

VManager Script Writers Guide

MPG Validation

February, 2009

Revision 0.3- DRAFT

Internal Use Only – Do Not Distribute





1

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2008, Intel Corporation. All rights reserved.



Contents

1	3	
2	What is VManager?	7
	2.1 Overview	7
	2.2 Features	7
	2.3 Deployment Methods.....	7
	2.3.1 Customization	7
	2.3.2 Portable Preload	8
	2.4 Dependencies.....	8
3	Writing Scripts For VManager	9
	3.1 VManager Script Environment	9
	3.1.1 VManager Folders	9
	3.1.2 VManager Registry Keys	10
	3.2 Install Scripts	13
	3.2.1 Install Script Flow Chart	13
	3.2.2 How Does Install Script Work?	15
	3.2.3 INITIAL PROCEDURES Block.....	15
	3.2.4 APPLICATION CHECK Block	16
	3.2.5 MAIN Block	16
	3.2.6 END PROCEDURES Block	17
	3.3 Config Scripts.....	19
	3.3.1 Config Script Flow Chart	20
	3.3.2 How Does Config Script Work?	22
	3.3.3 INITIAL PROCEDURES Block.....	22
	3.3.4 MAIN Block	23
	3.3.5 END PROCEDURES Block	23
	3.4 Test Scripts.....	25
	3.4.1 Test Script Flow Chart	26
	3.4.2 How Does Test Script Work?	28
	3.4.3 INITIAL PROCEDURES Block.....	28
	3.4.4 APPLICATION CHECK Block	29
	3.4.5 RUN LATER Block.....	30
	3.4.6 APPLICATION EXECUTION Block	31
	3.4.7 TEST EXECUTION Block	32
	3.4.8 END PROCEDURES Block	32
	3.4.9 Clean-up Script	34
	3.5 Script Deployment	36
	3.5.1 XML File How To	36
	3.5.2 Defining the Section to be Displayed in Install/Config Tab.....	38
4	Advanced Features	46
	4.1 VManager Customization Feature	46
	4.1.1 Add Custom Scripts to VManager.....	46
	4.1.2 Customization Feature Requirements	46
	4.1.3 Creating XML Files to Support Custom Scripts	47
	4.1.4 Procedure for Adding Custom Scripts to VManager.....	57
	4.2 Optional Reboot.....	59



4.3	Install Only	59
5	Appendix	60
5.1	Script Writing BMKs for VManager	60
5.2	VMLite and Script Writing for VManager	60
5.3	Troubleshooting.....	60
5.4	Contact Information	61



Revision History

Document Number	Revision Number	Description	Revision Date
N/A	0.1	Initial Draft	October 2008
N/A	0.2	Content Added	January 2009
N/A	0.3	VMLite Content Added	February 2009

§



2 *What is VManager?*

2.1 Overview

VManager is a Windows application that provides a graphical user interface to manage the execution of a collection of automation scripts to configure Windows software settings, install drivers, install applications, and execute automated tests. VManager uses XML files to obtain configuration information about each of the automation scripts and it uses this information to control the presentation and execution of the scripts.

2.2 Features

Most important VManager features include:

- User friendly interface that provides information about available scripts.

- Install, configure and test script execution.

- Allowance for multiple install script selections and their execution to minimize user's actions.

- "Install Only" feature for test scripts that allows test script preparation and execution on standalone test systems.

- "Optional Reboot" feature that minimizes time delay during multiple install scripts execution.

- Capability of adding custom scripts.

- Using flexible XML file structure for configuration purposes.

- Working with Windows registry keys to store important script information.

2.3 Deployment Methods

2.3.1 Customization

Customization is one of script deployment methods. It allows users to add their own custom scripts to VManager existing script selection. Customization is widely explained in **chapter 3: Advanced features**.



2.3.2 Portable Preload

Portable preload allows using all the available VManager features by setting up Portable Preload Server and Network.

Portable Preload Network is an isolated network with a Portable Preload server and client systems. Portable Preload Server is a networked computer system that has installed and configured Portable Preload Server software in order to serve Preloads to other client systems on the local Portable Preload Network.

After user uses Portable Preload to setup test platforms, VManager and its features are available for portable network clients (test platforms).

Portable Preload subject is beyond scope of this document and if you are interested in it, please refer to MPCS Automation Services website:

<http://msw.co.intel.com/AutomationServices/>

2.4 Dependencies

In order to function properly, VManager requires certain conditions to be met:

- Presence of .NET 2.0 or newer on a test system.
- VManager wrapper if one doesn't use Preload Environment.
- Network access to Chakotay server.
- Active Chakotay user account.



3 Writing Scripts For VManager

3.1 VManager Script Environment

VManager, located on test system requires certain environment conditions to be met in order to function properly: specific folders and registry entries.

Folders contain drivers, install applications, tools etc. Folders will be copied onto hard drive during system preload before VManager first launch. Their content will be updated with adding of new scripts to VManager.

Base registry entries are created by VManager itself during first execution. Windows Registry is a place where VManager creates and stores information necessary for running scripts on current test system.

