.MB_ICONEXCLAMATION rc.mb.MB_ICONWARNING	An exclamation-point icon appears in the message box.
rc.mb.MB_ICONINFORMATION rc.mb.MB_ICONASTERISK	An icon consisting of a letter “ i” in a circle appears in the message box.
rc.mb.MB_ICONQUESTION	A question-mark icon appears in the message box.
rc.mb.MB_ICONSTOP rc.mb.MB_ICONERROR rc.mb.MB_ICONHAND	A stop-sign icon appears in the message box.

Specify one of the following flags to define the modality of the message box. If no modality flag is specified, the message box will default to rc.mb.MB_APPLMODAL.

rc.mb.MB_APPLMODAL	The user must respond to the message box before continuing work in the window identified by the hWnd parameter. However, the user can move to the windows of other threads and work in those windows. Depending on the hierarchy of windows in the application, the user may be able to move to other windows within the thread. All child windows of the parent of the message box are automatically disabled, but popup windows are not.
rc.mb.MB_SETFOREGROUND	The message box becomes the foreground window. Internally, the system calls the SetForegroundWindow function for the message box.
rc.mb.MB_TOPMOST	The message box is created with the WS_EX_TOPMOST window style.

Example

```
rc.MessageBox("Testing");
rc.Sleep(4000);
rc.MessageBox("Testing", "Title", 10, rc.mb.MB_ICONSTOP);
```

SetDlgItemText

Sets the title or text of a control in a dialog box.

Syntax

SetDlgItemText(dialog, item, text)

Parameters

dialog	A handle to the dialog box that contains the control.
item	The control with a title or text to be set.
text	The text to be copied to the control.

Example

```
//Launch your app
rc.Run("\\Program Files\\YourAppDir\\YourApp.exe")

//Wait for main window to be active
var wnd = rc.WaitForWindow("", "Your Window", 5);
if (wnd == 0)
{
     rc.MessageBox("Sorry, something is wrong here!");
}
else
{
     // Set the control with id 1001 to some text
rc.SetDlgItemText(wnd, 1001, "Your new text here");
}
```

Execution

This section contains the following action related to execution:

Action	Description
Sleep	Causes the script execution to pause.

Sleep

Causes the script execution to pause.

Prototype

```
void Sleep(int waitTime)
```

Parameters

waitTime	Number of milliseconds to delay

Example

```
// Sleep 2 seconds
rc.Sleep(2000);
```

Communication

This section contains the following actions related to communication:

Action	Description
SendMail	Sends an e-mail message.
SendSnapshot	Sends an snapshot of the device screen as an attachment to the specified recipient.
SendSms	Sends an SMS message to the specified phone number.

SendMail

Sends an e-mail message. You must have e-mail settings in Avalanche configured in order for this to work.

Prototype

```
boolean SendMail(String recipient, String subject, String body)
```

Parameters

recipient	E-mail receipant
subject	E-mail subject
body	The body text of the e-mail

Returns

Success returns true, failure returns false.

Example

```
rc.SendMail("support@wavelink.com", "SendMessage Test", "Test
Message");
```

SendSnapshot

Sends an snapshot of the device screen to the specified recipient as an attachment. You must have e-mail settings in Avalanche configured in order for this to work.

Prototype

```
boolean SendSnapshot(String recipient,String subject, String body)
```

Parameters

recipient	Email recipient
subject	Email subject
body	The body text of the email

Returns

Success returns true, failure returns false.

Example

```
rc.SendSnapshot("support@wavelink.com", "Your Screen", "Here is
the device screen");
```

SendSms

Sends an SMS message to the specified phone number. You must have SMS options in Avalanche configured for this to work.

Prototype

```
boolean SendSms(String phoneNumber, String carrierName, String
body)
```

Parameters

phoneNumber	Phone number to send message to
carrierName	Telephone carrier
body	The body text of the email

Returns

Success returns true, failure returns false.

