Wavelink Telnet Client Scripting Reference Guide

tn-rg-script-20091006

Revised 10/06/2009

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

Chapter 1: Introduction

This document provides information about creating and executing scripts using Telnet Client. This chapter provides an overview of the document and scripting features. It includes:

- Document Assumptions
- Document Conventions
- About Telnet Client Scripting

Document Assumptions

This document is written with the assumption the reader and user of the Telnet Client possess:

- Knowledge of wireless networks and wireless networking protocols.
- Knowledge of TCP/IP, including IP addressing, subnet masks, routing, BootP/DHCP, WINS, and DNS.
- Knowledge of Wavelink Telnet Client.
- Knowledge or rudimentary experience with programming/scripting languages.

Document Conventions

The following table lists the document conventions used in this manual.

`Courier New`	Any time you type specific information into a text box (such as a file name), that option appears in the `Courier New` text style. This text style is also used for any keyboard commands that you might need to press. Examples: Type `Enter` to continue. Press `CTRL+ALT+DELETE`.
Bold	Any time you interact with an option (such as a button or descriptions of different options in a dialog box), that option appears in the **Bold** text style. Examples: Click **Open** from the **File** Menu. Select the **Update** option.
Italics	Any time this document refers to another section within the document, that section appears in the *Italics* text style. This style is also used to refer to the titles of dialog boxes. Examples: See *Naming Scripts* on page 19 for more information. The *Script Editor* dialog box appears.

About Telnet Client Scripting

The Script Editor is a component of the Wavelink Telnet Client. The Script Editor provides the ability to create and execute scripts that automate processes on the Telnet Client. The Script Editor is included in Telnet Client 5.1 and later versions.

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 Telnet Client, or from the Avalanche Console.

2 **Create scripts using the Script Editor.** You can use the Script Editor to manually build the script code.

-Or-

Create scripts using the Script Capture option. Capture the actions you want to include in your script to build the script code.

-Or-

Create scripts from text. Import a text file or create a text-based script using the Text Editor.

3 **Configure an execution method for the script.** Select from the available options the method you want to use to execute your script.

4 **Execute the script from the Telnet Client.** Using the activation method you selected for the script, you can activate and execute your script.

NOTE You can only run scripts while a Telnet session is connected to a host. If the connection drops, the script is terminated. If you switch between sessions, the script running in the first session remains suspended until that session is active again.

Scripting Actions Library

The actions and settings you will use to create Telnet Client scripts are located in a separate document called *Wavelink Telnet Client Scripting Commands Library*. The document explains the usage of each action in detail and provides examples for your reference. The *Scripting Commands Library* is located on the Wavelink web site.

Terminology

For conciseness and clarity, the term **Avalanche Console** applies to both Avalanche MC and Avalanche SE. This document will reference Avalanche Console for both products. For more information about each product, refer to the specific user guide.

Chapter 2: Launching the Script Editor

If you are using Avalanche Console to deploy the Telnet Client, you can launch the Script Editor from the Avalanche Console. Scripts created by or imported into the Script Editor are automatically deployed to the remote mobile devices.

To launch the Script Editor from Avalanche Console:

1. Ensure that the Telnet Client package is installed in the Avalanche Console.
2. Launch the Avalanche Console.
3. In the **Software Profiles** tab, locate the profile that contains the Telnet Client software package.
4. In the **Software Packages** tab, select the Telnet Client software package and click **Configure**.

   The *Configure Package* dialog box appears.
5. From the menu list, select **Script Editor** and click **OK**.

   The Script Editor appears.
6. Click **Add** to access the *Script Editor Configuration* dialog box.

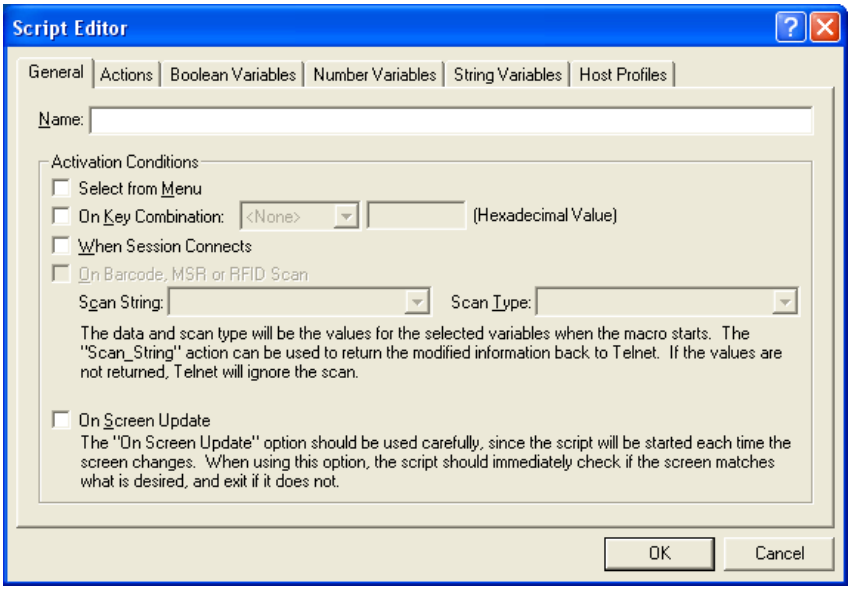


Figure 2-1. *Script Editor Configuration Dialog Box*

From this dialog box, you can configure and create scripts. For information about script creation, refer to *Chapter 3: Creating Scripts* on page 17.

Chapter 3: Creating Scripts

There are three ways you can create scripts:

- **Manually**. Using this method, you build the script code from scratch using the Script Editor.
- **Script Capturing**. Using this method, you generate the script code by enabling script capturing and performing the action you want the script to automate. The Script Editor records the key presses and cursor movements and saves the information as script code. You can edit the captured code to customize the way the script runs.
- **From Text.** Using this method, you generate the script code by importing text from any text editor or entering text in the Script Editor.

Once you create a script, you can save and deploy the script to the mobile devices you want to run the script.

This chapter provides information about creating scripts, including:

- Building Scripts Manually
- Performing Script Capturing
- Creating Scripts From Text
- Editing Scripts
- Importing Scripts
- Saving and Exporting Scripts
- Deploying Scripts
- Syncing Scripts
- Creating Log Files
- Script Nesting
- Field Data ID Feature

NOTE Screen captures in this document may differ according to device type.

Building Scripts Manually

When building a script manually, you build the code for the script line by line in the Script Editor. This section provides following information:

- Configuration Overview
- Naming Scripts
- Selecting Activation Methods
- Creating Script Code
- Creating Variables
- Selecting Host Profiles

Configuration Overview

The following steps provide an overview of how you create a script. The subsequent sections in this chapter describe each step in more detail.

To build a script:

1. Name the script.
2. Select an activation method.
3. Build the script code. In the **Actions** tab, create the code, line-by-line, that describes the actions you want the script to perform.
4. Create any variables that you need for your script in the **Boolean Variables**, **Number Variables**, or **String Variables** tabs.
5. Assign host profiles that can perform the script.

   Your script is complete and ready to activate upon a Telnet Client-to-host connection.

Naming Scripts

Give scripts a unique name according to the action the script performs.The script name is the name you will select from when activating the scripts. You can name your script in the **General Settings** tab.

To name scripts:

1 Launch the Script Editor.

2 Click **Add** to access the *Script Editor Configuration* dialog box.

3 In the **General Settings** tab, enter the name of the script.

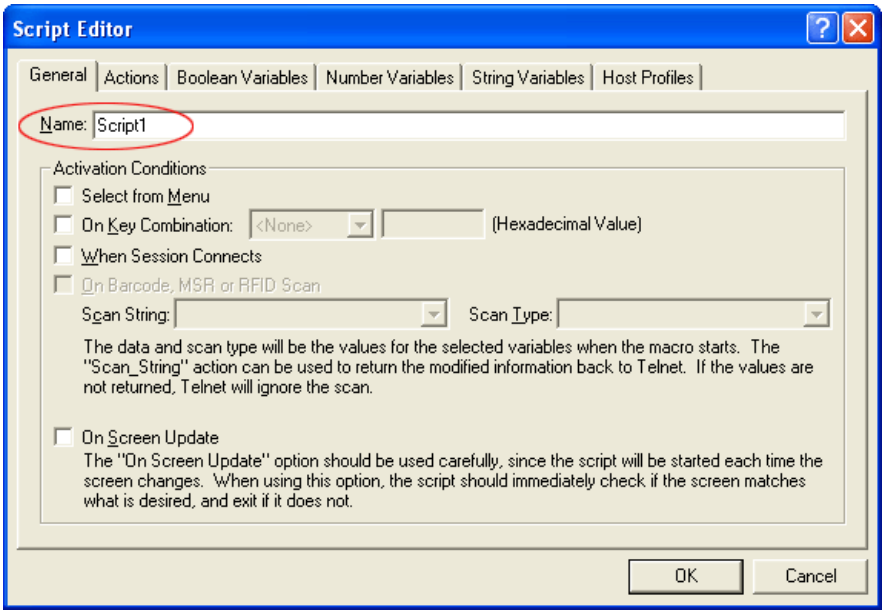


Figure 3-1. *Entering the Script Name*

4 Click **OK** to save the name and exit the *Script Editor Configuration* dialog box.

Selecting Activation Methods

The Activation Method determines the way you execute the script from the Telnet Client. A script with no activation method selected can only be called by another script. It cannot activate alone.

Activation methods are selected in the **General Settings** tab of the *Script Editor Configuration* dialog box. This section provides information about the different activation methods including:

- Select from Menu
- On Key Combination
- When Session Connects
- On Barcode, MSR or RFID Scan
- On Screen Update

Select from Menu

The **Select from Menu** option places a script execution selection in the Telnet Client menu.

To configure the Select from Menu method:

1. Select the **General** tab or the **Activate** tab in the Script Editor.
2. Enable the **Select from Menu** option.

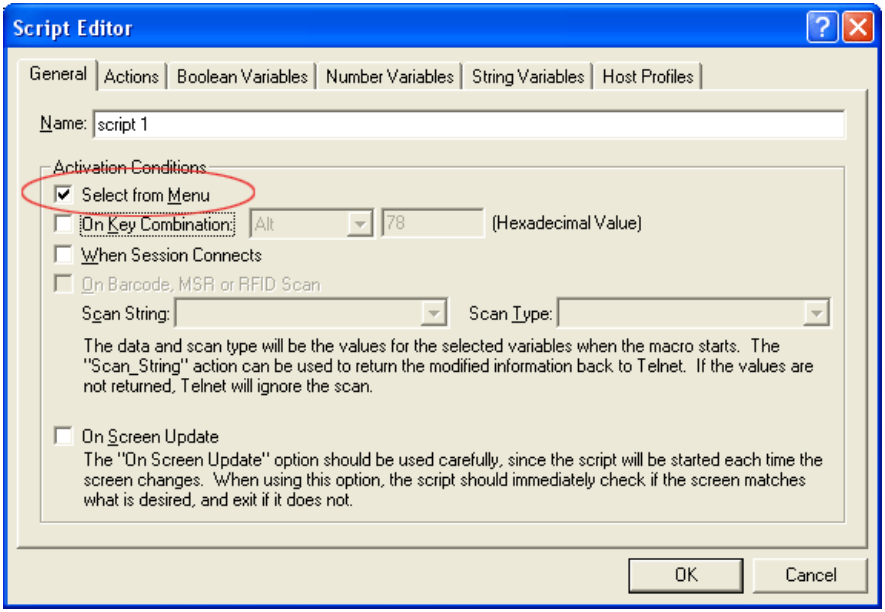


Figure 3-2. *Select from Menu*

3 Click **OK**.

On Key Combination

If you enable the **On Key Combination** option, you can launch the script by pressing the specified keys.

Use the Diagnostic Utility to obtain the key value. For more information about using the Diagnostic Utility refer to the *Wavelink Telnet Client User Guide*.

To configure the On Key Combination method:

1 Select the **General** tab or **Activate** tab in the Script Editor.

2 Enable the **On Key Combination** option.

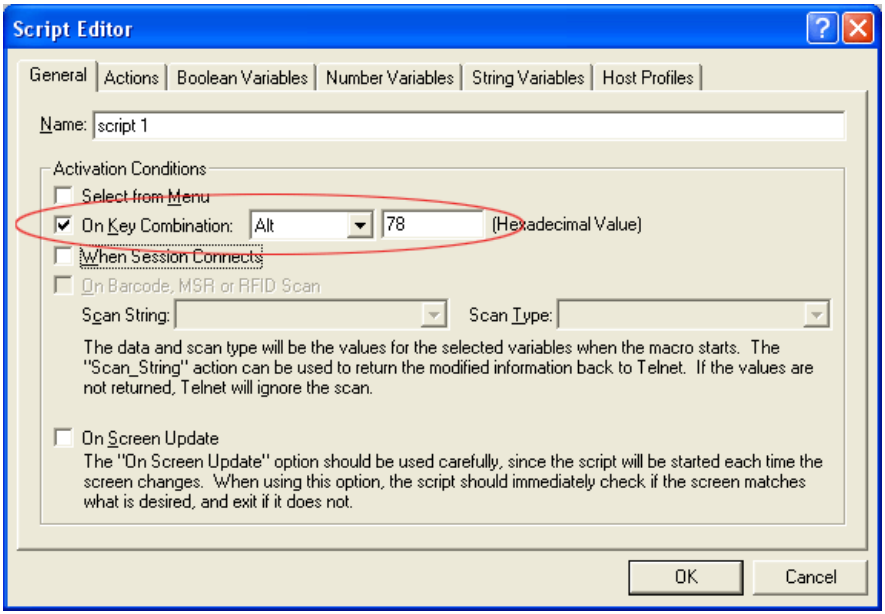


Figure 3-3. *On Key Combination*

3 Use the drop-down menu and text box to assign a key combination to the script.

4 Click **OK**.

When Session Connects

If you enable the **When Sessions Connects** option, the script activates when the host profile it supports is activated.

