AWScript Reference

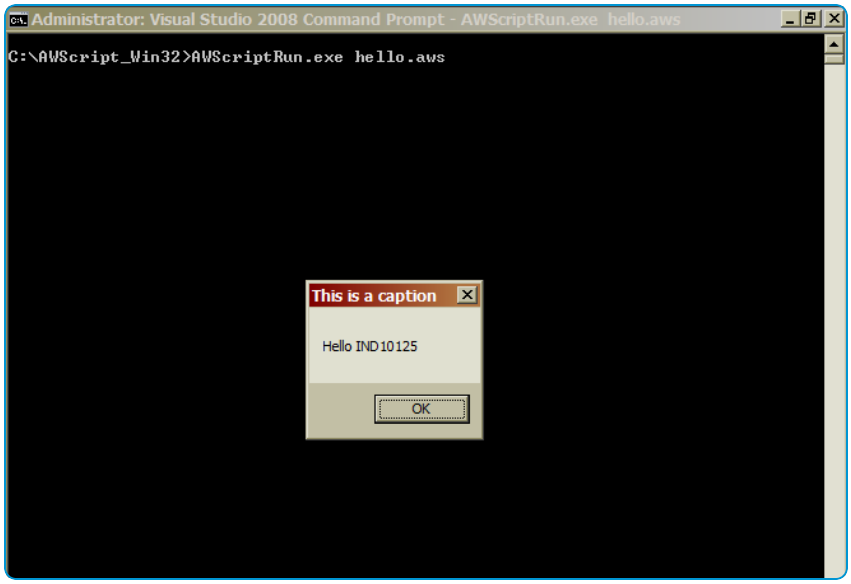
AWScript is a component that implements device-side scripting capabilities for Windows Mobile and Windows CE devices. It uses a dialect of BASIC as its core scripting language and adds AirWatch specific extensions on top. An AWScript script is a plain ascii or utf-8 text file with an extension of .aws. This file extension is handled specially by the other agent components like the Application Manager.

Follow the steps below:

1. Extract the 7z archive to a directory on your computer. Now create the following file using Notepad or another Text editor.

```
'
' Hello AWScript
'
ComputerName = regreadstring("HKLM","System\\CurrentControlSet\\
Control\\ComputerName\\ComputerName", "ComputerName")
Message = "Hello " + ComputerName
MessageBox("This is a caption", Message, 0)
```

2. Now open a command prompt and execute AWScriptRun.exe with the above file as argument. You should see a message box with your computer name.



Language Reference

Keywords

The following are reserved keywords and may not be used as variable or function names:

- MOD
- AND
- OR
- NOT
- LET
- DIM
- IF
- THEN
- ELSE
- FOR
- TO
- STEP
- NEXT
- WHILE
- WEND
- DO
- UNTIL
- EXIT
- GOTO
- GOSUB
- RETURN
- END
- TRUE
- FALSE

Built-In Functions

Function	Description
ABS(num)	Returns the absolute value of a specified number.
SGN(num)	Returns an Integer value indicating the sign of a number.
SQR(num)	Returns the square root of a number.
FLOOR(num)	Returns the largest integer less than or equal to the given numeric expression.
ROUND(num)	Returns a numeric value, rounded to nearest integer.
RND	Returns a random value in the range 0.0 to 0.99.
SIN(rads), COS(rads), TAN (rads)	Returns trigonometric ratios of the angle specified in radians.
ASIN(num), ACOS(num), ATAN(num)	Returns angles from ratios specified.
EXP(num)	Returns a Double value containing e (the base of natural logarithms) raised to the specified power.
LOG(num)	Returns natural logarithm of a specified number.
ASC(char)	Returns an Integer value representing the character code corresponding to a character.
CHR(num)	Returns the character associated with the specified character code.
LEFT(string, len)	Returns a string containing a specified number of characters from the left side of a string.
LEN(string)	Returns an integer containing the number of characters.
MID(string, start, len)	Returns a string containing a specified number of characters from a string. Start position is indexed

Function	Description
	from zero.
RIGHT(string, len)	Returns a string containing a specified number of characters from the right side of a string.
STR(num)	Returns a String representation of a number.
VAL(string)	Returns the number contained in the string.
PRINT param1, param2 . .paramN	Formats and Prints the parameters to standard output. This is not available on Windows Mobile/CE. End the statement with a semi-colon to append a new-line.

