

# Integrating Mercury Interactive Virtual User Generated Scripts with HP OpenView Internet Services



Summary .....	2
OVIS Script Probe .....	2
Mercury LoadRunner™ .....	2
Benefits of Mercury LoadRunner™ .....	2
Mercury VuGen .....	2
Platform Limitation .....	3
License Clarification.....	3
Requirements.....	3
Training Recommendations .....	4
Guides and Tutorials .....	4
Mercury LoadRunner QuickStart Guide .....	4
VuGen Tutorial: Tutorial.pdf.....	5
VuGen Manual .....	5
Steps for incorporating a VuGen script into OVIS .....	6
Overview .....	6
Details of OVIS Script Target Configuration.....	7
Example of how to configure the OVIS Script Probe target .....	7
Configuring multi-step transactions .....	8
Licensing and Support.....	8
Troubleshooting and Tips:.....	9
Appendix 1 .....	11
How to use "lparse.vbs" .....	11
Contents of lparse.vbs.....	11

## Summary

This document details the processes for incorporating Mercury Interactive Virtual User Generator (VuGen) scripts into hp OpenView Internet Services (OVIS).

### OVIS Script Probe

OVIS provides a mechanism for running scripts called the Script probe. With the Script probe you can monitor an application through a script without having to write a custom probe. Scripts may be written in VBScript, JavaScript or Perl or run using batch, command and UNIX shells.

This document focuses only on incorporating Mercury VuGen scripts into OVIS.

NOTE: Mercury VuGen is a shipping component of Mercury LoadRunner™.

### Mercury LoadRunner™

Using minimal hardware resources, Mercury LoadRunner emulates hundreds or thousands of concurrent users to apply production workloads to virtually any client platform or environment. Mercury LoadRunner load stresses an application from end to end—applying consistent, measurable, and repeatable loads—then uses the data to identify scalability issues that would impact real users in production.

As it drives load against the system, Mercury LoadRunner captures the end-user response times of key business processes and transactions to determine if service-level agreements (SLAs) can be met. Non-intrusive, real-time performance monitors obtain and display performance data from every tier, server, and system component, and diagnostics probes gather code-level data to isolate bottlenecks at the SQL or method level. This combination of end-user, system, and code-level visibility dramatically reduces time to problem resolution.

#### **Benefits of Mercury LoadRunner™**

- Minimize the risk of deploying systems that do not meet performance requirements.
- Minimize hardware and software costs by accurately predicting system capacity.
- Begin intelligent service-level management before go-live.
- Shorten test cycles to accelerate delivery of high-quality applications.
- Pinpoint end-user, system, and code-level bottlenecks rapidly and with ease.
- Reduce the cost of defects by testing early in the development cycle.

### Mercury VuGen

The Virtual User Generator, also known as VuGen, enables you to develop scripts for a variety of application types and communication protocols.

When testing or monitoring an environment, you need to emulate the true behavior of users on your system. Mercury VuGen can emulate an environment in which users concurrently work on, or access your system.

To do this emulation, the human is replaced with a virtual user, or a Vuser. The actions that a Vuser performs are described in a Vuser script. The primary tool for creating Vuser scripts is the Mercury Virtual User Generator, VuGen.

Mercury VuGen is a shipping component of Mercury LoadRunner™ so in order to install VuGen you download a 10 day trial of Mercury LoadRunner from [www.mercury.com](http://www.mercury.com).

**Platform Limitation:** VuGen records sessions on Windows platforms only. However, a recorded Vuser script can run on both Windows and UNIX platforms. The OVIS Script probe integration with VuGen scripts is only offered on Microsoft Windows.

**License Clarification:** The recording functionality contained within Mercury VuGen does not require a license to use when used for creating scripts for hp OpenView Internet Services. The functionality contained within Mercury LoadRunner™ excluding VuGen recording capabilities does require a license to use.

## Requirements

- OpenView Internet Services (OVIS) 6.0, 6.10, 6.11 (English only)
- Mercury Virtual User Generator (VuGen) 8.1.0 (Available with Mercury LoadRunner)
- Virtual User Generator installed with each probe location where the VuGen script is expected to execute.
- The OVIS Script Probe will need the absolute path to the VuGen script for it to execute it properly.
- The OVIS Script Probe will need the absolute path to the VuGen executable mmdrv.exe.

To get the most use out of this document please have Mercury Virtual User Generator 8.1.0 or higher installed (8.1.3 recommended). This white paper contains references to manuals and guides provided with those installations.