If you use this option, Wavelink strongly recommends that you limit the script to the appropriate host profiles. Because the script activates before any information appears on the emulation screen, you need to have your script wait for the appropriate screen to appear before activates. You should not have more than one script set to start when a session connects because the first script that starts will prevent any other scripts from running while it waits for the initial screen. Refer to *Selecting Host Profiles* on page 21 for more information.

To configure the When Session Connects method:

1 Select the **General** tab or **Activate** tab in the Script Editor.

2 Enable the **When Session Connects** option.

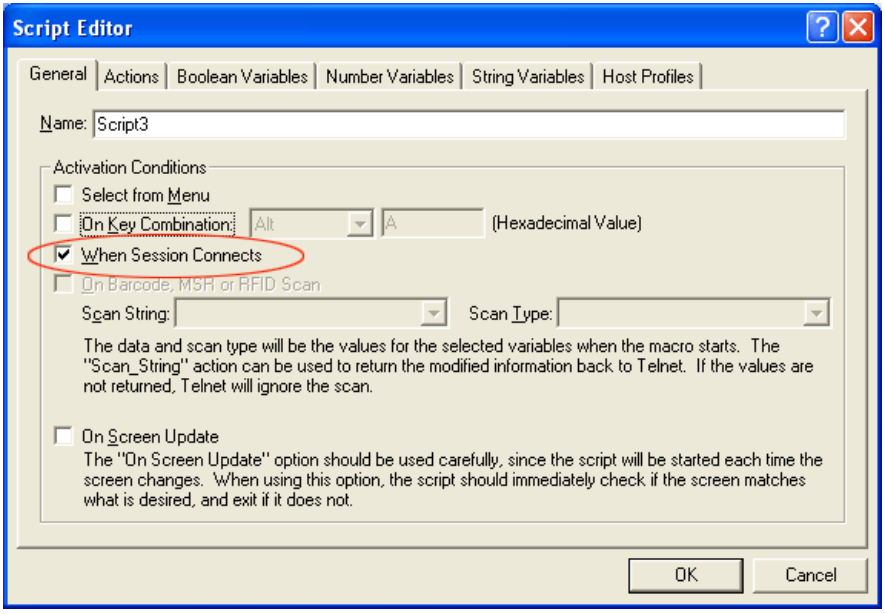


Figure 3-4. *When Session Connects*

3 Click **OK**.

On Barcode, MSR or RFID Scan

If you want to perform special processing on items scanned into the computer, the Scan Handler is often powerful enough to make the changes you need. The Scan Handler settings, found in the Configuration Manager, are located in **Emulation Parameters > Scanner > Common > Scan Handler**. However if the Scan Handler is insufficient, you can use a script to perform any processing you need.

Before you can activate the script for a scan, you must create a string variable and a number variable.

- The string variable allows you to obtain the initial scan data.
- The number variable allows you to obtain the type of scan data.

Refer to *Wavelink Telnet Client Scripting Actions Library* for the values of different symbologies. You can also use the `Get_Scan_Type_Name` and

`Get_Scan_Type_Value` commands to display or handle scan types. Using the `Get_Scan_Type_Value` means all types are specified in the editor and you can select the type you want to use.

Calling the `Scan_String` command before your script exits allows Telnet to handle the scanning data. Because you are specifying the data and type returned, the script can change either one. If the script exits without calling `Scan_String`, the scanned data disappears.

To configure the On Barcode, MSR, or RFID Scan method:

1. Create the `Scan_String` and `Scan_Type` variables.

   Once you create these variables, the **On Barcode, MSR or RFID Scan** option becomes available.

   Create these variables in the String Variables and Number Variables tabs. Refer to *Creating Variables* on page 19 for information on creating variables.

2. Select the **General** tab or **Activate** tab in the Script Editor.
3. Enable the **On Barcode, MSR or RFID Scan** option.
4. From the drop-down menu, select the **Scan_String**.
5. From the drop-down menu select the **Scan_Type**.
6. Click **OK**.

On Barcode, MSR or RFID Scan Example 1

The following is a sample script you can use if you want to insert a string (which could be just one character long) after the first six characters of any barcode at least six characters long.

A few notes about the sample script:

- **ScanData** is a string variable with the original barcode.
- **NewString** is a variable where you store the new barcode.
- **ScanType** is the number variable that keeps the type of scan data received.
- **OldLength** is an integer variable.

- **XXYY** is the string you insert.

```
  OldLength=String_Length(ScanData)
  If (Number_Greater_Than_Or_Equal(OldLength,6))
NewString=String_Combine(String_Left(ScanData,6), "XXYY"
)
  NewString =
String_Combine(NewString,String_Right(ScanData,
Number_Minus(OldLength,6)))
  Else
  NewString = ScanData
  End_If
  Scan_String(NewString,ScanType)
  Return
```

On Barcode, MSR or RFID Scan Example 2

This example converts any DataMatrix scan values to PDF417 scan values. The **ScanData** and **ScanType** variables described for the previous example are used again.

```
  If
(Number_Equal(ScanType,Get_Scan_Type_Value("DATAMATRIX"))
)
  Scan_String(ScanData,Get_Scan_Type_Value("PDF417"))
  Else
  Scan_String(ScanData,ScanType)
  End_If
  Return
```

On Screen Update

This option activates the script (if activation is allowed) every time the text on the emulation screen changes. This includes updates from the Telnet host or when the user presses a key and the key value is shown on the screen. It is recommended that you limit the host profiles that the script supports.

The following example generates a script that enters a command each time a particular string appears on the screen:

```
  Label:Start:
  If
(String_Equal(Get_Screen_Text_Columns(1,1,5),"Ready",
0,FALSE))
  Keypress_String("Proceed")
  Keypress_Key("Enter")
  End_If
  Wait_For_Screen_Update
  Goto: Start
  Return
```

If the script is set to activate when the session first connects, it will work as desired with one limitation. Because it is always activated, no other scripts can be activated during the emulation session.

Here is an alternate implementation:

```
  If (String_Equal(Get_Screen_Text_Columns(1,1,5),
“Ready”, 0, FALSE))
  Keypress_String(“Proceed”)
  Keypress_Key(“Enter”)
  End_If
  Return
```

If the script is set to run each time the screen updates, you get the desired behavior. Because the script is not active all the time, other scripts can be activated as well.

NOTE Use this option carefully as it can cause a script to be executed frequently.

To configure the On Screen Update method:

1. Select the **General** tab or **Activate** tab in the Script Editor
2. Enable the **On Screen Update** option.

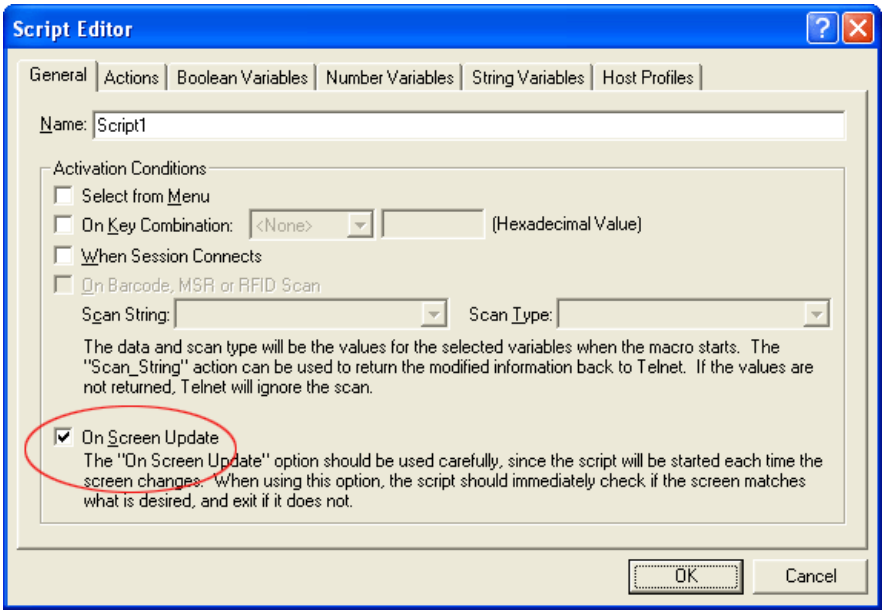


Figure 3-5. *Selecting the On Screen Update Method*

3 Click **OK**.

Creating Script Code

Once you name a script and select an activation method, you can use the **Actions** tab in the Script Editor to build the script. Refer to *Chapter 5: Building Scripts Manually* on page 51 for a detailed example of creating script code manually.

Creating Variables

There are three types of values recognized by scripting:

- Booleans (TRUE or FALSE values only)
- Numbers (integers)
- Strings

Every argument for every action is one of these three value types. Every action that returns a value returns one of these types. Variables provide a way

to save the result of an action for later use as an argument for another command.

You can create and edit variables located in the corresponding tabs while editing a script. You can also create new variables while editing an action.

When a script starts, all the variables will have known values: boolean variables are FALSE, number variables are 0, and string variables are empty. One possible exception to this is when a script activates another script. Refer to *Script Nesting* on page 34 for more information.

To create variables:

1 Determine which type of variable you want to create: boolean, number, or string.

2 From the Script Editor, select the tab that corresponds with the type of variable you want to create.

3 Click **Add**.

4 In the *Edit Variable* dialog box, enter the name of the new variable.

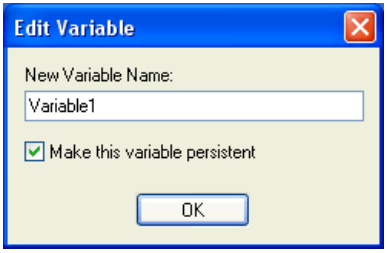


Figure 3-6. *Adding a New Variable*

5 If you would like the variable to be a persistent variable, enable the **Make this variable persistent** checkbox. For more information, refer to *Persistent Variables* on page 21.

6 Click **OK**.

The new variable appears in the corresponding tab.

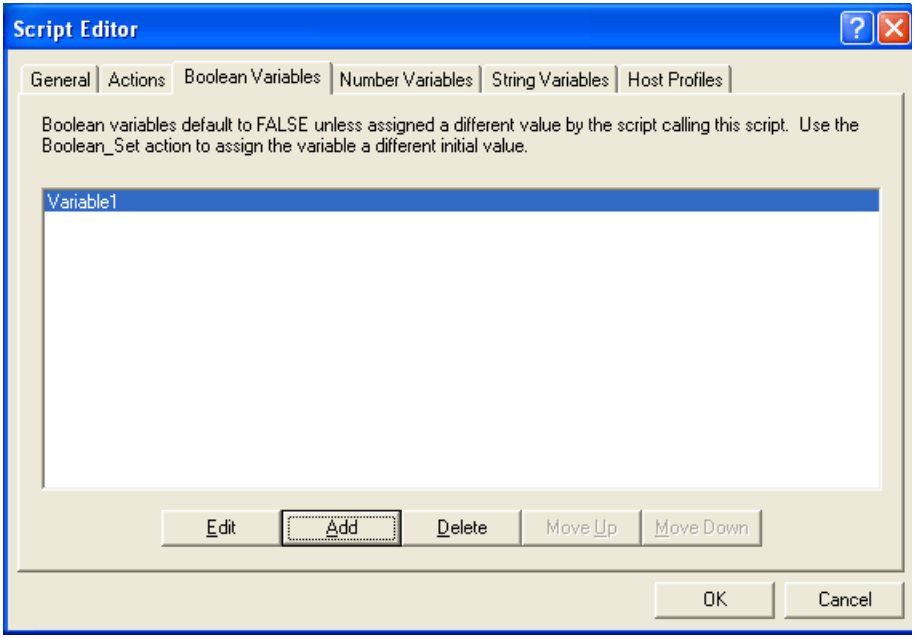


Figure 3-7. *New Variable*

Persistent Variables

When a variable is persistent, the value remains after the script exits. Persistent variables are not script or session specific; once a value is assigned, any script in any Telnet session can use a persistent variable to access that value.

Two scripts are considered to be referencing the same persistent variable if both scripts contain persistent variables of the same type and same name.

Because writing new values to persistent variables will slow your application, they should only be used when necessary. If you want to use a persistent variable that will change values frequently, write your script with a regular variable that only changes the value of the persistent variable before the script pauses or exits.

Selecting Host Profiles

For each script, you can specify which host profiles will be supported by that script. You may select host profiles from the **Host Profiles** tab.

If the script is generated by script capturing, limit the script to the host profile that was in use when the script was captured. The default - no host profile - allows the script to be run when any host profile is used.