3.1.1 VManager Folders

Folder	Content	Description
(1)C:\Apps	Install versions of applications	Default folder with subfolders where install files are stored. Script uses this folder to copy necessary application(s) onto the local drive from network location and install it in default location; Installed application will then be used by test script or user
C:\Bin	Applications, utilities	Default folder with VManager maintenance and miscellaneous files
C:\Drivers	Hardware components' drivers	Default folder with drivers required for install scripts
C:\Logs	TXT, LOG files	Default folder with log files monitoring VManager and script execution and progress
C:\Scripts	Scripts' executables	Default folder that contains all available scripts' executables and from where VManager executes a script chosen by user
C:\Software_Updates	Software patches, system updates, KBs files	Default folder for variety of software and system updates



		required by scripts
C:\VMLauncher	VManager system update file	Default folder containing application that updates VManager script environment

Where "C:\\" is actual **system drive**

Example 1:

"C:\Apps\CPU_Freq_Display" folder and "Intel(R) Frequency Display" INSTALL SCRIPT

This subfolder contains installation files for Intel(R) Frequency Display" tool. This folder is created during execution of "Intel(R) Frequency Display" INSTALL SCRIPT. Script copies necessary files to "C:\Apps\CPU_Freq_Display" and then executes them in order to install application on test system.

Example 2:

"C:\Bin\robocopy.exe"

VManager uses "robocopy.exe" to copy files and folders from server onto test system.

Example 3:

"C:\Drivers\Video\"

Subfolder contains video driver for test platform.

Example 4:

"C:\Logs\Manager.txt"

VManager log file.

3.1.2 VManager Registry Keys

Registry Key: "My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG"

Value	Type	Description
-------	------	-------------



PlatformFamily	REG_SZ	Platform family name based on ICH code name
PlatformName	REG_SZ	Platform name based on platform code name
PreloadServer	REG_SZ	Name of a server test platform was preloaded from
PreloadType	REG_SZ	Preload type (Network/DVD/Portable Preload)
VManagerCustomization	REG_SZ	Customization feature flag
VManagerRunCount	REG_DWORD	Number of VManager runs since fresh OS preload
VMLauncher	REG_SZ	VMLauncher run status flag

Registry Key: "My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP"

Value	Type	Description
Application Install Check	REG_SZ	Status flag that describes progress on 3 rd party application installation result during current test script execution
CleanUp	REG_SZ	Information flag for clean up script about current running script (related to CleanUpApp and Reboot keys)
CleanUpApp	REG_SZ	Flag that calls for clean up script (related to CleanUp and Reboot keys)
DebugTraceEnabled	REG_SZ	Are we going to remove this one?
IMgr	REG_SZ	Status check flag; checks if VManager has done running all of the chosen install scripts
InstallOnly	REG_SZ	Sets Test Script to Install Only mode (script copies all necessary files onto test platform without actual test execution)
Reboot	REG_SZ	"Reboot required" flag after running cleanup script (related to CleanUp and CleanUpApp keys)



Run Application Script	REG_SZ	Status flag that describes progress on 3 rd party application execution during current test script execution
RunNext	REG_SZ	Stores information about script that will be executed next
ScriptName	REG_SZ	Script progress flag for <i>ScriptName</i> script
TMgr	REG_SZ	Status check flag; checks if VManager has done running chosen test script
VManagerDirectoryUpdate	REG_SZ	Status on latest VManager directories update
VManagerVersion	REG_SZ	Current VManager version in use

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP\VMgrRebootFlag"

Value	Type	Description
ScriptName	REG_SZ	VManager reboot flag for current script being executed

Registry Key: "My
Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce"

Value	Type	Description
VManager	REG_SZ	Flag allows to automatically restart VManager after system reboot (quiet start – no UI); VManager restarts, finalizes script execution and displays summary window.

3.1.2.1 Test Script Temporary Keys

In addition to registry keys above there are two more created for **purpose of test script and ONLY for time of its execution.**



Scripter is advised to use this space to create KEYS and VALUES necessary to run test scripts.

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTSCRIPT"

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTSTATUS"

During next VManager run, clean-up script is executed to clean temporary registry entries.

3.2 Install Scripts

Install scripts make life around lab/ office easier. Users can choose application/ applications they need from VManager menu and install script will do rest of work for them. They don't need to know location of files or network connections. Install script automates, like the name says, installation process.