# Training Recommendations

If you are new to OVIS Script probe please review the information regarding the Script Probe in the OVIS User's Reference Guide. There are several in depth examples of how to configure and run the OVIS Script Probe, along with general troubleshooting information.

## Guides and Tutorials

The following section covers research materials, preparation guides, and supplemental materials. If you are new to Mercury LoadRunner™ or VuGen technology please review the section below and follow the guidelines offered.

After installing Mercury LoadRunner™ or Mercury VuGen the following guides and tutorials are available on the installed system. Below is a brief synopsis of each along with recommendations for new users.

Note: The location for the default installation is typically c:\program files\Mercury Interactive\VUser Generator.

**Mercury LoadRunner QuickStart Guide:** LR\_QuickStart.pdf

**Location:** <Mercury Installation Directory>\VUser Generator\tutorial

**Contents:** This guide offers a short, step-by-step overview and introduction to using Mercury LoadRunner™.

Introduction to Mercury LoadRunner™ Technology	<ul style="list-style-type: none"><li>a. The Testing Process.<ul style="list-style-type: none"><li>1. Creating the script</li><li>2. Designing the Scenario</li><li>3. Running the Scenario</li><li>4. Analyzing the results.</li></ul></li><li>b. Mercury LoadRunner™ Components.</li><li>c. Sample Application - Mercury Tours.</li></ul>
Creating the Script with VuGen	<ul style="list-style-type: none"><li>a. How do I start recording user activity.</li><li>b. How do I record a business process to create a script.</li><li>c. How do I view the script.</li><li>d. How do I verify the script recorded my actions.</li><li>e. How do I measure the business processes.</li></ul>
Using Web (Click and Script)	<All Parts> <p>For OVIS users familiar with WebRecorder there is a powerful tool in Mercury VuGen called Web (Click and Script) that is comparable to the ease of use found with OVIS WebRecorder but with greater flexibility and functionality. Web (Click and Script) is specifically designed for recording web transactions without complicated modifications.</p>

**VuGen Tutorial:** [Tutorial.pdf](#)

**Location:** <Mercury Installation Directory>\VUser Generator\tutorial

**Contents:** The Mercury LoadRunner Tutorial guide offers useful background on Mercury LoadRunner™ and VuGen, with an overall recording process explained in detail. Users new to Mercury LoadRunner™ or Mercury Virtual User Generator should review the following sections in the Mercury LoadRunner Tutorial.

Introducing LoadRunner	<All Parts>
The Power of LoadRunner	<All Parts>
Building Scripts	<All Parts>
Playing Back your script	<All Parts>
Solving Common Playback problems	<All Parts>
Preparing a Script for Load Testing	How do I measure business processes? How do I verify Web page content? Did my test succeed?

- Note: If you are an hp partner with access to hp SSU, prior to implementing this integration, it is highly recommended to review the computer based training on VuGen located on the HP education site:

<http://www.hp.com/go/ssu>

**VuGen Manual:** [db.pdf](#)

**Location:** <Mercury Installation Directory>\VUser Generator\help\

**Contents:** Covers Virtual User Generator Basics, Applications, and Advanced topics. Use this manual for developing VuGen scripts. Included in this manual are examples on developing Citrix, Siebel, and SAP application scripts.

Note on Citrix, Siebel, and WebTransactions:

Rich documentation on how to create Citrix, Siebel, and WebTransactions exists in the VuGen Manual (db.pdf). Before attempting a new recording with VuGen on an application type where you have not previously used VuGen to record, please review the db.pdf manual. Also if experiencing difficulties with a recording please refer to this manual first since it contains an extensive amount of information.

# Steps for incorporating a VuGen script into OVIS

## Overview

Integrating a VuGen script into OVIS relies on the following steps. This document will only walk through steps directly associated with OVIS. For VuGen specific materials please refer to *The Mercury LoadRunner Tutorial*, *LoadRunner Quick Start Guide*, or the *VuGen reference manuals*. The Mercury manuals cover the necessary steps to create robust VuGen scripts. The outline below merely shows the steps that a typical OVIS-VuGen user would go through.