To select host profiles:

1 From the Script Editor, select the **Host Profiles** tab.

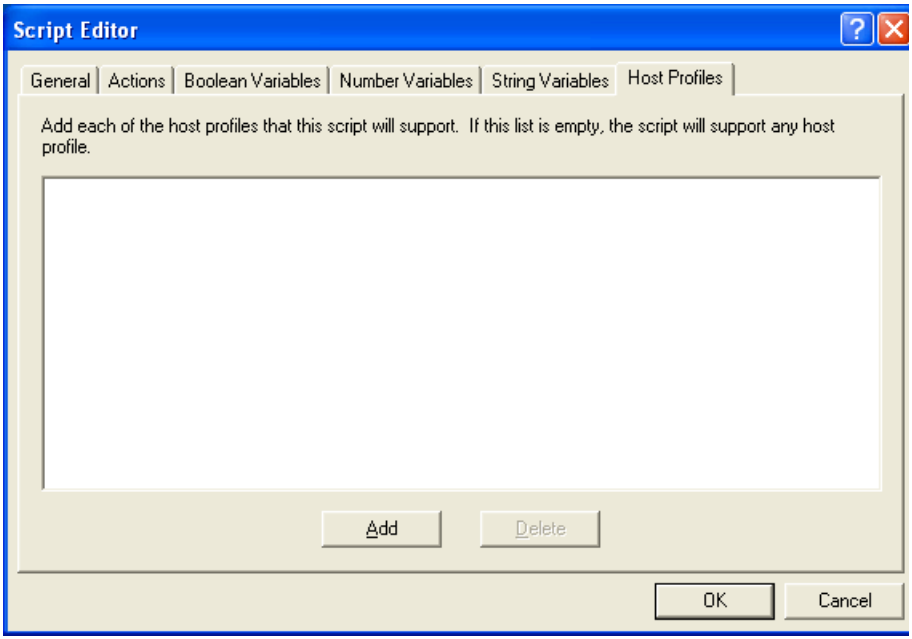


Figure 3-8. *Host Profiles Tab*

2 Click **Add**.

The *Select Host* dialog box opens.

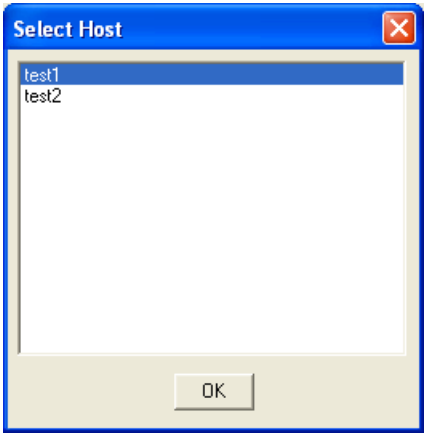


Figure 3-9. *Selecting Host Profiles*

3 Select which host you want to use from the list of Avalanche hosts.

NOTE If you have not created any host profiles, this dialog box will be empty.

4 Click **OK**.

The host appears in the **Host Profiles** tab.

Performing Script Capturing

Script capturing is an easy way to generate a script that automates actions or processes. When script capturing is activated, it captures key presses and mouse/pen cursor movements so those actions can be replayed when the script is activated.

To perform a script capture:

1 Position your mouse or cursor on the emulation screen you want to be at when the automated process starts.

2 From the **Term** or **Options** menu, select **Scripting > Start Capture**.

The *Script Capturing Initialization* dialog box appears, asking if you want to verify the current screen text.

3 Select **Yes** to verify the current screen text.

Select **No** if you do not want to verify the current screen text.

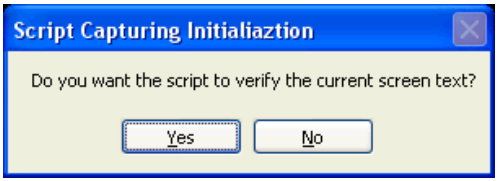


Figure 3-10. *Script Capturing Initialization Dialog Box*

Selecting **Yes** makes the captured script start with an `If_not` command that tells the script to exit if the correct screen is not currently shown. Unless you know that your script will only run from the correct screen (for example, a script that is run only when a session first starts, or a script called by another script), you should select **Yes**.

NOTE If you select **No**, click **Verify Screen Contents** and **Save Cursor Position** when you start your script capture. This causes your script to wait for Telnet to finish updating the screen before processing script actions.

---

4 Perform any actions you want to include in the script.

5 Each time the screen changes, click **Verify Screen Contents** button.

NOTE Some devices may only display buttons labeled **Screen**, **Cursor** and **Stop**. The **Screen** button refers to the **Verify Screen Contents** button. The **Cursor** button refers to the **Save Cursor Position** button. The **Stop** button refers to the **Stop Capturing** button.

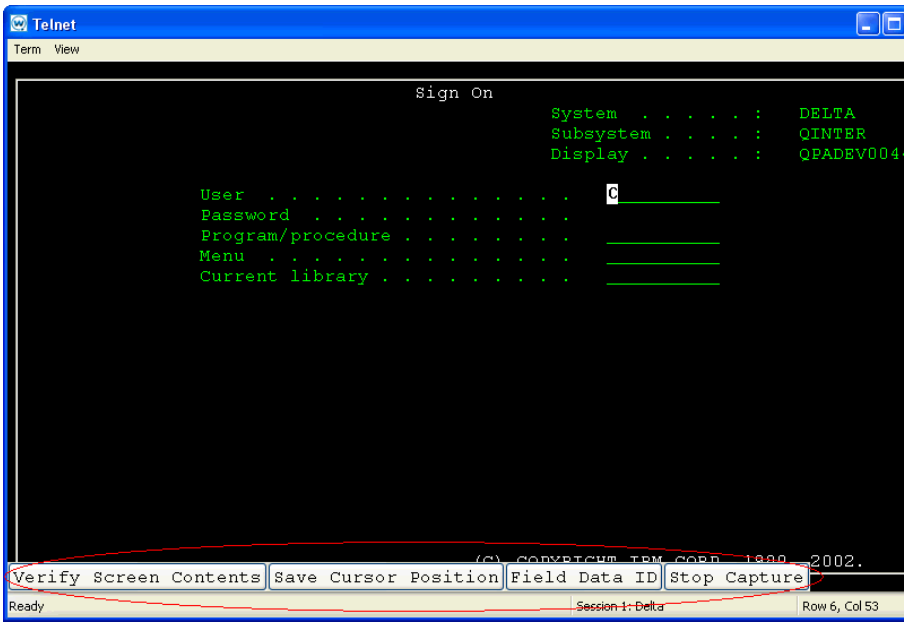


Figure 3-11. *Verify Screen Contents and Save Cursor Position Buttons*

NOTE Clicking the **Verify Screen Contents** button causes the generated script to pause and wait for the screen update. The pauses are necessary because scripts can run much faster than the interaction with the Telnet host.

---

6 When you finish capturing the behaviors you want in the script, click **Stop Capture**.

Once you capture a script, the Script Editor opens, allowing you to name the script and select an activation method. You can use the **Actions** tab to add actions for any error condition that the user may encounter.

Creating Scripts From Text

You can use the Script Editor to generate scripts from standard text. The text can be entered in the Script Editor's built-in text editor, or imported from a text-editing program such as Notepad. This section provides information about creating scripts from text, including the following:

- Importing a Text File
- Building Scripts with the Text Editor

Importing a Text File

Create a script in any standard text editor and then import the script into the Text Editor.When creating the script text, you must use the same actions and commands provided in the Script Editor.

NOTE For more information, refer to the *Wavelink Telnet Client Scripting Actions Library.*

To import a text file:

1 Launch the Script Editor.

2 Click **Import Text**.

The *Select the Script Text File* dialog box appears.

3 Navigate to and select the text file.

4 Click **Open**.

The file is imported into the Script Editor.

Building Scripts with the Text Editor

You can also create a text-based script using the Text Editor. The Text Editor will validate the script syntax when you build the script.

To build a script:

1 Launch the Script Editor.

2 Click **Add As Text**.

The Text Editor opens.

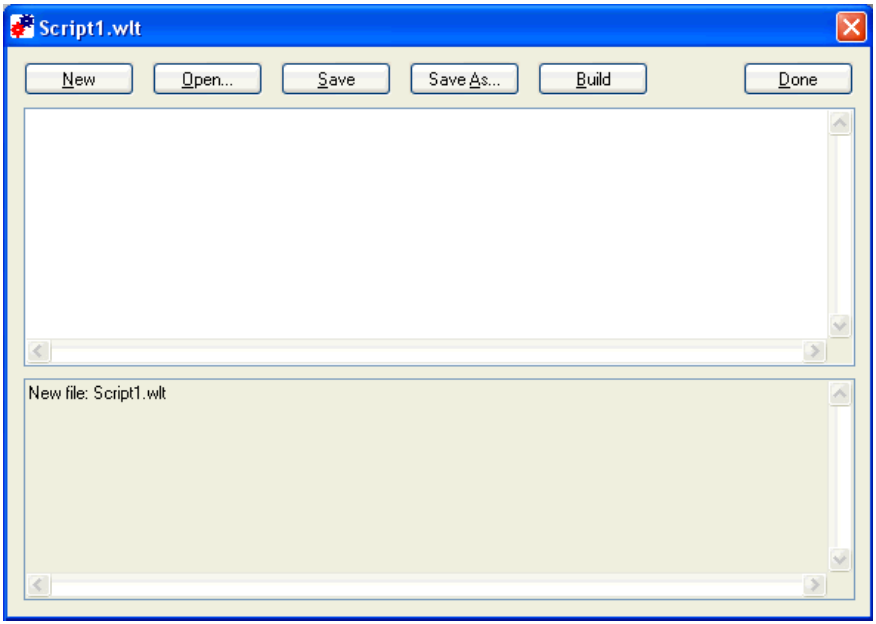


Figure 3-12. *Text Editor*

3 Enter the text into the Text Editor. You must include a script name, along with script variables and activation methods. If desired, you may also assign a Host Profile to be supported by the script.

NOTE For more information, refer to *Naming Scripts* on page 11, *Selecting Activation Methods* on page 11, *Creating Variables* on page 19, and *Selecting Host Profiles* on page 21.

---

4 After entering your text, click **Build** to verify the script text. The Text Editor displays build information in the bottom section of the window. If the build information includes an error message, single-click the message to display the *Script Editor Error Help* dialog box.

The *Script Editor Error Help* dialog box displays the error number, a description of the error, and explains how to fix the error.

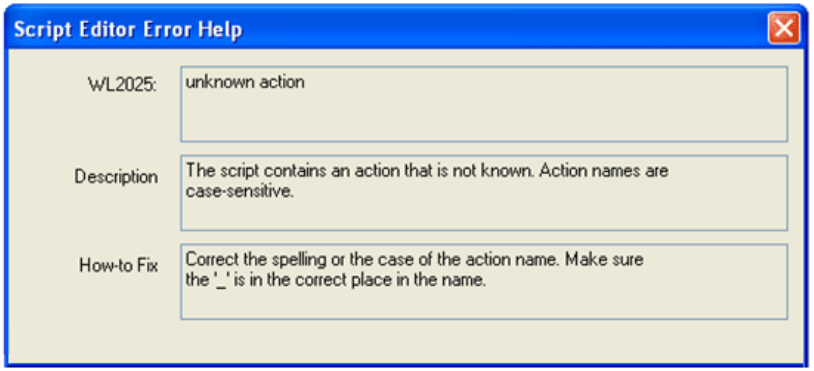


Figure 3-13. *Script Editor Error Help*

For a full description of each error message, refer to *Appendix B: Script Build Errors* on page 71.

5 When you have finished building the script, click **Save** or **Save As** to save a copy of the script text.

6 Click **Done** to add the script to the Script Editor, or click **New** to begin building a new script.

Editing Scripts

You can edit scripts that are created manually, scripts that are generated using script capturing, and scripts created from text.

To edit scripts:

1 Launch the Script Editor.

2 Select the script you want to edit from the script list.

3 Click **Edit**.

4 Make the desired changes in the Script Editor configuration dialog box.

5 Click **OK** to save your changes.

Once you have completed editing the script you have two options:

- Export the script to a specified location. Refer to *Saving and Exporting Scripts* on page 31 for more information.
- Execute the script by launching the Telnet Client and performing the activation method you assigned to the script. Refer to *Chapter 4: Executing Scripts* on page 47 for more information.

To edit scripts with the Text Editor:

1. Launch the Script Editor.
2. Select the script you want to edit from the script list and click **Edit As Text**.

   The Text Editor appears.
3. Edit the script as desired, then click **Build** to verify the script syntax.
4. When you have finished editing the script, click **Save** or **Save As** to save a copy of the script text.
5. Click **Done** to return to the Script Editor, or click **Open** to edit another script.

Importing Scripts

You can use the import button in the Script Editor to import previously created scripts.

NOTE You can only import scripts that have been created using the Script Editor.

To import a script:

1. From the Script Editor, click **Import**.

   The *Select the Script File* dialog box opens.