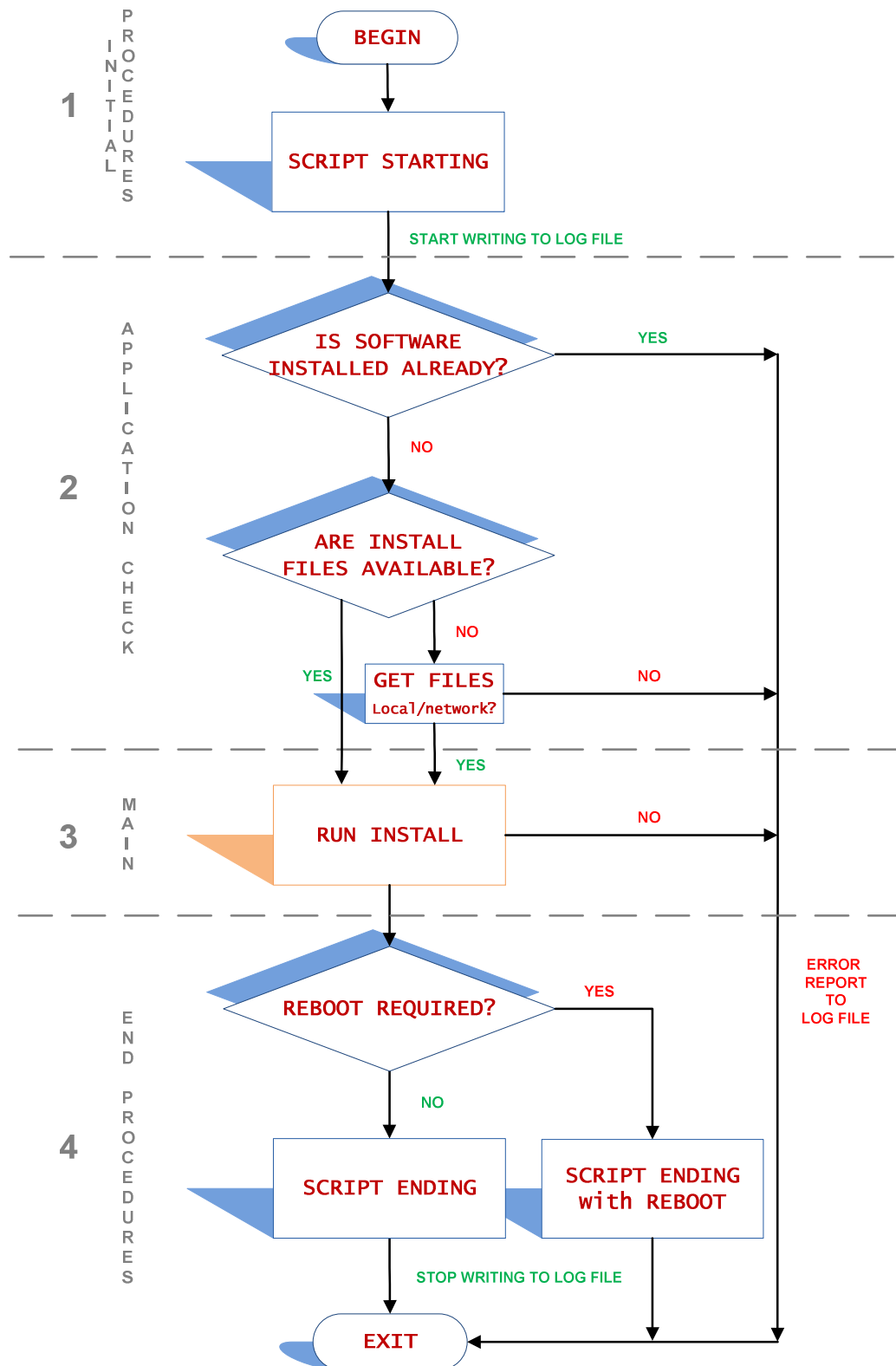
Install script is very similar to config script from logical point of view. Main difference is that there are few more conditions that have to be checked at starting point and nature of *main block* is different: it is an execution of install process of 3rd party software.

Install script is presented on flow chart below.

Once selected and executed, script starts with initial procedures. Next is the main part of the script, where scripter creates automated installation process for software application. Based on changes system can be forced to reboot. Finally end procedures like registry update and displaying summary screen can be applied.

It is recommended to send script progress information to Manager.txt log file located in "**C:\Logs**" folder.

3.2.1 Install Script Flow Chart





3.2.2 How Does Install Script Work?

This part describes logic blocks from flow chart above. Knowledge presented here will help scripter to understand how install script works and allow to create one.

There are two categories of logic blocks in install script:

BKMs (best known methods) block – parts of the script that are recommended by this document - they help to keep script organized.

Required blocks – these blocks are required by VManager. Their role is to directly interact with our VManager script environment.

This chapter will mention BKMs and explain each required block to help scripter with writing VManager compatible scripts.

Install script structure can be divided by 4 logical blocks:

Initial procedures

Application check

Main part

End procedures

Each block will be explained further in document.

3.2.3 INITIAL PROCEDURES Block

Script execution begins here.

These are recommended BKMs for this block:

start writing to log file - it can include script date and version, time when execution began etc.

display message window on screen with script information and progress

check script's time stamp

prepare network environment by unmapping all existing network drives, creating needed connections



3.2.4 APPLICATION CHECK Block

This block is introduced in Config Script Chapter. Since we are going to install application, we need to know if it's already installed on test system and if not, where to get files from.

These are recommended BKM's for this block:

check if software you are trying to install is already installed; the easiest way to do so is to check for presence of *.EXE or other significant file

If software is not installed you need to get install package; what's the location? local or network?

install files are usually located on server; this requires creating network connection

depending on install package size it is strongly advised to copy all the install files onto local HDD of test system before execution of main part of test script to speed up progress and reduce problems related to network connectivity

3.2.5 MAIN Block

This block represents main part of script. Scripter uses this space to create code that executes automated 3rd party application INSTALLATION.

It is usually a combination of simulated keystrokes, mouse movement and window/control manipulation.

NOTE: Some applications don't require installation. They work with files available in install package. This significantly simplifies or even eliminated main block execution.

This block introduces new value for registry key known from config script chapter.

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP

Value	Value Type	Value Data (possible values)	Description
Script Name	REG_SZ	RUNNING	Script progress flag for <i>Script Name</i> script. VManager monitors script execution based on this value.