1. Create a Mercury VuGen script recording.
2. Successfully replay the recorded transaction in VuGen.
  - a. If the recorded transaction cannot be successfully replayed in VuGen then it will not work when run with OVIS Script Probe.
3. If Multi-Step output is a requirement for monitoring the recorded transaction add the necessary start/end transaction keys to the VuGen script.
4. Add any robust scripting features that may be necessary for repeated executions. For example modify the VuGen script to handle dynamic content, add pattern matching checks, or modify the script to support variable inputs (file or command line).
5. Save the VuGen transaction and if necessary export the runtime element to a zip file so it can be easily transported to the necessary OVIS Probe Locations.
  - a. Do not forget to unpack the zipfile on the desired OVIS Probe Locations if you had exported to zip.
  - b. The directory location of the VuGen executable and VuGen scripts on the Probe Location will be key in configuring the Script Target. So be sure to track them carefully.
  - c. If you relocated a VuGen script to another location from where it was recorded, please replay the transaction in VuGen before attempting to integrate with OVIS.
6. In the OVIS Configuration Manager create a new target of probetype Generic Script Service.
7. Configure the new target to reference the newly created VuGen Script and VuGen executable location.
  - a. For the probe to execute properly an absolute path will be required by the probe.
8. Add Objectives.
9. Add a Probe Location.
  - a. This location will require VuGen be installed and will require the VuGen script and LRParse.vbs (located in Appendix 1 of this document) files.
10. Save the configuration.
  - a. If the Target being configured in OVIS is to run locally on the OVIS Management Server then you can test the target before saving using the 'Test on Management Server' feature.
  - b. If the target is remote to the OVIS Management Server, after saving the target it can be tested with the TIPs 'rerun' feature from the OVIS Dashboard.
11. Check results and troubleshoot issues if necessary.

## Details of OVIS Script Target Configuration

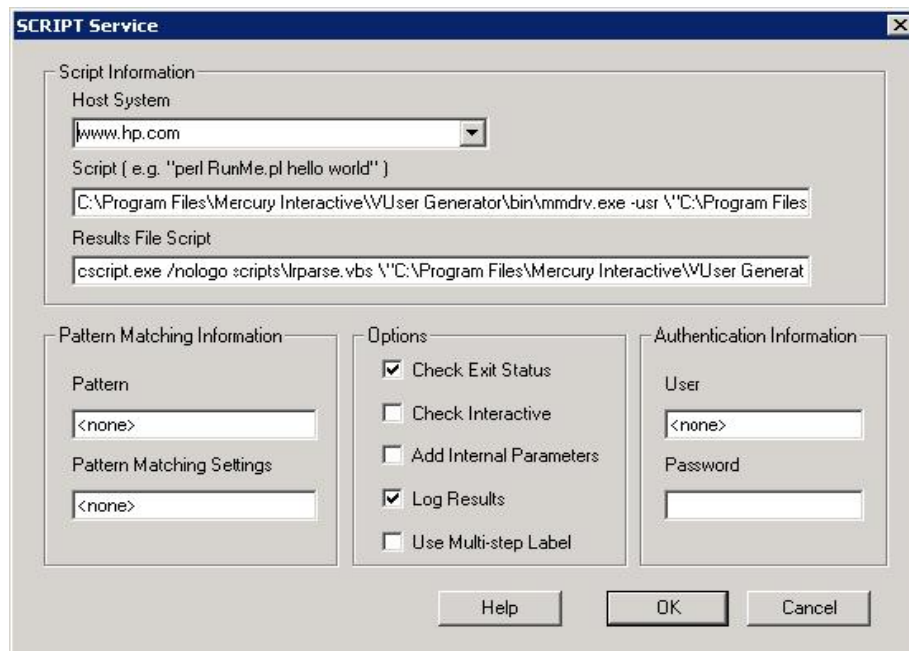
The OVIS Script Probe is a tool that can take script inputs and script outputs for the purpose of monitoring a transaction. In the case of integrating VuGen Scripts with the OVIS Script probe you will need the following.

1. The absolute location and name of the VuGen script to execute on the Probe Location.
2. The absolute location of mmdrv.exe on the Probe Location.
  - a. Mmdrv.exe is usually located at "c:\Program Files\Mercury Interactive\VUser Generator\bin".

### Example of how to configure the OVIS Script Probe target

We are going to use OVIS to launch the VuGen script **HpWeb.usr** with VuGen's **mmdrv.exe** program and output the results to the log file **HpWeb.usr\_out.log**. The Script probe then parses the output log file HpWeb\_out.log with the Results Script File lrpars.vbs (provided in Appendix 1). It searches for errors and then reports the appropriate Availability and Response Time information into OVIS for processing.