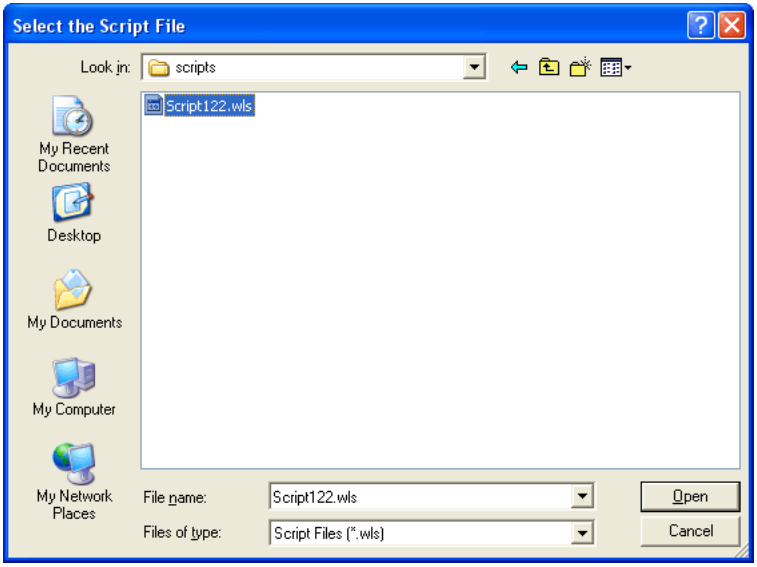


Figure 3-14. *Importing a Script File*

2 Navigate to and select the script file.

3 Click **Open**.

The name of the file is imported into the Script Editor.

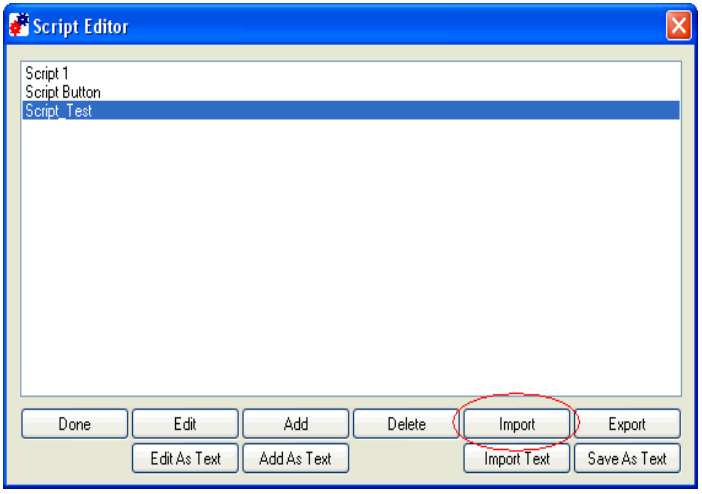


Figure 3-15. *Imported Script File*

Once you have imported the file, you can edit the script. Refer to *Editing Scripts* on page 28 for more information.

Saving and Exporting Scripts

After you finish building a script, your script is automatically saved in the Script Editor. You can also export a script and save it in a specific location on the network.

Scripts are saved as `.wls` files. Scripts can not be viewed outside the Script Editor and must be imported back into the Script Editor to view or edit.

To export a script:

1 From the script list, select the script you want to export.

2 Click **Export**.

-Or-

To save the script as text, select **Save As Text**.

The *Create the Script File* dialog box opens.

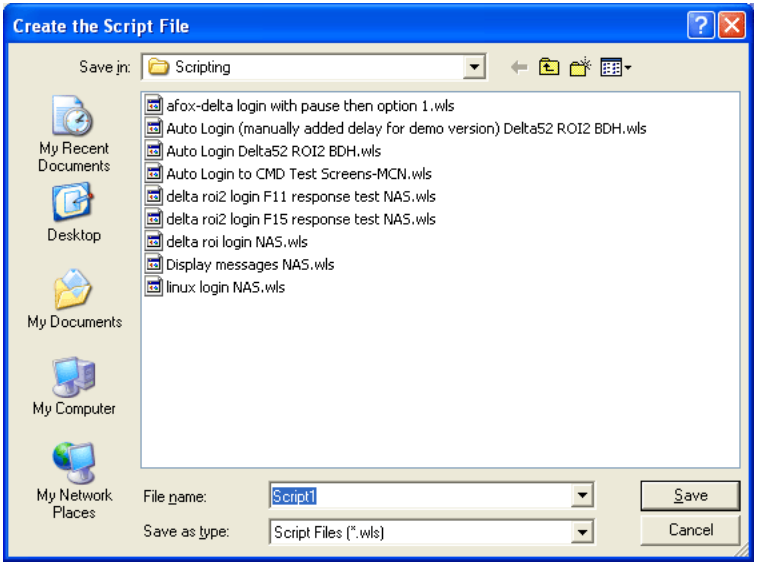


Figure 3-16. *Exporting a Script*

3 Navigate to the location where you want to export your script.

4 Click **Save**.

To view an exported script, you need to import the script into the Script Editor. Refer to *Importing Scripts* on page 29 and *Importing a Text File* on page 26 for more information.

Deploying Scripts

Scripts are deployed to the Telnet Client the next time the client syncs with the Avalanche Console.

Syncing Scripts

The section provides information about syncing scripts that are edited or created on the mobile device to the Script Editor in the Avalanche Console.

When you create or edit a script on a mobile device, you need to sync the script to the Script Editor in Avalanche Console. This allows you to edit and modify the scripts from the Avalanche Console.

To sync scripts:

1 Launch the Avalanche Console.

2 From the list of mobile devices, right-click the device to which you want to sync.

A menu list appears.

3 Select **Session Monitor**.

As you connect to Session Monitor, an *Authorizing* dialog box appears and then the *Sync Script* dialog box appears.

4 Click **Yes** to sync your scripts from the mobile device to Avalanche Console.

The Script Editor opens and display all scripts.

If you edited a script on the mobile device that is also saved in the Script Editor in the Avalanche Console, both scripts display in the Script Editor

after syncing. The original script retains the original name. The script that was edited on the mobile device appears as `[Original Name] Telnet`.

Creating Log Files

Use the `Logging_On` and `Logging_Off` commands in your code to generate log files. Each action executed, with the values of its arguments and the results of the action, is written to the log file. When you configure the `Logging_On` action, you can set the File Path name to where the log file is stored.

When a script calls another script, the logging for the calling script is suspended. Logging resumes when the called script exits and the suspended script resumes. It is possible to have a script called by another script (or a script calling itself recursively) use the same logging file.

Entering the Logging_On Action

Include a `Logging_On` action in your script code to generate a log file. Place the `Logging_On` action at the point you want to begin logging.

To enter the Logging_On action:

1. From the **Actions** tab, click **Insert**.
2. From the **Action** drop-down menu, select **Logging_On**.
3. Click the **File Path** tab.
4. In the **Constant String** text box, enter the location where you want to store the log file.
5. Click the **Overwrite Previous** tab.
6. Set the Override Previous option.

   When you set the **Override Previous** option to FALSE, the latest log file will not replace the existing file. Instead, a separate log file is created for each log.

   When you set the **Override Previous** option to TRUE, the most recent log file will replace the existing log file.

NOTE Because logging slows the performance of Telnet, and takes up space on your devices, you may not want to include it in end-user scripts. Set the **Override Previous** value to TRUE to keep the log files from getting too large.

7 Click **OK**.

The code is added to the **Actions** tab.

Entering the Logging_Off Action

If the script exits, logging automatically terminates, so you usually will not need the `Logging_Off` action. However, if you only want to log a portion of the script, you can enter a `Logging_Off` action to stop the logging.

To enter a Logging_Off action:

1 From the **Actions** tab, click the **Insert** button.

2 From the **Action** drop-down menu, select **Logging_Off**.

3 Click **OK**.

The code is added to the **Actions** tab.

Script Nesting

You can program a script to call other scripts or to call itself. This makes it easier to take a block of functionality and use it multiple times or to solve problems that can be described recursively.

A factorial is the product of all positive integers from 1 to the given number. For example, the factorial of 5 (usually written as 5!) = 1 x 2 x 3 x 4 x 5 = 120. Here is an example of a script that uses recursion (a script calling itself) to calculate factorials:

```
  If (Number_Equal(ArgumentValue,1))
Comment: The factorial of 1 is 1
  Return
  End_If
  If (Number_Not_Equal(ArgumentValue,0))
Comment: The factorial of X is X multiplied by the
factorial of X - 1
  Temp=ArgumentValue
```

```
   ArgumentValue=Number_Minus(Temp,1)
   Call: Factorial
   ArgumentValue <-> ArgumentValue
   ArgumentValue = Number_Multiply(Temp,ArgumentValue)
   Return
   End_If
   ArgumentValue = Ask_Number("Enter a number:",
"Factorial Calculator",1,12,0)
   Call: Factorial
   ArgumentValue <-> ArgumentValue
Ask_OK(String_Combine("The factorial is",
Number_To_String_Decimal(ArgumentValue)), "Result")
   Return
```

This script uses two integer variables, `ArgumentValue` and `Temp`.

When a script calls another script, the calling script can assign values to the called script variables. The factorial example script knows it is being called recursively because the `ArgumentValue` variable is not 0. If `ArgumentValue` is 0, the script will ask for the number to calculate the factorial with. Each time the script is called, the `ArgumentValue` variable of the calling script is assigned the final value in the called script's `ArgumentValue` variable. This keeps the results of the called script's actions from being lost. (If the value were not returned, then there would be a "<--" instead of a "<->" in the Call action's argument list.) When you add the action for calling a script, you need to specify which variables in the called script will be assigned a value, and what that initial value will be.

NOTE If you wanted to be more efficient, you could create a `While` loop that performs the multiplications to calculate the factorial. You could also return the proper response for each factorial, since numbers higher than 12 exceed the maximum number value. However, there are some problems that are easiest to solve using recursion. The example above should give you an idea how you could go about using it.

Field Data ID Feature

The Field Data ID feature allows you to use scripting to configure Field Data Identifiers. Field Data Identifiers assign a unique identification, such as a letter, to each field on the screen. Any time a barcode beginning with that identifier is scanned, the information automatically populates in the corresponding field.

NOTE This feature is only available for IBM 5250 emulation.

This section provides the following information:

- Adding a Field Data ID or Symbology
- Editing a Field Data ID or Symbology

Adding a Field Data ID or Symbology

Use the *Field Data ID* dialog box to add a Data ID or Symbology to a field.

To add a Field Data ID or Symbology:

1. Access the IBM 5250 emulation screen.
2. From the **Term** menu, select **Scripting > Start Capture**.

   The *Script Capturing Initialization* dialog box appears.
3. Click **Yes**.

   The *Select Screen Text* dialog box appears.
4. Select the desired text string(s) from the list and click **OK**.
5. Place the cursor on the field where you want to add a Data ID.
6. From the bottom of the emulation screen, click the **Field Data ID** button.

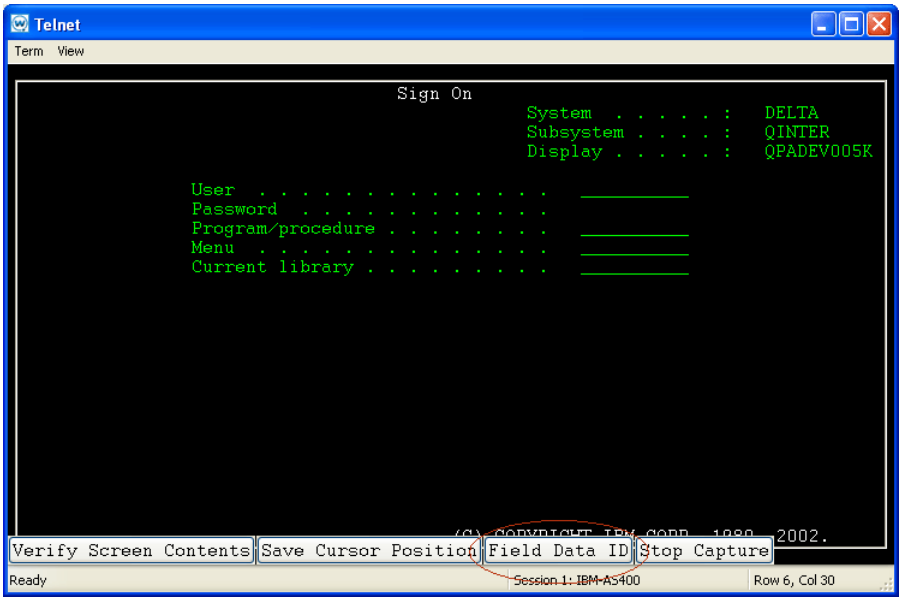


Figure 3-17. *Field Data ID Button*

The *Field Data ID* dialog box appears.

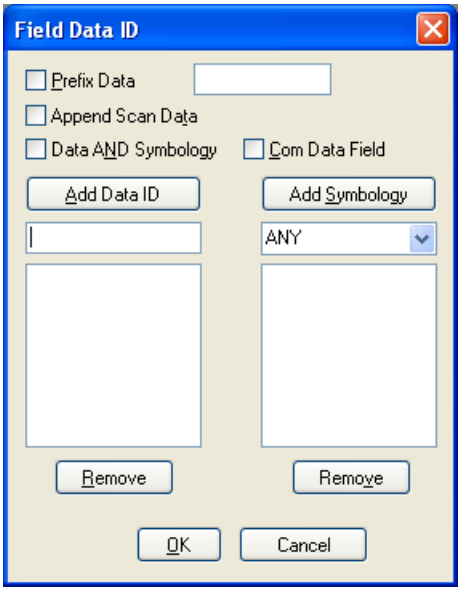


Figure 3-18. *Field Data ID Dialog Box*

7 In the text box below the **Add Data ID** button, enter the desired Data ID.

8 Click **Add Data ID**.

The new Data ID appears in the list below the text box.