Control Structures

Branching

The GOTO statement transfers control to a labelled statement. The GOSUB statement transfers control to a sub-routine

GOTO *label*

```
first:
print "Hello "
print "World "
goto first
```

GOSUB *label*

```
GOSUB PrintMessage1
GOSUB PrintMessage2
end

PrintMessage1:
   print "Hello "
return

PrintMessage2:
   print "World";
return
```

If Statements

The syntax for a single line, If statement has the form:

IF *condition* THEN *statement* ELSE *statement2*

```
if (rnd * 100.0) <= 50 then print "Heads !"; else print "Tails!";
```

While there is no direct support for multi-line if blocks, it can be simulated using the single line IF and GOTO as illustrated below:

```
IF x <= 1- THEN GOTO line 10
    statement1
    statement2
Line10:
```

For Loops

The syntax of a FOR statement is illustrated below:

```
For i = 1 TO 10 Step 1
    Print i
Next i
```

While Loops

WHILE statements are terminated by a WEND keyword.

```
a = 1
WHILE a <= 10
    Print a
WEND
```

Do Until Loops

```
b = 1
DO
    Print b
UNTIL a > 10
```

AirWatch Extensions

A set of additional functions are available for use with Provisioning and general device side scripting.

Function	Description
beep()	Plays a notification alert sound.
udid()	Returns a string representing the AirWatch Device UDID which is a unique identifier for each device.
crlf	Represents a platform specific line ending, can be used in print statements and message boxes.
instr(sTarget, sSearch, iCaseSensitive)	Returns the position of the search string within the target string. Set iCaseSensitive to 1 to do a case sensitive search, 0 to ignore case. Returns -1 if the search string is not found.
filecopy(sSrc, sDest, iFail_if_ exists)	Copies a File. Specify source and destination paths for a file copy. iFail_if_exists indicates if the function should fail if the destination file already exists. (Valid values are 1 and 0) Returns 1 on success, 0 on failure.
filemove(sSrc, sDest)	Moves a file to a different path. Specify source and destination paths for moving the file. Returns 1 on success, 0 on failure.
filedelete(sSrc)	Path to a file to be deleted. Returns 1 on success, 0 on failure.
dircreate(sSrc)	Full Path to a directory to be created, intermediate missing directories will also be created. Returns 1 on success, 0 on failure.
dirrename(sSrc, sDest)	Moves a directory, specify source and target directories. Returns 1 on success, 0 on failure.
dirremove(sSrc)	Deletes a directory and all its contents, specify the full path to the directory. Returns 1 on success, 0 on failure.
abort(sMessage)	Deletes a directory and all its contents, specify the full path to the directory. Returns 1 on success, 0 on failure.
run(sExe, sArguments, iTimeout)	Runs an executable on the device, specify complete path to the executable, any arguments it takes or an empty string if there are no arguments. Timeout in seconds to wait for process to finish. Returns 1 if program was started successfully, 0 on error.
wakeup()	Wakes up the device from sleep. No arguments and return values.
regaddkey(sHive, sPath)	Adds a key to the registry, specify hive which can be

Function	Description
	any of HKCU, HKLM. Path to the new key (Example: Software\AirWatch) Returns 1 on success, 0 on Error
regremovekey(sHive, sPath)	Removes a key from the registry, the key must not have any child keys. Specify Registry Hive and Path. Returns 1 on success, 0 on Error.
regwritestring(sHive, sPath, sName, sValue)	Adds a string property to the registry. Specify hive, registry key, property name and property value Returns 1 on success, 0 on Error.
regreadstring(sHive, sPath, sPropertyName)	Read a string from the registry. Specify hive, registry key, property name Returns string value, empty string if property doesn't exist.
regwritedword(sHive, sPath, sName, sValue)	Adds an Integer property to the registry. Specify hive, registry key, property name and property value. Returns 1 on success, 0 on Error.
provisioning_getproductversion ()	Returns the version of the provisioning product.