Example

```
var mu = rc.GetCurrentDevice();
var msg = new String().concat("Device Info:\n",
      "\n",
      "Manufacturer: ", mu.getManufacturer(), "\n",
      "Device: ", mu.getModel(), "\n",
      "Oem Info: ", mu.getOemInfo(), "\n",
      "IP Address: ", mu.getIpAddress(), "\n",
      "Cmd Ver: ", mu.getCommandVersion(), "\n",
      "Carrier: ", mu.getCarrierId(), "\n",
      "Phone #: ", mu.getPhoneNumber(), "\n",
      "AMC Address: ", mu.getAgentAddress(), "\n",
      ""
      );

var carrier = new String(mu.getCarrierId());
```

```
var phone = new String(mu.getPhoneNumber());

if (phone != "")
{
      rc.SendSms(phone, carrier, msg);
}
else
{
      msg = new String().concat("Hey, the following device has no
      phone number",msg);
      rc.SendSms("2065551212", "T-Mobile", msg);
}
```

Server

This section contains the following actions related to the server:

Action	Description
GetLicenseCounts	Gets Remote Control licensing information.
GetLogger	Gets a logger for writing to the Remote Control log files.

GetLicenseCounts

Gets Remote Control licensing information.

Prototype

```
licenseCounts GetLicenseCounts()
```

Returns

licenseCounts object

int getAvailableCount()

Number of available Remote Control licenses

int getTotalCount()

Total Remote Control licenses

Example

```
var lc = rc.GetLicenseCounts();
rc.SendMail("sales@wavelink.com",
      "RC License count info",
      new String().concat(
            "Hey Wavelink, we need to buy some more licenses!\n",
            "We only have ", lc.getAvailableCount(),
            " licenses out of ",
            lc.getTotalCount(), " left")
      );
```

GetLogger

Gets a logger for writing to the Remote Control log files.

Prototype

```
logger GetLogger()
logger GetLogger(String name)
```

Parameters

name	Logger name

NOTE To use directed logging, you must be familiar with log4j logging and manually configure the configuration file.

Returns

Logger object

Example

```
var log = rc.GetLogger();

log.debug("This is a debug message");
log.info("This is a informational message");
log.error("This is a error message");
log.fatal("This is a fatal message");
```

Sample Scripts

In addition to the example code shown for each action, the following sample scripts use JavaScript to perform functions using the Remote Control JavaScript editor. The sample scripts include:

- Displaying Registry Information
- Track Files and Directories
- Start a Process
- Email Screenshot
- Email Device Information
- Send a Keyboard Command
- Enumerate Processes and Kill Non-System Processes
- Transfer and Play .wav File
- Change Registry Key and Reboot
- Change Registry Value
- Change the Remote Control Password
- FizzBuzz

Displaying Registry Information

This script displays a message on the device containing Avalanche server information. It reads this information out of the device registry.

```
var hKey = rc.OpenRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\Wavelink\\Avalanche");
if (hKey != 0)
{
      var val = rc.RegQueryValue(hKey, "AmcServerVersion");
      var serverVersion = new
      String(val.value.getStringData());


      val = rc.RegQueryValue(hKey, "AmcServerIpAddress");
      var serverAddress = new
      String(val.value.getStringData());


      rc.CloseRegKey(hKey);


      var msg = new String().concat("AMC Info:\n",
      "\n",
      " Server: ", serverAddress, "\n",
      "Version: ", serverVersion, "\n",
      "\n"
      );


      rc.MessageBox(msg, "AMC Server Info", 5);


}
else
{
      rc.MessageBox("Sorry, I cannot find the info");
}
```