9 If you want to remove a Data ID, select the desired ID from the list and click **Remove**.

10 From the drop-down menu below the **Add Symbology** button, select the desired Symbology.

11 Click **Add Symbology**.

The new Symbology appears in the list below the text box.

12 If you want to remove a Symbology, select the desired Symbology from the list and click **Remove**.

13 If you want to add a prefix to the data, enable the **Prefix Data** checkbox and enter the prefix in the available text box.

- If you want to append scan data in the field, enable the **Append Scan Data** checkbox.
- If you want to add a Data ID and a Symbology, enable the **Data AND Symbology** checkbox.
- If you want the field to be the Com Data Field for the screen, enable the **Com Data Field** checkbox.

14 Click **OK** to close the *Field Data ID* dialog box and save your changes.

Editing a Field Data ID or Symbology

If you want to make changes to a Data ID or Symbology, you must manually edit the script you created using the Field Data ID feature. For more information about editing scripts, refer to *Editing Scripts* on page 28.

Chapter 4: Executing Scripts

This section provides information about activating scripts using each of the available activation methods:

- Select From Menu
- On Key Combination
- When Session Connects
- On Barcode, MSR, or RFID Scan
- On Screen Update
- From Web Pages

For information on assigning an activation method to a script, refer to *Selecting Activation Methods* on page 19.

NOTE Screen captures may differ according to device type.

Select From Menu

If you assigned this activation method, you can activate the script from the **Term** menu of the Telnet Client.

To activate:

1 Launch the Telnet Client on the mobile device.

2 From the **Term** menu, select **Scripting > Execute Script**.

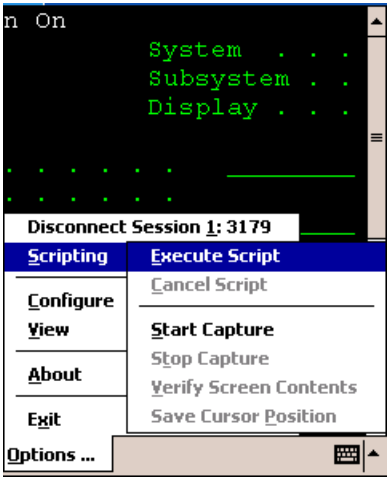


Figure 4-1. *Executing Scripts from the Menu*

3 If more than one script is available for the current host profile, select which script you want to use from the list.

NOTE This option is not available if a script is running for the current session or if the session is not connected.

On Key Combination

If you assigned this activation method, the script activates when you press the specified key combination (as long as it is currently possible for script to run).

To activate:

1 Launch the Telnet Client.

2 Enter the key combination you assigned to execute the script.

When Session Connects

If you assigned this activation method, the script activates when the host profile it supports is activated.

To execute when the session connects:

1 Launch the Telnet Client.

2 From the **Term** or **Options** menu, select **Connect**.

3 Select the host to which you want to connect.

4 Click **OK**.

The script runs upon connection.

On Barcode, MSR, or RFID Scan

If you assigned this activation method, the script activates with each barcode, MSR, or RFID scan.

On Screen Update

If you assigned this activation method, the script activates (if activation is allowed) every time the text on the emulation screen changes. This includes updates from the Telnet host or when the user presses a key and the key value appears on the screen.

From Web Pages

You can launch Wavelink scripts from web pages using the `wls` type, followed by the script name.

Launching Scripts from Web Pages Example 1

This example launches a script called `WebAuto` which launches when the web page first loads.

```
<title>TE70 Test1 - Launch Telnet Scripts</title>
<meta http-equiv="OnStartup" content="wls:WebAuto">
```

Launching Scripts from Web Pages Example 2

This example launches a script called `WebClick` when a user clicks the hyperlink "`here`" on the web page.

```
  <p>
  Click <a href="wls:WebClick">here</a> to launch the
&amp;quot;WebClick&amp;quot; script.
  </p>
```

Chapter 5: Building Scripts Manually

This chapter provides information about building a script manually. The purpose of this example is to demonstrate (step-by-step) how to create a script. Values, names and variables are used for example purposes only. The example values may differ according to device type and operating system.

The following script is the example used:

```
Comment:Verify that this is the desired screen
  If_Not(String_Equal(Get_Screen_Text_Length(1, 36, 7),
"Sign On", 0, FALSE))
  Return
  End If
  Set_Cursor_Positiong(6, 53)

  Message("Starting Script" 3)

  Keypress_String("User Name")
  Keypress_Key("Down Arrow")
  Keypress_String("Password")
  Keypress_Key("Enter")

Comment:Wait for the desired screen.

  While_Not(String_Equal(Get_Screen_Text_Length(11,1,
16), "9. FUNCTION KEYS",0,FALSE))
  Wait_For_Screen_Update
  End_While

  Keypress_String("9")

  Message("String Done",3)
  Return
```

NOTE Screen captures may differ according to device type.

The following sections contain the tasks required to create the example script:

- Launching the Script Editor
- Naming the Script and Selecting the Activation Method
- Building the Script Code.

Launching the Script Editor

Launch the Script Editor from the Avalanche Console or from the Telnet Client. For detailed instructions about launching the Script Editor, refer to *Chapter 2: Launching the Script Editor* on page 13.

Naming the Script and Selecting the Activation Method

In the **General** tab, name the script and select which method you want to use to activate the script. For detailed instructions, refer to *Naming Scripts* on page 19 and *Selecting Activation Methods* on page 19.

Building the Script Code

Once you name the script and select an activation method, you can begin entering the code. You build the code in the **Actions** tab. The example script performs the following actions upon completion and activation:

- Verifies that the script is starting on the right screen and sets the correct cursor position.
    - If the script is not on the correct screen, the script ends.
    - If the emulation screen is correct, the script enters a user name and password in the correct locations.
- Verifies that the emulation screen is correct.
- Selects a specific menu and exits.

The steps to enter this example code are divided into three main sections:

1. Verifying the Script Starts on the Correct Screen
2. Entering the User Name and Password
3. Verifying the Screen and Navigating Menus

Verifying the Script Starts on the Correct Screen

This portion of script verifies that the script is starting on the right screen and sets the correct cursor position. If the script is not on the right screen, the

script ends. Once the script verifies that it is starting on the correct screen, a message displays "`Starting Script`."

To build the script code:

1 From the **Actions** tab, click the **Insert** button.

The *Action Editor* dialog box opens.

2 From the **Action** drop-down menu, select **Comment**.

3 Click the **Comment** tab and enter `Verify that this is the desired screen` in the **Constant String** text box.

NOTE Comments can be any string you want. This is just an example.

---

4 Click **OK**.

The code is added to the **Actions** tab.

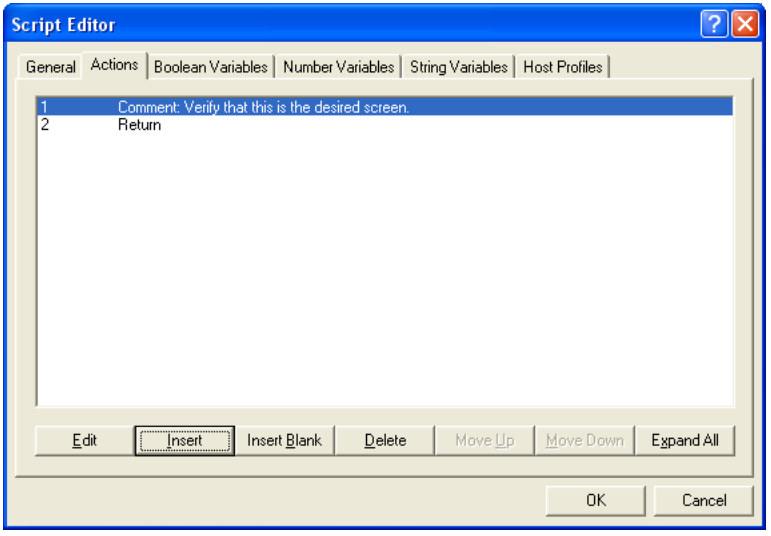


Figure 5-1. *Entering a Comment*

5 Click the **Insert** button.

6 From the **Action** drop-down menu, select **If_Not**.

7 Click the **Test** tab.

8 Enable the **Action** drop-down option and select **String_Equal** from the drop down menu.

9 Click the **Edit Action Value** button.

10 Click the **Test 1** tab.

11 Enable the **Action** drop-down option and select **Get_Screen_Test_Length** from the drop down menu.

12 Click the **Edit Action Value** button.

13 Click the **Row** tab and enter the number `1`.

14 Click the **Column** tab and enter the number `38`.

15 Click the **Maximum Length** tab and enter the number `7`.

16 Click **OK**.

17 Click the **Test 2** tab and enter `Sign On` in the **Constant String** text box.

18 Click the **Maximum Length** tab and enter the number `0` in the **Constant Number** text box.

19 Click the **Ignore Case** tab and enable the **False** option.

20 Click **OK** until you return to the **Actions** tab in the Script Editor.

The code appears in the **Actions** tab.

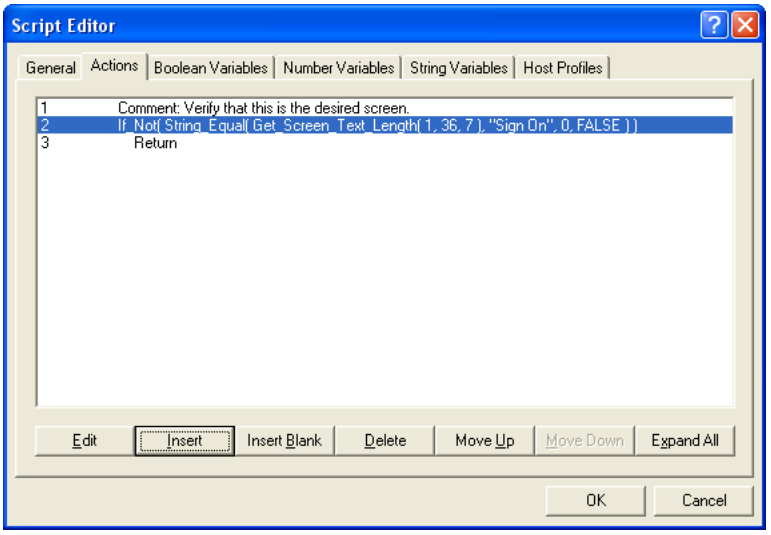


Figure 5-2. *Entering an IF_Not Action*

21 Click the **Insert** button

22 From the **Action** drop-down menu, select **Return**.

23 Click **OK**.

The code appears in the **Actions** tab.

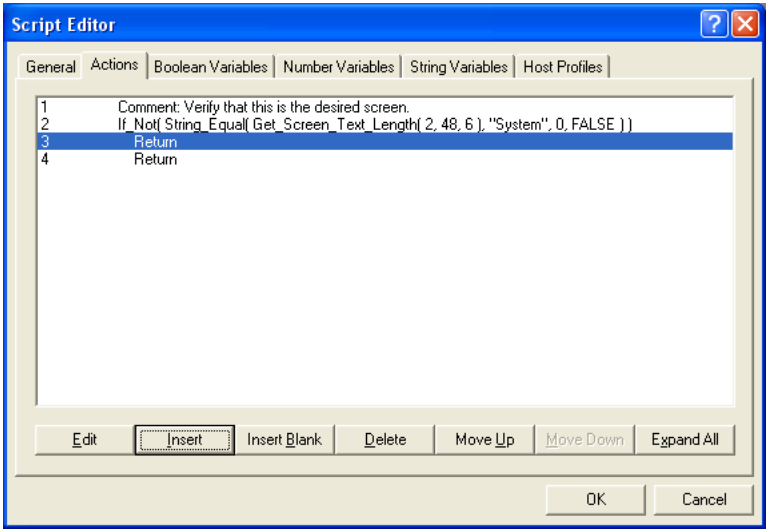


Figure 5-3. *Entering a Return*

24 Click the **Insert** button.

25 From the **Action** drop-down menu, select **End_If**.

26 Click **OK**.

The code appears in the **Actions** tab.

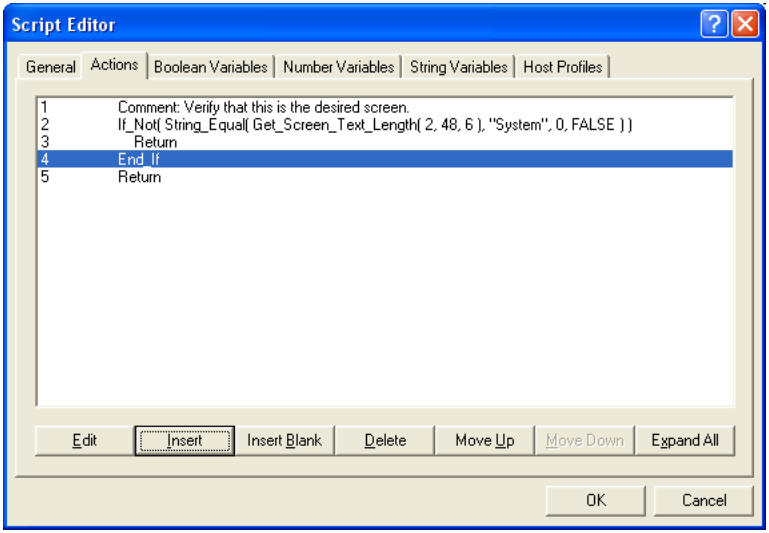


Figure 5-4. *Entering the End_If Action*

27 Click the **Insert** button.

28 From the **Action** drop-down menu, select **Set_Cursor_Position.**

29 Click the **Row** tab and enter `6` in the **Constant Number** text box.

30 Click the **Column** tab and enter `53` in the **Constant Number** text box.

31 Click **OK**.

The code appears in the **Actions** tab.