Shown below is an example of how to take the VuGen script syntax and the Results File script syntax and configure it in the OVIS Configuration Manager Script Probe dialog.



**Host System:** Use this field as a target naming field for appropriate presentation.

**Script:** This field specifies the execution parameters for the VuGen Script. Included in the field is the VUser script name/path, mmdrv name/path, and an output file name/path.

(Note the example syntax is actually on a single line, it is shown on multiple lines for formatting.)

```
c:\program files\Mercury Interactive\VUser Generator\bin\mmdrv.exe -usr \"C:\Program Files\Mercury Interactive\VUser Generator\scripts\citrix\citrix.usr\" -VuGen_win 0 -drv_log_file \"c:\Program Files\Mercury Interactive\VUser Generator\hp_citrix_out.log\"
```

**Results File:** This field specifies the lrpars.vbs/name location for execution and takes as input the VuGen script output.

```
cscript.exe /nologo c:\temp\lrparse.vbs \"c:\Program Files\Mercury Interactive\VUser Generator\hp_citrix_out.log\"
```

### **Configuring multi-step transactions**

When setting up multi-step transaction monitoring, your VuGen script can have start and end calls around each transaction that you want to measure. Transactions without start and end calls will not be broken out in OVIS. The VuGen output file will then provide transaction start and end information used by OVIS to report each step of the multi-step transaction. Each complete transaction in the VuGen output will correlate to one step in OVIS with a Target Name (step name), Response Time, and Availability (0 or 1).

The following is an example of the start and end calls in a VuGen script:

```
lr_start_transaction("Login");  
...  
lr_end_transaction("Login", LR_AUTO);
```

And the following is an example of what is logged:

```
Test_transaction.c(60): Notify Transaction Login ended with Pass status (Duration: 1.1717)
```

\*The above start and end transaction times can be created through two methods. The first way is by adding the start/end transaction methods to the VuGen script directly. The second way is by using the graphical transaction workflow in the VuGen recorder.

### **Licensing and Support**

The standard OVIS target licensing model applies to Script probe targets configured in OVIS. For example, one Script probe target configured at one Probe Location will require one Standard OVIS Target license.

Support for VuGen is available through the standard HP Support Chain. Although if you have an existing support contract with Mercury Interactive you are recommended to go directly to Mercury for questions regarding VuGen. For questions regarding the OVIS Script probe with VuGen please contact HP Support.



## Troubleshooting and Tips:

### Data from a VuGen script does not show up in the OVIS Dashboard

1. In the OVIS Configuration Manager interface, select the 'Status' tree item. Verify that the status of the VuGen script target is "Available".
2. If the status is "Unavailable", open the configuration for the target in OVIS Configuration Manager. Copy the command line to a text editor, remove escape characters. Execute the modified script command line in a command window and check the contents of the output log file.
3. If the log file contains errors, attempt re-running the script in VuGen. If the probe fails in VuGen then correct the problems in VuGen before attempting to integrate it with OVIS.
  - a. If the VuGen script successfully executes in VuGen but not in OVIS then verify the following.
    - i. The OVIS Script probe target is configured for the correct target.
    - ii. The paths specified in the OVIS Script probe target are absolute.
    - iii. The version of the VuGen script is correct.
    - iv. Verify there are user specific environment variables (NTLM) that the VuGen script is dependant upon that are different when used by the OVIS Service. The OVIS Service by default will execute a probe as LocalSystem. If you require specific user data or permissions for your transaction then consider using the RunAs feature for impersonation.

### Tips for using VuGen with OVIS

1. Only one multi-step service target created with VuGen should be placed under an OVIS Service Group. Steps from distinct service targets are not distinguishable when placed under a single Service Group.
2. VuGen has 3 modes for recording HTTP targets: HTTP, HTML, and Click and Script.
  - a. The HTTP mode records the underlying URL requests that are made to the server.
  - b. The HTML mode recognizes the content of each page and more closely resembles a real user interaction.
  - c. The Click and Script uses the DOM to navigate through web pages.

If the text of a link on a web page changes, this will not be detected by a VuGen script running in URL mode but will be detected by a script in HTML mode and the Click & Script mode. The VuGen HTTP mode is similar to the OVIS WebRecorder URL mode and the VuGen HTML mode is similar to the OVIS WebRecorder IE mode. While the Click and Script mode uses very similar approaches to web monitoring as the WebRecorder IE mode.