This registry key is **REQUIRED**. It should be created at the beginning of this block's execution. Its value has to be changed in case of exception (for example: error getting files) which will result in script error.

3.2.6 END PROCEDURES Block

Script execution ends here. It is recommended in this block to include:

finish writing to log file (it can include script date and version, time when execution began etc.)

display window on the screen with message about finishing script execution

prepare network environment by unmapping all existing network drives

delete install files if necessary

Based on configuration needs, system reboot might be required to apply changes.

Install script requires exactly the same registry and script entries like config script.

IN CASE OF SYSTEM REBOOT

Two registry entries are **REQUIRED** for VManager script environment. They have to be created before **System Reboot** is forced:

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP

Value	Type	Data (possible values)	Description
Script Name	REG_SZ	ERROR ABORTED INSTALLED	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of <i>Script Name</i> and "Data" field at the end of script execution and will also finish running.

In addition **Data:** ERROR and ABORTED can contain supplementary information based on a reason of exception.



Example:

ABORTED: Script_Name is already installed.

ERROR: Script_Name – Can't locate install files.

My Computer\HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"

Value	Type	Data	Description
VManager	REG_SZ	System_drive:\Scripts\VMManager.exe - NOUI	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of <i>Script Name</i> and "Data" field at the end of script execution and will also finish running.

IN CASE WHEN SYSTEM REBOOT IS NOT NECESSARY

Scripter is **REQUIRED** to:

Create registry entry

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP

Value	Type	Data (possible values)	Description
Script Name	REG_SZ	ERROR ABORTED INSTALLED	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of <i>Script Name</i> and "Data" field at the end of script execution and will also finish running.

In addition **Data:** ERROR and ABORTED can contain additional information based on a reason of exception.

Make entry at the end of the script to call VManager:



```
$strVMgr = @HomeDrive & "\scripts\VMManager.exe"  
Run(@ComSpec & " /c " & $strVMgr & " -NOUI", "", @SW_HIDE)
```

This AutoIT snippet of code is example of quiet command line execution of VManager located here:

```
System_drive:\Scripts\VMManager.exe -NOUI
```

Required registry keys and script entry presented above come to play when VManager executes multiple scripts. Scripter should monitor script execution and in case of any problems register appropriate **Data type** for **Script Name value**.

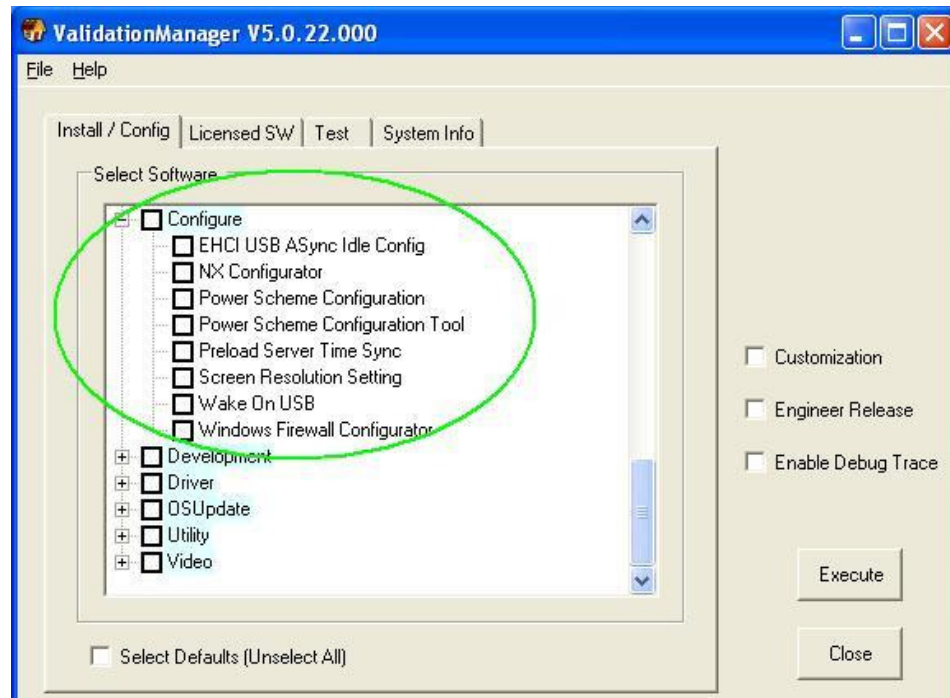
If one of a queued config scripts executes with error, VManager based on script status moves forward to another script.

3.3 Config Scripts

Config script is the "easiest to write" script available in VManager. Config script allows user to change system configuration in clean and straightforward way without need of knowing folder structure, specific features or system registry.

For config script we can distinguish main blocks presented on flow chart below.