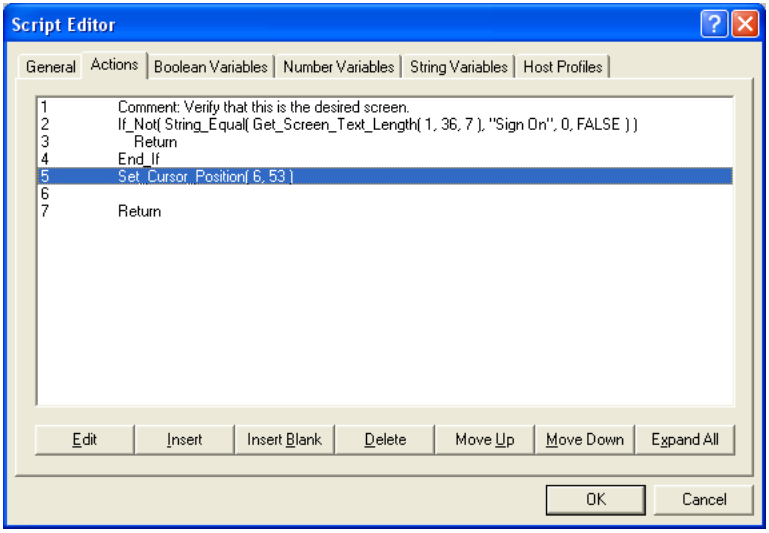


Figure 5-5. *Entering the Set_Cursor_Position Action*

32 From the **Action** tab, click the **Insert Blank** button to insert a blank line in the code.

33 Click the **Insert** button.

34 From the **Action** drop-down menu, select **Message**.

35 Click the **Message** tab and enter `Starting Script` in the **Constant Number** text box.

This code displays a "`Starting Script`" message on the emulation screen.

36 Click the **Timeout (Seconds)** tab and enter the number `3` in the **Constant Number** text box.

37 Click **OK**.

The code appears in the **Actions** tab.

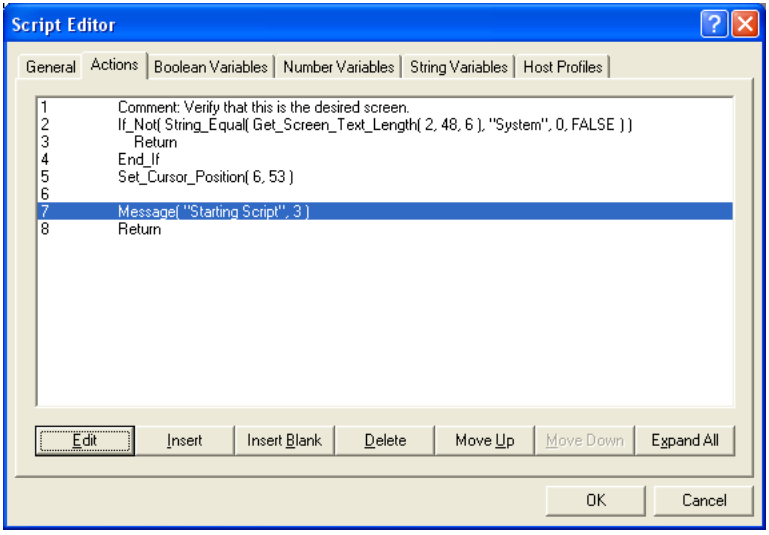


Figure 5-6. *Entering the Message Code*

Entering the User Name and Password

This portion of the script enters the login information.

To build the script code:

1. Click the **Insert Blank** button to insert a blank line in the code.
2. Click the **Insert** button.
3. From the **Action** drop-down menu, select **Keypress_String**.
4. Click the **Characters** tab and enter `User Name` in the **Constant String** text box.
5. Click **OK**.

   The code appears in the **Actions** tab.

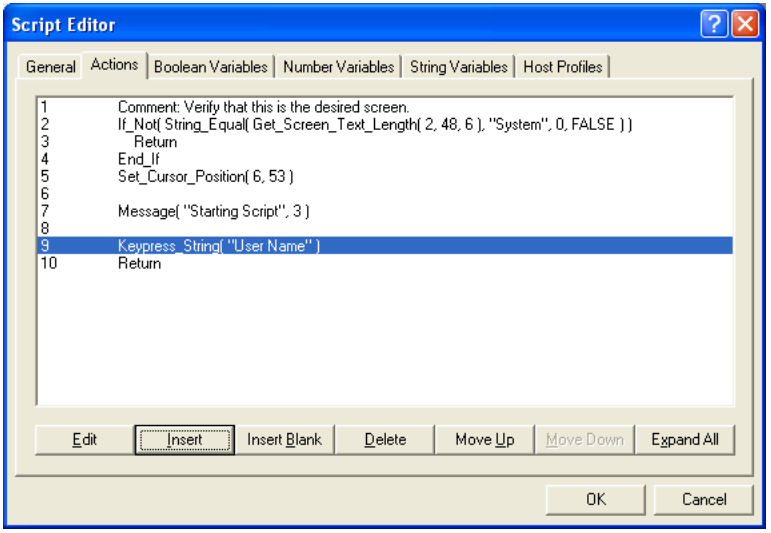


Figure 5-7. *Entering the User Name Code*

6 Click the **Insert** button.

7 From the **Action** drop-down menu, select **Keypress_Key**.

8 Click the **Key** tab.

9 From the **Emulation** drop-down menu, select the emulation type.

10 From the **Key** drop-down menu, select **Down Arrow**.

11 Click **OK**.

The code appears in the **Actions** tab.

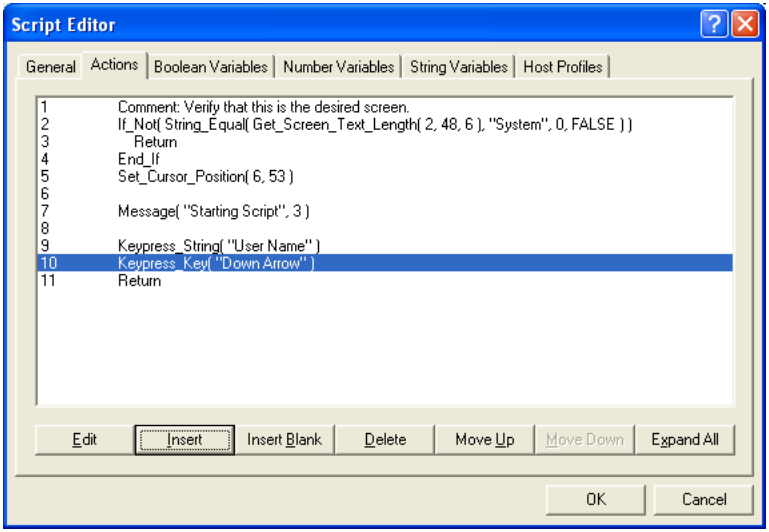


Figure 5-8. *Entering the Down Arrow Keypress*

12 Click the **Insert** button.

13 From the **Action** drop-down menu, select **Keypress String**.

14 Click the **Character** tab and enter `Password` in the **Constant String** text box.

15 Click **OK**.

The code appears in the **Actions** tab.

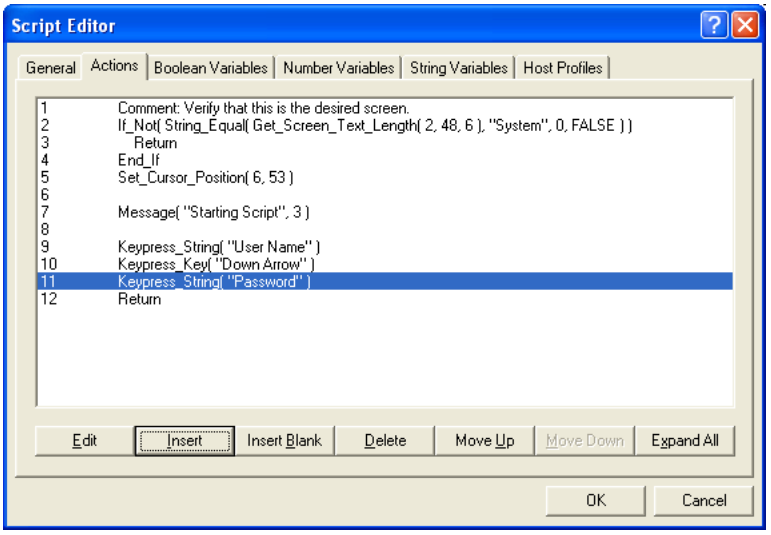


Figure 5-9. *Entering the Password Action*

16 Click the **Insert** button.

17 From the **Action** drop-down menu, select **Keypress_Key**.

18 Click the **Key** tab.

19 From the **Emulation** drop-down menu, select your emulation type.

20 From the **Key** drop-down menu, select **Enter**.

21 Click **OK**.

The code appears in the **Actions** tab.

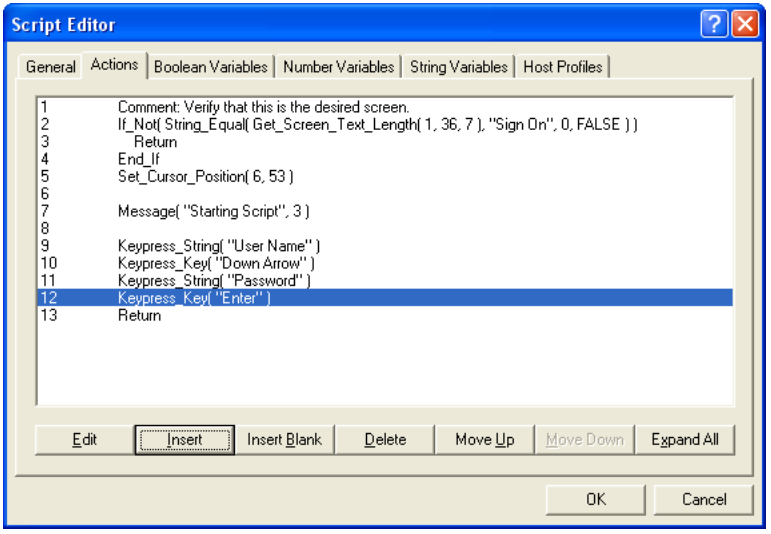


Figure 5-10. *Entering an Enter Action*

Verifying the Screen and Navigating Menus

This portion of the code verifies that you are on the correct screen after login. If you are not on the correct screen, the script will wait until the correct screen appears. Once the you are on the correct screen, the script navigates to a specific menu. The script displays the message "`Script Done`" and the script exits.

To build the script code:

1. Click the **Insert Blank** button.
2. Click the **Insert** button.
3. From the **Action** drop-down menu, select **Comment**.
4. Click the **Comment** tab and enter `Wait for desired screen` in the **Constant String** text box.
5. Click **OK**.

   The code appears in the **Actions** tab.

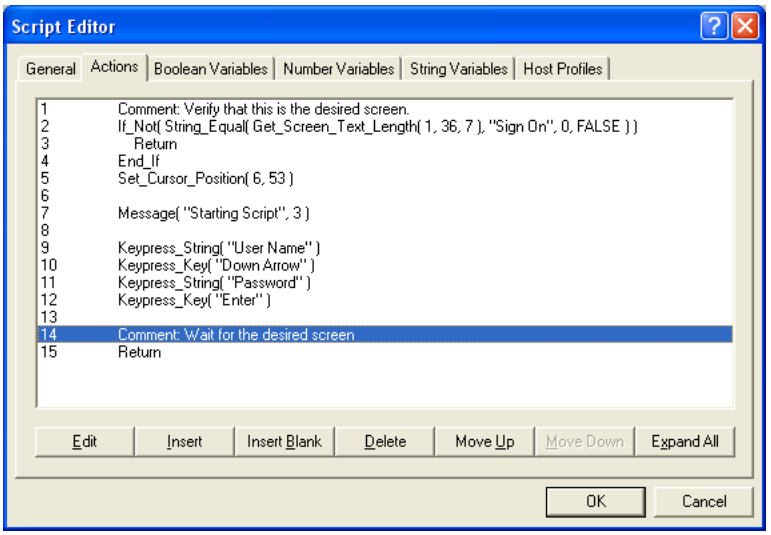


Figure 5-11. *Entering the Delay Action*

6 Click **Insert**.

7 From the **Action** drop-down menu, select **While_Not**.

8 Click the **Test** tab.

9 From the **Action** drop-down menu, select **String_Equal**.

10 Click the **Edit Action Value** button.

11 Click the **Test 1** tab.

12 From the **Action** drop-down menu, select **Get_String _Text_Length**.

13 Click the **Test 2** tab and enter `9. FUNCTION KEYS` in the **Constant String** text box.

14 Click the **Maximum Length** tab and enter the number `0` in the **Constant Number** text box.

15 Click the **Ignore Case** tab and enable the **FALSE** option.

16 Click the **Test 1** tab.

17 Click the **Edit Action Value** button.

18 Click the **Row** tab and enter the number `11` in the **Constant Number** text box.

19 Click the **Column** tab and enter the number `1` in the **Constant Number** text box.

20 Click the **Maximum Length** tab and enter the number `16` in the **Constant Number** text box.

21 Click **OK** until you return to the **Action** tab.

The code is added to the **Action** tab.