### Tips for using Scripts in OVIS

1. Screen savers popping up can cause problems for certain kinds of probes.
2. Depending upon the type of service being monitored there may be limitations on how many scripts can be run at a time. For example SAP can have limitations for user logins.
3. Test applications that require an interaction with the probe system Desktop, must have the "Allow service to interact with desktop" option checked on the "HP Internet Services" service.

4. Some applications may not run or record appropriately in terminal server mode. For example Citrix does not record well through terminal server.

# Appendix 1

## How to use "lrparse.vbs"

The following Appendix contains lrparse.vbs. To use lrparse in OVIS, copy the entire contents within the script section below to a file named lrparse.vbs.

Make sure that only the code contents of the section are copied to the new file and not the content header. For example to create the lrparse.vbs file, create a new file called 'lrparse.vbs' and copy the contents after 'Contents of lrparse.vbs' up to the start of the following section. After you copy the contents to the lrparse.vbs file then save your changes. For this file to be distributed by OVIS to OVIS Probe Locations please copy the file to the <OVIS install dir>\newconfig\distrib\ directory.

### Contents of lrparse.vbs

```
' This script parses a Mercury Interactive LoadRunner/VuGen output file <output.log>
' for Availability and Response Time in multi-step
' transactions.

' A transaction result contains a Name, Pass or Fail condition, and duration time.
' The pass, fail, and duration time values for each transaction are
' totaled to calculate Availability and Total Response Time.

' Sample transaction line (Before LoadRunner 7.8):
' Action1.c(32): Notify Transaction 2_access_cwf_community ended with Pass status (Duration: 22.8923).
' Sample transaction line (LoadRunner 7.8):
' user_init.c(55): Notify: Transaction 1_login ended with Pass status (Duration: 2.5309).
' Sample transaction line (European System LoadRunner 7.8):
' Action1.c(32): Notify: Transaction 2_access_cwf_community ended with Pass status (Duration: 22,8923).

' The results below are echoed to stdout for OVIS processing
' Each transaction will generate 3 values for an OVIS step
' "StepName=2_access_cwf_community"
' "StepAvailability=1"
' "StepResponseTime=22.8923"

' If a step fails, the transaction line will be echoed to stdout for the OVIS error message.
' Note, the last transaction failure is used for the OVIS error message.
' "ErrorInfo=Action1.c(88): Notify Transaction test_login ended with Fail status (Duration: 0.1345).
' "StepName=test_login"
' "StepAvailability=0"
' "StepResponseTime=0.1345"

' Total Availability and Response Time are based on the values for each step
' If failCount > 0, "Availability=0"
' If failCount equals 0 and passCount > 0 "Availability=1"
' and the total "ResponseTime" is equal to the sum of transaction duration times

' If the file to parse is not found, "Availability=0" and "ResponseTime=0".

strFileName = WScript.Arguments.Item(0)
Set fsoObject = CreateObject("Scripting.FileSystemObject")

On Error Resume Next
Set fReport = fsoObject.OpenTextFile(strFileName, 1)
if Err.Number <> 0 then
' Log error message first
WScript.Echo "ErrorInfo=Error opening file " + strFileName
WScript.Echo "Availability=0"
WScript.Echo "ResponseTime=0"
wscript.Quit 1
```

```

end if

availability = 0
responseTime = 0

passCount = 0
failCount = 0

errorFound = FALSE

strQuote = chr(34)
strAvailPattern = "Pass"
strRespPattern = "Duration:"
strNotifyTrans = "Notify Transaction"
strNotifyTrans78 = "Notify: Transaction"
strLocaleDecPoint = GetLocaleDecimalPoint() 'Decimal Point for machine's locale

Do while False = fReport.AtEndOfStream
  strLine = fReport.ReadLine()
  ' Check for the pattern "Duration:"
  pos = InStr(strLine, strRespPattern)
  if (pos > 0) then
    errorFound = FALSE
    call getTransactionName (strLine)
    ' pass the string and pattern position for
    ' response time processing
    call getResponseTime(strLine, pos)
    ' Check if test passed or failed
    call checkStatus(strLine)
  end if
Loop

fReport.Close

' Check for availability
if (passCount > 0) and (failCount = 0) then
  availability = 1
end if

' Make sure there is a decimal point in the response time
responseTime = Replace(responseTime, strLocaleDecPoint, ".")

WScript.Echo "Availability=" & availability
WScript.Echo "ResponseTime=" & responseTime

function getResponseTime(byval strLine, pos)

' example string to get test response time (Duration:)
' vuser_init.c(56): Notify Transaction 1_login ended with Pass status (Duration: 30.2158)
' pos is the pointer to location of Duration:

if (pos > 0) then
  durationTime = mid(strLine, pos, Len(strLine))
  durationTime = Replace(durationTime, "Duration:", "")
  durationTime = LTrim(durationTime)

  ' Locate the decimal point in the duration time
  ' time format xxx.xxxx (four places to the right of decimal point)
  decPtPos = InStr(durationTime, strLocaleDecPoint)
  closePos = InStr(durationTime, ")")
  if (decPtPos > 0) and (closePos > decPtPos) then
    ' Add durationTime to responseTime in its locale
    durationTime = mid(durationTime, 1, decPtPos + 4)
    responseTime = responseTime + durationTime
  end if
end if
end function