Track Files and Directories

This script counts all files and directories for a given directory as well as the total size of all files. In addition, it keeps track of the largest file. For the first file encountered, the script displays detailed information on the device screen. Once all files and directories have been traversed, it displays a message on the device containing the information collected about the directory.

```
var directory = "\\temp\\";

var dir = rc.GetDir(directory).toArray()
var files = 0;
var dirs = 0;
var size = 0;
var largest_size = 0;
var largest_name = "";
var isFirst = true;

for (i in dir)
{
      var fName = dir[i].getName();
      if (isFirst == true && dir[i].isFile())
      {
            showFirstFile(dir[i]);
            isFirst = false;
      }

      if (dir[i].getSize() >= largest_size)
      {
            largest_size = dir[i].getSize();
            largest_name = fName;
      }

      size += dir[i].getSize();
      if (dir[i].isFile())
```

```
            files++;
      else
            dirs++;
}


var message = new String().concat("There are ", (files+dirs),
" files in ", directory, "\n", files, " are files and ",
dirs, " are directories\n", "they total ", size, " bytes",
"\n\n", "the largest file is ", largest_name, " at ",
largest_size, " bytes."

);
rc.MessageBox(message);


function showFirstFile(f)
{
var msg = new String().concat("Here is the first file info:\n",
"name: ", f.getName(), "\n", "size: ", f.getSize(), "\n",
"modified:", f.getModified()

);
      rc.MessageBox(msg);
      rc.Sleep(4000);
}
```

Start a Process

This script demonstrates how to start a process and wait for the window to become active.

```
rc.Run("pxl")
var wnd = rc.WaitForWindow("Excel Mobile", 5)
if (wnd > 0)
{
      rc.PlaySound("Alarm4.wav")
      rc.Sleep(2000);
      rc.PostMessage(wnd, rc.wm.WM_CLOSE, 0, 0)
}
"Done"
```

Email Screenshot

This script grabs a snapshot of the current screen and e-mails it to a specific e-mail address.

NOTE You must have e-mail options configured in Avalanche before you can use `SendSnapshot`.

```
rc.SendSnapshot("somebody@yourcompany.com",
      "Script Snapshot",
      "Here is the screen you wanted!");
```

Email Device Information

This script retrieves basic information from a device (IP, mac, battery, etc.), and e-mails it to support@wavelink.com

```
var mu = rc.GetCurrentDevice();

var msg = new String().concat("Device Info:\n",
      "\n",
      "Manufacturer: ", mu.getManufacturer(), "\n",
      "Device: ", mu.getModel(), "\n",
      "Oem Info: ", mu.getOemInfo(), "\n",
      "Battery Level:",
      rc.GetSystemInformation().getBatteryLevel(), "\n",
      "IP Address: ", mu.getIpAddress()
      );

rc.SendMail("support@wavelink.com", "Device Info", msg);
```

Send a Keyboard Command

This script sends a complex keyboard combination like CTRL+ALT+SHIFT+K.

```
rc.SendKey(rc.vk.VK_K| rc.vk.SHIFT_STATE | rc.vk.CTRL_STATE |
rc.vk.ALT_STATE);
```

Enumerate Processes and Kill Non-System Processes

This script enumerates all processes and kills all the non-system processes. (This may cause session disconnect.)

```
var proc = rc.GetProcesses().toArray()

var msg = "Oh my, what to kill!<br><br>"
for (i in proc)
{
      var killMsg = "";
      var procName = proc[i].getName();
      if (procName != "NK.EXE" &&
            procName != "filesys.exe" &&
            procName != "device.exe" &&
            procName != "shell32.exe" &&
            procName != "gwes.exe" &&
            procName != "services.exe"
            )
     {
            killMsg = new String().concat(" << KILL (",
            proc[i].getPid(), ")");

            //Feel free to uncomment the following if you want
            //actual kills
            //rc.KillProcess(proc[i].getPid());
      }
      msg = msg.concat(procName, killMsg, "<br />")
}
msg.concat("")
```

Transfer and Play .wav File

This script transfers a `.wav` file to the device and plays it on the device twice.

```
var dest = "\\temp\\Infend.wav";
if (rc.PutFile("c:\\temp\\Infend.wav", dest))
{
      rc.PlaySound(dest);
      rc.Sleep(1000);
      rc.PlaySound(dest);
}
```