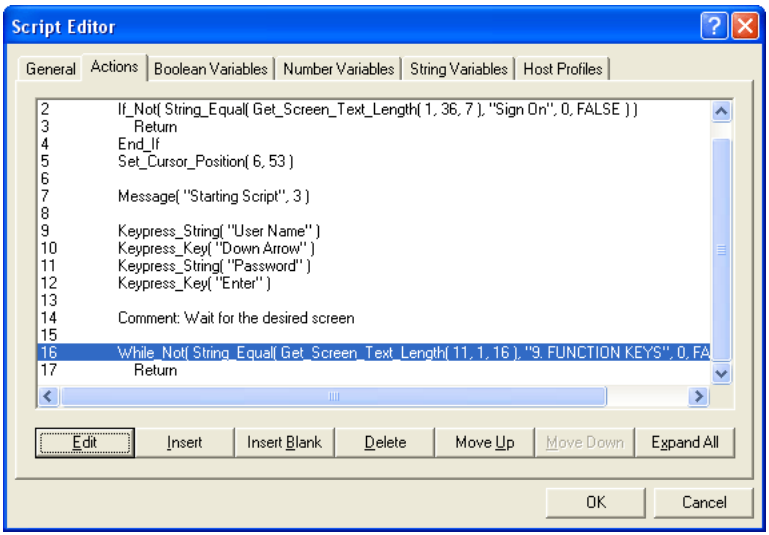


Figure 5-12. *Entering a While_Not Statement*

22 Click the **Insert** button.

23 From the **Action** drop-down menu, select **Wait_For_Screen_Update**.

24 Click **OK**.

The code appears in the **Actions** tab.

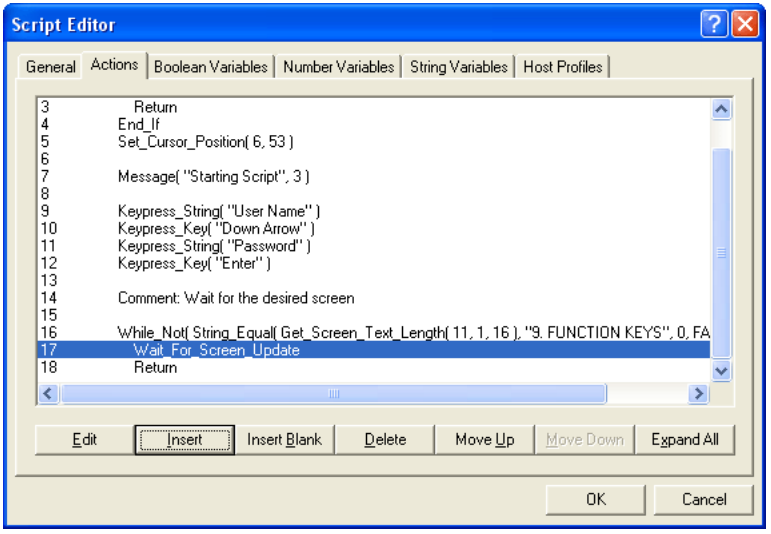


Figure 5-13. *Entering a Wait_For_Screen_Update Action*

25 Click the **Insert** button.

26 From the **Action** drop-down menu, select **End_While**.

27 Click **OK**.

The code appears in the **Actions** tab.

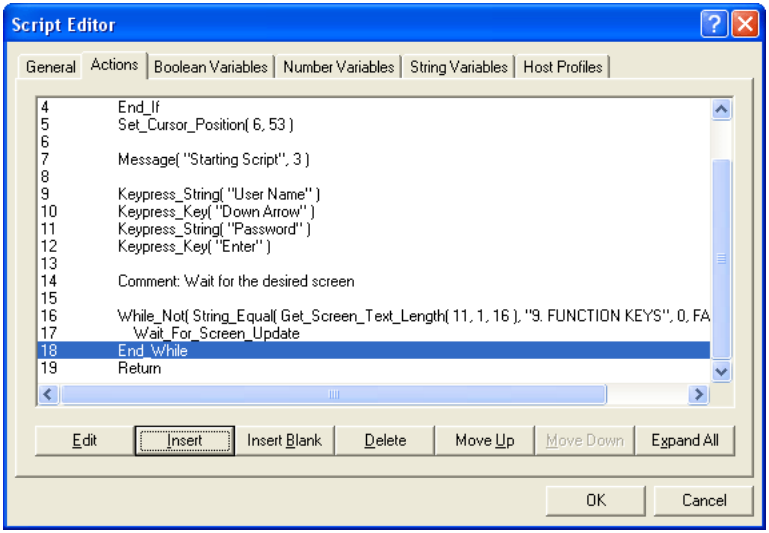


Figure 5-14. *Entering an End_While Action*

28 Click the **Insert Blank** button to insert a blank line in the code.

29 Click the **Insert** button.

30 From the **Action** drop-down menu, select **Keypress_String**.

31 Click the **Characters** tab and enter the number 9 in the **Constant String** text box.

32 Click **OK**.

The code is added to the **Actions** tab.

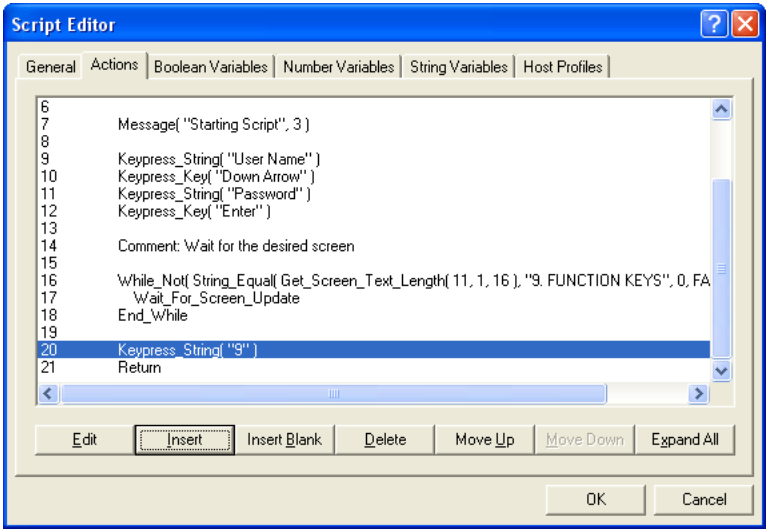


Figure 5-15. *Entering a Keypress Action*

33 Click the **Insert Blank** to insert a blank line in the code.

34 Click the **Insert** button.

35 From the **Action** drop-down menu, select **Message**.

36 Click the **Message** tab and enter `Script Done` in the **Constant String** text box.

37 Click the **Time(Milliseconds)** tab and enter the number `3` in the **Constant Number** text box.

38 Click **OK**.

The code is added to the **Actions** tab.

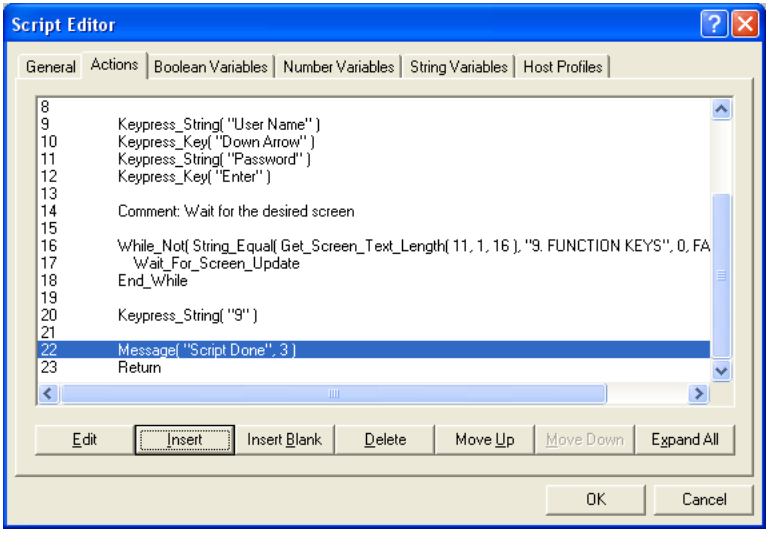


Figure 5-16. *Entering a Message Action*

The code is complete.

39 Click **OK** to save the code in the Script Editor script list.

Once you complete the script you have two options:

- Export the script to a specified location. Refer to *Saving and Exporting Scripts* on page 37 for more information.
- Execute the script by launching the Telnet Client and performing the activation method you assigned to this script. Refer to *Chapter 4: Executing Scripts* on page 47 for more information.

Appendix A: Examples

This appendix provides example scripts. Use these script examples to customize your own scripts according to preference.

Example 1: Beep

This is an example of a script that tells the device to beep if the word `ALARM` appears on the top five rows of the screen.

Example Code

```
If_Not(Search_Screen("ALARM",1,5,FALSE))
Return
End_If
Beep(1000,200,5)
Delay(200)
Beep(1500,500,9)
Return
```

Notes

This example should be set to activate each time the screen changes. Make sure that the `ALARM` text disappears quickly after being shown. Otherwise, the alarm will go off each time the screen updates (because the user pressed a key, each character from a bar code scanned was shown on the screen, etc.).

Here is an alternate implementation that waits for the `ALARM` text to disappear. The limitation with this version is that no other scripts will be able to run until the `ALARM` text is removed from the screen.

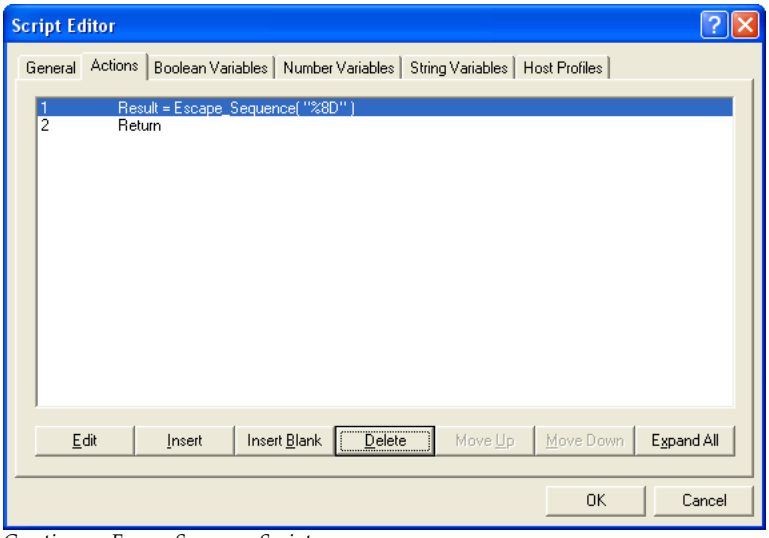
```
If_Not(Search_Screen("ALARM",1,5,FALSE))
Return
End_If
Beep(1000,200,5)
Delay(200)
Beep(1500,500,9)
While(Search_Screen("ALARM",1,5,FALSE))
Wait_For_Screen_Update
End_While
Return
```

Example 2: Escape Sequence

This is an example of using an escape sequence to turn off the Codabar symbology.

Example Code

```
Result=Escape_Sequence("%8D")
Return
```



Creating an Escape Sequence Script

Notes

You need to create a string variable named `Result` for this example, since the `Escape_Sequence` command returns a string value.

Refer to *Creating Variables* on page 27 for more information on creating variables.

Example 3: Request Information

This example requests the mobile device user to input some information and displays the result.

Example Code

```
  Result=Ask_String("Enter a string:","Length
Calculator",0,200,"")
  Ask_OK(String_Combine("The string"",String_Combine(
Result, String_Combine(""is",String_Combine(
Number_To_String_Decimal(String_Length(Result)),"characte
rs long." )))),"String Length")
  Return
```

Notes

This example also requires a string variable named `Result`. The `ASK_OK` instruction uses actions inside of actions to get several layers deep. You could also use variables to break that instruction into several short instructions.

Example 4: Display Screen Button

This example displays a log out button that appears in the in the bottom-left corner of the screen (row two, column five) when the text "`logged out`" appears on the screen. Pressing the button allows you to exit Telnet.

Example Code

```
  If_Not(String_Equal( Get_Screen_Text_Columns(2,5,10),
"logged out", 0, FALSE ) )
  Return
  End_If
  Button_Create_View( "Exit", 1000,1,0,ButtonPressed)
  While_Not( ButtonPressed)
  If_Not(String_Equal(Get_Screen_Text_Columns(2,5,10),
"logged out",0,FALSE))
  Return
  End_If
  Wait_For_Screen_Update
  End_While
  Exit_Application(0)
```

Notes

This example uses a boolean variable "`ButtonPressed`" to know if the screen button is pressed. The button is destroyed when the script exits so you do not need to delete the script.

The `While_Not` loop uses the `Wait_For_Screen_Update` action to detect if the `logged out` text is no longer there so that the scripts do not continually loop.