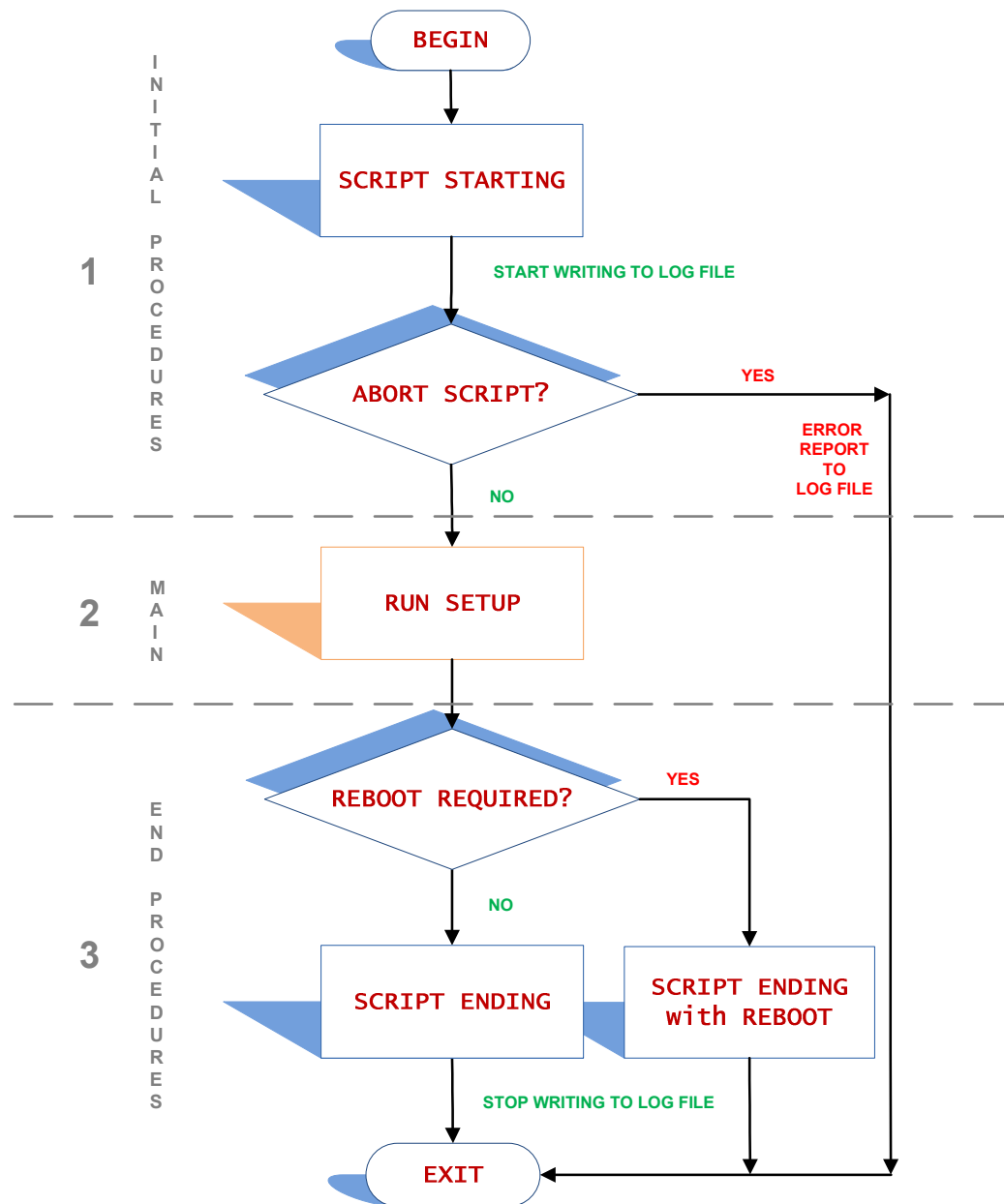
Execution begins with selecting script from "VManager Configure" menu:



Once selected and executed, script starts with initial procedures. Next is the main part of script, where scripiter creates all configuration changes that have to be applied to test system. Based on changes requirement system can be forced to reboot and finally end procedures like registry update and displaying summary screen can be applied.

It is recommended to send script progress information to Manager.txt log file located in "C:\Logs\" folder.

3.3.1 Config Script Flow Chart





3.3.2 How Does Config Script Work?

This part describes logic blocks from flow chart presented above. Knowledge presented here will help scripter to understand how config scripts work and let him/her to create one.

Flow diagram represents two categories of logic blocks in config script:

BKMs (best known methods) block – parts of the script that are recommended by this document – they help to keep script organized

Required blocks – these blocks are **required** by VManager. Their role is to directly interact with our VManager script environment.

This chapter will mention BKMs and explain **each required** block to help scripter with writing VManager compatible scripts.

Config script is divided by three parts:

Start procedures

Main part

End procedures

Each block will be explained below.

3.3.3 INITIAL PROCEDURES Block

Script execution begins here.

It is recommended in this block to include things like:

start writing to log file (it can include script date and version, time when execution began etc.)

display message window on screen with script information and progress

check script's time stamp

prepare network environment by unmapping all existing network drives, creating needed connections

Another feature worth mentioning here is capability of **script aborting**. In case of previous script execution and changes already made, script can check that and abort unnecessary script execution to save time.



3.3.4 MAIN Block

This block represents main part of script. Scripter uses this space to create code that executes CONFIGURATION. This is where script GUI is called, where user can make changes that will affect for example test system, 3rd party software etc.

Example:

VManager config script "EHCI USB ASync Idle Config"

Uses Hotfix Registry Key for Windows XP SP3 (KB918005); script checks and changes value of REG_SZ word in registry to either Enable or Disable USB ASync Idle feature.

3.3.5 END PROCEDURES Block

Script execution ends here. It is recommended in this block to include things like:

finish writing to log file (it can include script date and version, time when execution began etc.)

display window on the screen with message about finishing script execution

prepare network environment by unmapping all existing network drives

Based on configuration needs, system reboot might be required to apply changes.