```

```

' Make sure there is a decimal point in the duration time
durationTime = Replace(durationTime, strLocaleDecPoint, ".")
WScript.Echo "StepResponseTime=" + durationTime

else
' Always set Error message before StepResponseTime
WScript.Echo "ErrorInfo=Incorrect format for Duration xxx" + strLocaleDecPoint + "xxxx time: " + strLine
WScript.Echo "StepResponseTime=0"
errorFound = TRUE
end if

end if

end function

function checkStatus (byval strLine)

' Check if test passed or failed and update the counters

'Look for the string "Pass"
passPos = InStr(1, strLine, "Pass")
if ((passPos > 0) And (not errorFound)) then
    passCount = passCount + 1
    WScript.Echo "StepAvailability=1"
else
    failCount = failCount + 1
    ' Always set Error message before StepAvailability
    if (not errorFound) then
        ' The transaction failed
        WScript.Echo "ErrorInfo=" + strLine
    end if
    WScript.Echo "StepAvailability=0"
end if

end function

function getTransactionName (byval strLine)

' Get the test Transaction Name
dim firstParm

namePos = InStr(1, strLine, strNotifyTrans, vbTextCompare)
if (namePos > 0) then
    strName = mid(strLine, namePos, Len(strLine))
    strName = Replace(strName, strNotifyTrans, "")
    strName = LTrim(strName)
    ' Find the space at the end of transaction name
    endPos = InStr(strName, chr(34))
    if(endPos > 0) then
        strName = mid(strName, endPos+1, len(strLine))
        endPos = InStr(strName, chr(34))
        strName = mid(strName, 1, endPos - 1)
        WScript.Echo "StepName" + "=" + strName
    else
        WScript.Echo "ErrorInfo=Unable to find '" + strNotifyTrans + "' or '" + strNotifyTrans78 + "' for Step Name: " + strLine
        WScript.Echo "StepName=<unknown>"
        errorFound = TRUE
    end if
elseif (namePos = 0) then
    ' LoadRunner 7.8 version
    namePos = InStr(1, strLine, strNotifyTrans78, vbTextCompare)
    if (namePos > 0) then
        strName = mid(strLine, namePos, Len(strLine))
        strName = Replace(strName, strNotifyTrans78, "")
        strName = LTrim(strName)
    end if
end if

```

```

' Find the second quote in the pair
firstParm = InStr(strName, chr(34))
if (firstParm > 0) then
    strName = mid(strName, firstParm+1, len(strLine))
    endPos = InStr(strName, chr(34))
    strName = mid(strName, 1, endPos - 1)
    WScript.Echo "StepName" + "=" + strName
else
    WScript.Echo "ErrorInfo=Unable to find '"' + strNotifyTrans + "'" or "'" + strNotifyTrans78 + "'" for Step Name: " + strLine
    WScript.Echo "StepName=<unknown>"
    errorFound = TRUE
end if
else
    ' Always set Error message before StepName
    WScript.Echo "ErrorInfo=Unable to find '"' + strNotifyTrans + "'" or "'" + strNotifyTrans78 + "'" for Step Name: " + strLine
    WScript.Echo "StepName=<unknown>"
    errorFound = TRUE
end if
end if

end function

function GetLocaleDecimalPoint()
    value = 1234.00
    localeNumber = FormatCurrency(value,2,-1)
    pos = Instr(1, localeNumber, "0") 'Decimal Point for locale
    pos = Pos -1
    GetLocaleDecimalPoint = Mid(localeNumber, pos, 1)
end function

```