Sample Voice-Enabled Emulation Scripts

This section contains example scripts that perform various Voice-Enabled Emulation functions. For information on using the sample voice scripts, refer to *Wavelink Voice-Enabled Emulation User Guide.*

Play_Screen Sample Script

The following example script converts the current Telnet Client screen into speech that the user can hear.

```
nNumRows=Get_Screen_Rows
nCurrentRow=1
While(Number_Less_Than_Or_Equal(nCurrentRow,nNumRows))
  Speech_From_Text(Get_Screen_Text
  (nCurrentRow,1),FALSE)
  nCurrentRow=Number_Plus(nCurrentRow,1)
End_While
Return
```

Get_Number_Test Sample Script

The following example script converts a spoken number into text that displays on the mobile device. This script must be used in conjunction with the `Get_Number` sample script.

```
Speech_From_Text("Say a number",FALSE)
Call:Get_Number
   nResult <--> nResult
Ask_OK(Number_To_String_Decimal(nResult),
"Number Returned")
Return
```

Get_Number Sample Script

The following example script is called by the `Get_Number_Test` script. It retrieves the appropriate number for the `Get_Number_Test` script to display.

```
Comment:This script is designed to be called by other
scripts.
Comment:The result of the Speech-to-Text will be in the
nResult variable.
```

```
Comment:The number.bnf file must be available as a
grammar file.

Speech_To_Text(sResult,”number”)
nResult=0
While_Not(String_Empty(sResult))
 nNextSpace=String_Find_First(sResult,””,FALSE)
 nResult=Number_Plus(nResult,String_To_Number_Decimal
 (sResult))
 If_Number_Less_Than(nNextSpace,0))
    Break
 End_If
 nNextSpace=Number_Plus(nNextSpace,1)
 sResult=String_Right(sResult,Number_Minus
 (String_Length(sResult),nNextSpace))
End_While
Return
```

Speech_Button_Demo Sample Script

The following example script creates the following buttons on the screen: `Digits, State, Play Screen, Done`. When selected, the buttons allow the user to verbally input data.

```
While_Not(bExit)

 If_Not(bButtonsVisible)
    Button_Create_View(“Digits”,999,1,6,bGetDigits)
    Button_Create_View(“State”,999,16,5,bGetState)
    Button_Create_View(“PlayScreen”,1000,1,11,
    bPlayScreen)
    Button_Create_View(“Done”,1000,13,4,bExit)
 End_If

 Wait_For_Screen_Update

 If(bPlayScreen)
    bPlayScreen=FALSE
    Button_Remove_All
    bButtonsVisible=FALSE
    Delay(1)

    nNumRows=Get_Screen_Rows
    nCurrentRow=1
    While(Number_Less_Than_Or_Equal
    (nCurrentRow,nNumRows))
     Speech_From_Text(Get_Screen_Text
     (nCurrentRow,1),FALSE)
     nCurrentRow=Number_Plus(nCurrentRow,1)
    End_While
 End_If
```

```
 If(bGetDigits)
    bGetDigits=FALSE
    Button_Remove_All
    bButtonsVisible=FALSE

    Message("Say 1 or more digits...",0)
    szResult=""
    Speech_To_Text(szResult,"connected_digits")
    Message_Clear
    szResult=String_Strip_Characters(szResult,"",FALSE)
    Keypress_String(szResult)
 End_If

 If(bGetState)
    bGetState=FALSE
    Button_Remove_All
    bButtonsVisible=FALSE

    Message("Say a USA state...",0)
    szResult=""
    Speech_To_Text(szResult,"usa_states")
    Message_Clear
    Keypress_String(szResult)
 End_If

End_While
Button_Remove_All
Return
```

Appendix B: Script Build Errors

This appendix describes the error messages that appear in the *Script Editor Error Help* dialog box For more information about accessing this dialog box, refer to *Building Scripts with the Text Editor* on page 26.

Error Number	Error Message	Description	Fixing the Error
WL1001	You should use two backslash characters ("\\") to represent a single backslash character.	Use the string "\\" to represent the backslash character.	Add one more backslash ' ' in the string, next to the single backslash that is already there.
WL1002	Not enough parameter values for the action; there must be at least two.	The actions `Boolean_And()` and `Boolean_Or()` should have two or more parameters. Each of the actions can have up to five parameters.	Pass 2, 3, 4, or 5 parameters to `Boolean_And()` and to `Boolean_or()`.
WL1003	Call to non-existing script.	This error occurs if the name doesn't exactly match the name of a script or if the script does not exist.	Either call a script that exists or write the non-existing script. Make sure that the name of the script matches exactly, including the case of the letters.
WL1004	Division by 0 is not allowed.	The divisor passed to `Number_Divide()` is zero, and dividing by zero is an undefined mathematical operation.	Make sure to never use 0 as the divisor in `Number_Divide()`.
WL1005	No variable assigned to the value for this action.	The action returns a value, but the script is not using that value.	Create a variable and assign the action's value to the variable.
WL1006	Call to script references non-existing (or incorrect type) variable.	The variable that is being exchanged with the `Call` action cannot be found in the called script, or the variable type doesn't match the type in the called script.	Reference a variable that exists or a variable that is the correct type.

Error Number	Error Message	Description	Fixing the Error
WL1007	The host profile does not exist.	The script activates on a host profile, but that host profile does not exist because the name does not exactly match a host profile or because the host profile has not yet been created.	Either use a host profile that exists or create the host profile. Make sure that the name of the profile matches exactly, including the case of the letters.
WL2001	Text line is too many characters.	The line of text contains too many characters. Long lines of text cause the script to be hard to read or understand.	Break the line into multiple lines. Assign actions to variables instead of passing actions to other actions.
WL2002	Keyword `Script` already used.	There can be one keyword `Script` in each script; this error occurs when more than one `Script` keyword in the script.	Remove any extra `Script` keywords so that there is only one.
WL2003	Keyword `Script` must be the first statement.	The keyword `Script` must be the first statement in a script.	Move all statements before `Script` to after `Script`.
WL2004	Missing `(` after keyword.	Almost all actions and keywords must have a `(` after them. The `(` is missing.	Add a `(` after the keyword.
WL2005	Missing parameter for keyword or action.	The action or keyword requires one or more parameters, and there are no parameters present.	Add the correct number of parameters to the action or keyword.
WL2006	Second parameter for keyword must be TRUE or FALSE. Invalid parameter for keyword. The second parameter must be a quoted string. Unknown terminal type. Unknown keypress key.	This error indicates an invalid parameter passed to an action or keyword. The most common cause of this error is an invalid parameter to `Keypress_Key()`.	Carefully check the parameters passed to the action or keyword and study the error message to help fix the problem.

Error Number	Error Message	Description	Fixing the Error
WL2007	Missing) after parameter to keyword. Missing) after parameter to action.	Most actions and keywords use (and). This action or keyword is missing the right parenthesis.	Add), probably at the end of the line.
WL2008	Extra character(s) after parameter.	There are one or more extra characters after a parameter to an action or to a keyword.	Make sure that after a parameter, there is a right parenthesis) or a comma and another parameter. Make sure that after the last), there are no more characters.
WL2009	The variable name is already in use by a string variable.	Variable names may be declared once. This error occurs when a variable name is used in another variable declaration.	Change the name of the variable in the second declaration.
WL2010	The variable name is already in use by a number variable.	Variable names may be declared once. This error occurs when a variable name is used in another variable declaration.	Change the name of the variable in the second declaration.
WL2011	The variable name is already in use by a boolean variable.	Variable names may be declared once. This error occurs when a variable name is used in another variable declaration.	Change the name of the variable in the second declaration.
WL2012	Invalid activation method.	The method in the Activate action is not valid because it is not spelled correctly or because it's not a valid method.	Check the spelling. The method is case-sensitive. Make sure the method is in the list of activation methods.
WL2013	Non-printable character is not allowed.	A non-printable character is in the line of text. This can happen when pasting text from another program.	Re-type the line of text in the script editor.

Error Number	Error Message	Description	Fixing the Error
WL2014	Blank characters are not allowed in the label. Blank characters are not allowed in the macro name. Invalid character is not allowed. Invalid character is at the beginning of the line.	An invalid character is present in the script text.	Remove or change the character that the error message says is invalid.
WL2015	Parameter cannot be empty. Not enough parameters; there must be 3 for `On_Key`. Not enough parameters; there must be 3 for `On_Input`. The label cannot be blank. The macro name cannot be blank.	A parameter is missing from `Activate`, `Keypress_Key`, `Label`, `Goto`, or `Call`.	Add the missing parameter. All parameters must be on the same line of text.
WL2016	Too many parameters.	The action contains too many parameters.	Check how many parameters the action needs and remove the extras.
WL2017	Invalid key value; must be 1-0xFFFF.	The key value for `Activate(On_Key)` cannot be zero.	Change the zero value to a valid key value.
WL2018	Invalid key modifier; must be `None` or `Alt` or `Ctrl` or `Shift`.	`Activate(On_Key)` contains a modifier that is not recognized or that is not supported.	Change the modifier to match exactly one of the supported modifiers; `None` or `Alt` or `Ctrl` or `Shift`.
WL2019	Missing `:` after keyword.	There must be a colon (`:`) after the keyword, before the label, for the actions `Goto`, `Label`, and `Call`.	Add a colon after the keyword, before the label.
WL2020	Extra characters after action.	There are some characters after the end of the action, after the last `)`.	Remove all characters after the last `)`. Each action must be on a separate line.

Error Number	Error Message	Description	Fixing the Error
WL2021	The value `Goto` is an action, which is not allowed. The value `Activate` is a keyword, which is not allowed. No spaces or tabs are allowed in variable names. Unknown string variable.	A keyword or variable is being used for a variable name, or a variable name has spaces or `Activate` is using an unknown variable.	Use a variable name that is not an action or keyword. Make sure there are no spaces or tabs in the variable name. For `Activate`, define the variable before using `Activate`.
WL2022	Missing `=` after variable.	When setting a variable, use `=`.	Add a `=` after the variable.
WL2023	Missing assignment for variable.	After the `=` there must be a variable, action or constant, all on the same line.	If the assignment is split into two lines, combine them. If there is nothing after `=` add a variable or action or constant.
WL2024	Invalid assignment for variable.	A variable is being assigned to an unknown variable or to an unknown action or to an incorrect value.	If assigning to a variable or action, correct the case-sensitive spelling of the variable or action.
WL2025	Unknown action.	The script contains an action that is not known. Action names are case-sensitive.	Correct the spelling or the case of the action name. Make sure the `_` is in the correct place in the name.
WL2026	Variable type is not the same as the action return type.	The return type of the action must be the same as the type of the variable.	Change the type of the variable or use another variable that has the same type as the action.
WL2027	Variable type is not the same as the variable type.	The type of the two variables must be the same.	Change the type of the variable or use another variable that has the same type.
WL2028	Variable type is not the same as the assignment type.	The type of the two variables and the constant must be the same.	Change the type of the variable or constant. Use another variable or constant that has the same type.

Error Number	Error Message	Description	Fixing the Error
WL2029	No closing quote is found in string. The parameter is not valid.	A string must be surrounded by quotation marks ("). A parameter must be a string, number, variable, or action.	If the string is missing a quotation mark, add one to the end of the string. If the parameter is not valid, check its spelling and the case of its letters. If the parameter is a variable, make sure the variable is declared above this line.
WL2030	The label cannot be an action. The label cannot be a variable. The parameter type doesn't match the return type for the action. The parameter type must be the type.	Labels cannot be variables or actions. If an action is used as a parameter, the action return type must be the same as the parameter type.	For labels, use a unique name that is not an action or variable name. Instead of using an action as a parameter, use a variable that calls the action in a line before.
WL2031	Unexpected character.	The line of text is not quite right because a character is extra or out of place.	Use the error message information to help determine which character is incorrect.
WL2032	Not enough parameter values.	The action is missing one or more parameters.	Check the specification for the action and add parameters to it.
WL2033	Could not open file; error.	A script file could not be opened, probably because the user does not have the correct permission for the file or its folder.	Verify read and write permissions for the file and its folder.
WL2034	File is too small.	A script file must contain at least three bytes so that the script editor can determine if it's Unicode, UTF-8, or ANSI text.	Use a different file that is larger.
WL2035	File encoding not supported: Unicode big endian.	Script files may be encoded as Unicode (little endian), ANSI, or UTF-8. Script files cannot be encoded as Unicode big endian.	Open the file in Notepad; select **File > Save As > Encoding**; change to something other than Unicode big endian.

Error Number	Error Message	Description	Fixing the Error
WL2036	Could not write to file.	A script file could not be written, probably because the user does not have the correct permission for the file or its folder.	Verify read and write permissions for the file and its folder.
WL2037	Memory allocation failed.	There is not enough memory left for the program to run correctly.	Restart the computer.
WL2038	Too many nested calls.	An action has a parameter that is an action that has a parameter that is an action and so on, too many times.	Make one action call per line of text, assigning the return value to a variable. Use the variable as a parameter to another action. Do not use actions as parameters.
WL2039	Script error.	The script contains a generic problem.	Use the error message to help determine the cause of the problem. Try commenting-out or removing some lines of the script to isolate the problem.

Appendix C: Wavelink Contact Information

If you have comments or questions regarding this product, please contact Wavelink Customer Service via email or telephone.

Email: customerservice@wavelink.com

Phone: 1-888-699-WAVE(9283)

Index

T

V

W