IN CASE OF SYSTEM REBOOT

In addition, two registry entries are **REQUIRED** for VManager script environment. They have to be created before **System Reboot** is forced:

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP

Value	Type	Data (possible values)	Description
Script Name	REG_SZ	ERROR ABORTED INSTALLED	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of <i>Script Name</i> and "Data" field at the end of script execution and will also finish running.



In addition **Data:** ERROR and ABORTED can contain additional information based on a reason of exception.

Example:

ERROR: Script_Name – main block execution error.

ABORTED: Script_Name – configuration already changed.

My Computer\HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"

Value	Type	Data	Description
VManager	REG_SZ	System_drive:\Scripts\VMManager.exe - NOUI	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of <i>Script Name</i> and "Data" field at the end of script execution and will also finish running.

IN CASE WHEN SYSTEM REBOOT IS NOT NECESSARY

If System Reboot is not necessary scripiter is **REQUIRED** to:

Create registry entry

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP

Value	Type	Data (possible values)	Description
Script Name	REG_SZ	ERROR ABORTED INSTALLED	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of <i>Script Name</i> and "Data" field at the end of script execution and will also finish running.

Make entry at the end of the script to call VManager:



```
$strVMgr = @HomeDrive & "\\scripts\VMManager.exe"  
Run(@ComSpec & " /c " & $strVMgr & " -NOUI", "", @SW_HIDE)
```

This AutoIT snippet of code is example of quiet command line execution of VManager located here:

```
System_drive:\Scripts\VMManager.exe -NOUI
```

Required registry keys and script entry presented above come to play when VManager executes multiple scripts. Scripter should monitor script execution and in case of any problems register appropriate **Data type** for **Script Name value**.

If one of a queued config scripts executes with error, VManager based on script status moves forward to another script.

3.4 Test Scripts

The main purpose of test script is to automate and execute needed tests common for platform and software validation. Depending on requirements it can include installation, execution and automation of 3rd party software.

Test script is the most complex script available in VManager environment.

The main focus in test script is on application install and execution. This might require multiple system reboots during tests script execution and script capability to pick up its execution after mentioned reboots. This is where registry entries come to play.

Test script is presented on flow chart below.

Once selected and executed, script starts with initial procedures. Next step is Applications Install Check where scripts check if required applications are already installed on test platform. If not, test script executes either install script or built-in applications install function. This step is repeated until all required applications are required.

Next step (Run Later block), allows to copy and install all required software necessary for script execution. If selected, script stops its execution here and is available for later execution. Test platform can now be disconnected from network etc... and test script is still available for later execution on test platform. In this step CleanUp Script is being set up for later execution in order to terminate still running applications, clean up temporary files and etc...

If execution continues, script enters main part where all previously installed applications have to be executed. Test script, based on registry entries, checks current application status and moves to next application execution.

Once all required applications are running, test script moves to actual test execution. Depending on needs, each application may require steps to be executed. In this part test script continuously checks if all test parts are executed.

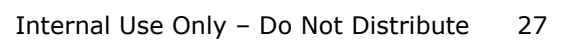


Finally, test script ends with execution of VManager and summary window.

At this point test script can still be executed due to its nature.

It is recommended to send script progress information to Manager.txt log file located in "**C:\Logs**" folder.

3.4.1 Test Script Flow Chart





3.4.2 How Does Test Script Work?

This part describes logic blocks from flow chart above. Knowledge presented here will help scripter to understand how test script works and allow to create one.

There are two main categories of information in install script:

BKMs (best known methods) activities – parts of the script that are recommended by this document - they help to keep script organized.

Required activities – parts that are required by VManager. Their role is to directly interact with our VManager script environment.

This chapter will mention BKMs and explain each **Requirement** to help scripter with writing VManager compatible scripts.

Test script structure can be divided by 6 logical blocks:

Initial Procedures

Application Check

Run Later

Application Execution

Test Execution

End Procedures

Each block will be explained further in document.

3.4.3 INITIAL PROCEDURES Block

Script execution begins here.

These are recommended BKMs for this block:

start writing to log file - it can include: script date and version, time when execution began etc.

display message window on screen with script information and progress

check script's time stamp

prepare network environment by unmapping all existing network drives, creating needed connections

Also for purpose of test script, VManager creates two registry keys that can be used by scripter to control test script flow and execution. Registry keys are:



For 32bit OSes location is:

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTSCRIPT"

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTSTATUS"

For 64Bit OSes location is:

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intel\MPG\TESTSCRIPT"

Registry Key: "My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\
Wow6432Node\Intel\MPG\TESTSTATUS"

Both keys are described closer in Chapter 2.1: *VManager Script Environment*.

These two registry keys are significant for test script execution. Scripter, if decides, can use this space to create and use own registry values to control flow of test script.

3.4.4 APPLICATION CHECK Block

This part looks different comparing to config and install scripts. For test script, this is where script has to install required applications that will be used in blocks 4 and 5 of flow chart. This means that install script, described in **Chapter 2.2**, can be a part of this block as long as scripter decides to use it instead of creating install functions built-in into test script. This block needs to have a capability of checking if required application is already installed or not. Reboot might be necessary depending on installed application requirements. Reboot restarts script and it is necessary to check for actual script progress to continue execution.