Examples

Picking a file based on a stanza parameter

- Based on the encryption setting of a symbol 9090g, pick either a wpa2 or a wep setting file and copy it to 9090g-ce5_irr.apd.
- Based on the country setting, pick a country specific configuration.
- Based on the number of sessions for the gls application.

```
if regreadstring("HKLM", "Software\\ACME\\Wireless", "symbol-9090g-ce5_
encryption") = "wpa2" then filecopy("9090g-ce5_irr_wpa2.apd", "9090g-ce5_
irr.apd", 0)
if regreadstring("HKLM", "Software\\ACME\\Wireless", "symbol-9090g-ce5_
encryption") = "wep" then filecopy("9090g-ce5_irr_wep.apd", "9090g-ce5_irr.apd",
0)
```

```
if regreadstring("HKLM", "Software\\ACME\\General", "country") = "US" then
filecopy("9090g-ce5_reg.apd.default", "9090g-ce5_reg.apd", 0)
```

```
if regreadstring("HKLM", "Software\\ACME\\General", "country") = "JP" then
filecopy("9090g-ce5_reg.apd.japan", "9090g-ce5_reg.apd", 0)
```

```
ansi_reg_src = ansi.reg + regreadstring("HKLM", "Software\\ACME\\Apps", "gls_
num_of_sessions") + "_sessions"; if fileexists("\\Program Files", ansi_reg_src)
then filecopy(ansi_reg_src, "ansi.reg", 0)
```

Generate a configuration file based on stanza settings

```
if regreadstring("HKLM", "Software\\ACME\\Wireless", "symbol
    -9090g-ce5_ encryption") <> "wpa2" then goto end_encrypret
    = mergetemplate("\\Program Files\\fusion.ini.wpa2tmp", "\\Program
    Files\\fusion.ini", "HKLM", "Software\\ACME")
end_encryp:
```

The fusion.ini.wpa2tmp is listed below:

```
# FUSION CONFIG FILE
# This defines a configuration file for the Fusion
Radio
#######################################
"Profile Name"="DC"
"ESS_ID"={{WIRELESS\wpa2_essid}}          #(wpa2_essid from stanza file goes here)
#Parameters for IP Addressing Type selection
#Type: zero bit=0 for DHCP, zero bit=1 for Static
IP
"IPAddressType"=dword:00000000            #30mW, 15mW, 5mW, 1mW, 0= full power
"Transmit Power"=dword:00000000           #0=CAM, 1=Fast Power Save, 2=Max Power Save
"BatteryUsage"=dword:00000001             #0=Infrastructure, 1=Ad-Hoc
"OperatingMode"=dword:00000000            #1=2412 MHz, 2=2417MHz, 3=2422MHz, 4=2427MHz,
"Channel"=dword:00000001                  5=2432MHz, 6=2437MHz, 7=2442MHz, 8=2447MHz, 9=2

"Authentication"=dword:00000000           #0=None, 1=LEAP, 2=PEAP, 3=TTLS
#Key to selectencryption type
"Encryption"=dword:00000006               #0=open, 1=40bit WEP, 2=128bit WEP, 3=TKIP
"PassKey"={{WIRELESS\wpa2_psk}}
#EOF cfg
```

...And the generated file fusion.ini has the replaced values

```
# FUSION CONFIG FILE
# This defines a configuration file for the Fusion
Radio
########################################
"Profile Name"="DC"                                   #(wpa2_essid from stanza file goes here)
"ESS_ID"=VZkBobWwB6Tz
#Parameters for IP Addressing Type selection
#Type: zero bit=0 for DHCP, zero bit=1 for Static
IP
"IPAddressType"=dword:00000000                        #30mW, 15mW, 5mW, 1mW, 0= full power
"Transmit Power"=dword:00000000                       #0=CAM, 1=Fast Power Save, 2=Max Power Save
"BatteryUsage"=dword:00000001                         #0=Infrastructure, 1=Ad-Hoc
"OperatingMode"=dword:00000000                        #1=2412 MHz, 2=2417MHz, 3=2422MHz, 4=2427MHz,
                                                      5=2432MHz, 6=2437MHz, 7=2442MHz, 8=2447MHz, 9=2
"Channel"=dword:00000001                              #0=None, 1=LEAP, 2=PEAP, 3=TTLS
"Authentication"=dword:00000000
#Key to selectencryption type                         #0=open, 1=40bit WEP, 2=128bit WEP, 3=TKIP
"Encryption"=dword:00000006
"PassKey"=2CQH74VfWV3AJ3vgcceZ8Jegu
Mi87FIlj58TNfLix395OZOB30LjXSNk3Lw7yzO
#EOF cfg
```

Provisioning Conditional Execution

Skip all download actions for a product

```
count = provisioning_getactioncount();
for i = 0 to count - 1
 if provisioning_getactiontype(i) = "Download" then provisioning_ setactionstate
(i, 0)
next i
```