Change Registry Key and Reboot

This script changes the registry key `HKEY_LOCAL_MACHINE\Comm\Tcpip\Parms` value `KeepAliveTime` to 1000 ms and reboots the device.

```
var hKey = rc.OpenRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Comm\\Tcpip\\Parms");
if (hKey != 0)
{
      rc.RegSetValue(hKey, rc.reg.REG_DWORD, "KeepAliveTime",
      1000);
      rc.CloseRegKey(hKey);
      rc.MessageBox("Your TCP/IP Params have changed.
      Rebooting...");
      rc.Sleep(5000);
      rc.Reboot();
}
else
      rc.MessageBox("Sorry, you don’t have any TCP/IP params");
```

Change Registry Value

This script reads the `counter` value from `HKLM\Software\Wavelink\RemoteControlTest`, increments it, and stores the new value.

```
var hKey = rc.OpenRegKey(rc.reg.HKEY_LOCAL_MACHINE,
"Software\\Wavelink\\RemoteControlTest");
if (hKey != 0)
{
      var oldVal = 0;
      var val = rc.RegQueryValue(hKey, "counter");
      if (val.getValid())
      {
            oldVal = val.value.getDWORDData();
      }
      ar newVal = oldVal + 1;
      rc.RegSetValue(hKey, rc.reg.REG_DWORD, "counter",
      newVal);
      rc.CloseRegKey(hKey);

      var msg = new String().concat("Counter Info:\n",
      "\n",
      "Old Value: ", oldVal , "\n",
      "New Value: ", newVal , "\n",
      "\n"
      );

      rc.MessageBox(msg, "Counter Test", 5);

}
else
{
      rc.MessageBox("Sorry I can not find the registry key");
}
```

Change the Remote Control Password

This script opens the Remote Control window on the device, opens the configuration menu, and changes the password to “Clepsydra”.

```
rc.Run("\\Program
Files\\Wavelink\\Avalanche\\Apps\\WLRMTCTL\\WLRCTrigger.exe")
var wnd = rc.WaitForWindow("WLRC", "Wavelink RC", 5);
if (wnd == 0)
{
      rc.MessageBox("Sorry, something is wrong here!");
}
else
{
      rc.PostMessage(wnd, rc.wm.WM_COMMAND, 126, 0);
      var configWnd = rc.WaitForWindow("Dialog", "Configure",
      5);
      if (wnd == 0)
      {
            rc.MessageBox("Could not access configuration
            screen.");
      }
      else
      {
            rc.SetDlgItemText(configWnd, 1019, "Clepsydra");
            rc.PostMessage(configWnd, rc.wm.WM_COMMAND, 1, 0);
      }
}
```

FizzBuzz

This script outputs the numbers from 1 to 100, replacing the numbers divisible by 3 with the word "Fizz" and the numbers divisible by 5 with the word "Buzz", and the numbers divisible by 3 and 5 with the word "FizzBuzz".

```
var myCount=new Array();
var x;
for (x=1; x < 100; x++)
{
      myCount[x-1] = "";
      if ((x % 3) == 0)
            myCount[x-1] = "Fizz";
      if ((x % 5) == 0)
            myCount[x-1] += "Buzz";
      if (myCount[x-1] == "")
            myCount[x-1] = x;
}
myCount.toString();
```

Appendix: Wavelink Contact Information

If you have comments or questions regarding this product, please contact Wavelink Customer Support.

E-mail Wavelink Customer Support at: CustomerService@wavelink.com

For customers within North America and Canada, call the Wavelink Technical Support line at 801-316-9000 (option 2) or 888-699-9283.

For international customers, call the international Wavelink Technical Support line at +800 9283 5465.

For Europe, Middle East, and Africa, hours are 9 AM - 5 PM GMT.

For all other customers, hours are 7 AM - 7 PM MST.