These are recommended BKM's for this block:

make sure you have needed install script executables available if you decide to use previously created scripts; if not you need to create required application install functions in your test script.

check if software you are trying to install is already installed; the easiest way to do it is to check for presence of *.EXE or other significant file(refer to *Application Install Check block in Install script chapter for more info*)

remember that you need to get install files and check for installed software (refer to *Application Install Check block in Install script chapter for more info*)

install files are usually located on server; this requires creating network connection (refer to *Application Install Check block in Install script chapter for more info*)



reboot might be required. Script needs to know how to continue execution afterwards. It can be achieved using registry keys (refer to *Application Install Check block in Install script chapter for more info*)

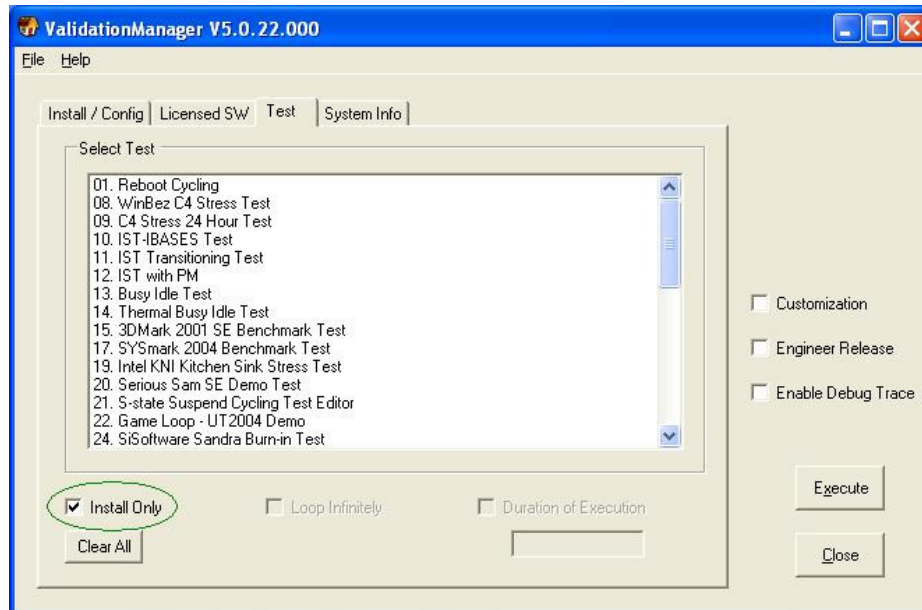
it's a good idea to keep your registry updated. You should create registry values in pointed registry keys with script and applications current status. This could be part of check that will be made at the beginning of Application Execution block (point **1.2.4**).

Example: creating temporary keys (for lifetime of test script) with information that required applications are installed can give a "green light" for Application Execution block described in point **1.2.4**.

3.4.5 RUN LATER Block

"Install Only" is an essential part of our VManager Script environment. It allows postponing test script execution for later.

The idea behind is to end execution of test script after environment is prepared by installing all necessary 3rd party applications (Application Check block) and before execution of test part of the script (block 4 and 5). Test script now can be executed on the same test platform in future when for example there is no network access to get needed applications.



There are two registry key value data options for this feature.

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP for 32Bit OSes



or

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intel\MPG\TESTAPP
for 64Bit OSes

Value	Type	Data (possible values)	Description
InstallOnly	REG_SZ	YES / NO	Sets Test Script to Install Only mode (script copies all necessary files onto test platform without actual test execution)

By selecting "Install Only" registry value is set to "YES". Scripter is **REQUIRED** to create mechanism that will check for this feature in registry and according to flag one will either stop execution of test script (data value = YES) after all applications are installed or continue script execution (data value = NO).

At this moment script execution is moving to the last block from flow diagram "End Procedures" described in point 1.2.6 of this chapter to run end procedures and terminate test script.

3.4.6 APPLICATION EXECUTION Block

Execution of this block follows application check block. Based on temporary registry keys' values (created by scripter) script knows that required applications are installed and can start their execution. This block prepares for actual test run in next step (test execution block of flow chart).

Also, at this point scripter is required to set up clean up script in order to clean temporary files and still running processes. Clean up script will be executed next time user executes VManager. For instructions how to setup clean up script and for related registry values please refer to point **1.3 CleanUp Script** at the end of this chapter

These are recommended BKM's for this block:

use registry keys' values to control application execution and status flow

in case of more than one application, script has to execute one application after another. It's good to create mechanism that checks if any of required applications are already running

it is a good idea to keep your registry updated. You should create registry values in pointed registry keys describing current status of test script and 3rd party applications. This could be part of check that will be made at the beginning of Test Execution block (point 1.2.5).



3.4.7 TEST EXECUTION Block

This block is a heart of test script. Scripter uses this space to create code that executes automation of 3rd party application(s).

3rd party application usually features test, benchmark etc... that is executed by test script using on combination of simulated keystrokes, mouse movements and window/control manipulation.

These are recommended BKM's for this block:

- execution of all tests
- using 2 designated registry keys to store all the test execution progress information

This block requires registry key known from config and install script chapters.

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP for 32Bit OSes
or

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intel\MPG\TESTAPP for 64Bit OSes

Value	Type	Data (possible values)	Description
Script Name	REG_SZ	RUNNING	Script progress flag for <i>Script Name</i> script. VManager monitors script execution based on this value.

This registry key is **REQUIRED**. It should be created at the beginning of this block's execution. Its value has to be changed in case of exception (for example: error getting files) which will result in script error.

3.4.8 END PROCEDURES Block

Script execution ends here. It is recommended in this block to include:

- finish writing to log file only about script execution. Some scripts based on their nature require to continue logging (example: loop-based test scripts)

- display window on the screen with message about finishing script execution and reason



prepare network environment by unmapping all existing network drives

continue to monitor running tests

Scripter is **REQUIRED** to create registry entry to summarize VManager Execution:

Create registry entry

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP for 32Bit OSes

or

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intel\MPG\TESTAPP
for 64Bit OSes

Value Name	Type	Value Data (possible values)	Description
Script Name	REG_SZ	ERROR ABORTED INSTALLED	Script progress flag for <i>Script Name</i> script. VManager will display message consisting of this <i>Script Name</i> and "Value Data" field at the end of script execution and will also finish running.

In addition **Value Data:** ERROR and ABORTED can contain additional information based on a reason of exception.

Example:

"ERROR: File not found"

"ABORTED: Application already installed"

Make code entry at the end of the test script to call VManager:

```
$strVMgr = @HomeDrive & "\scripts\VManager.exe"
```

```
Run(@ComSpec & " /c " & $strVManager & " -NOUI", "", @SW_HIDE)
```

This AutoIT snippet of code is an example of quiet command line execution of VManager located here:



System_drive:\Scripts\VManager.exe -NOUI

Required registry key and script entry presented above finalize execution of VManager.

Because of the nature of test script, only one test script can be executed at a time.

Scripter should monitor script execution and in case of any problems register appropriate '**Value Data**' value for '**Script Name**' value name.

3.4.9 Clean-up Script

Because of the nature of test script and its interaction with additional programs it is necessary to create CLEAN-UP script that will remove all temporary content that is a result of their execution.

It is **REQUIRED** to terminate execution of all 3rd party applications and test script before running VManager again. This can be implemented in CLEAN-UP script.

In addition things like:

- Temporary files
- Running processes
- Scheduled tasks

should also be removed.

Keep in mind that 2 test scripts cannot be queued in VManager!

VManager will automatically erase these two registry keys with all their content during next run

For 32Bit OSes

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTSCRIPT"
Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTSTATUS"

And respectively for 64Bit OSes:

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node
\Intel\MPG\TESTSCRIPT"
Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node
\Intel\MPG\TESTSTATUS"

REQUIRED:



During next VManager run, VManager will call clean-up script if you set **CleanUpApp** registry value located in this registry key. Two other values: **CleanUp** and **Reboot** are mentioned here as BKMs and can be used for clean-up script purpose. If you decide to use CleanUp script with registry keys mentioned below you need to setup **Value Names** listed below in **End Procedures Block** of test script

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intel\MPG\TESTAPP" (32Bit OS)

Registry Key: "My
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intel\MPG\TESTAPP"
(64Bit OS)

Value Name	Type	Value Data	Description
CleanUp	REG_SZ	Name of test script to clean up after Example: 99. Benchmark Test	Information flag for clean up script about current running test script. Based on name of the test script clean-up script can call termination of running script, scheduled tasks and applications, clean-up temporary files, etc.
CleanUpApp	REG_SZ	Path and name of clean-up script Example: C:\Scripts\CleanUpScript.exe	Flag that calls particular clean-up script. Scripter can choose to create one clean-up script with array of test scripts to clean or single clean-up script for each test script created.
Reboot	REG_SZ	1 or 0 1 – reboot 0 – no reboot	"Reboot required" flag after running cleanup script; depending on need flag is set either to 0 or 1 (no reboot/reboot needed)

After clean-up procedures, clean up script should have capability of calling VManager in order to display VManager main window.

Now, all temporary files are deleted and previously running processes terminated. VManager can start normal functionality and scripts are ready for execution.

3.5 Script Deployment

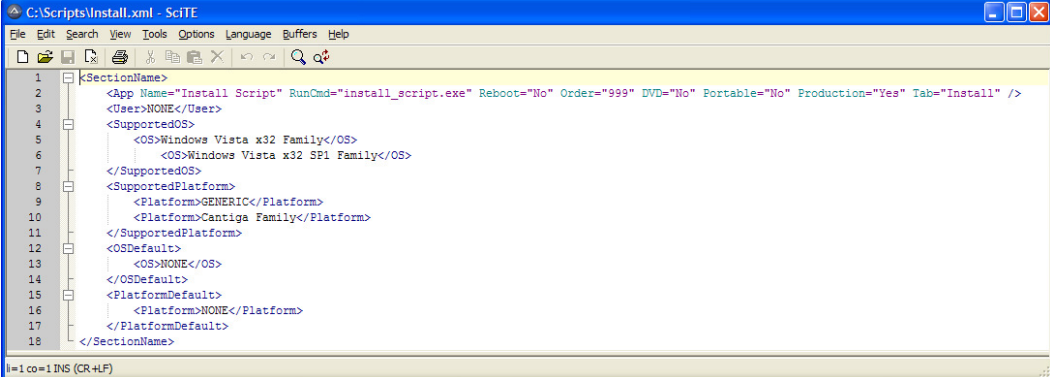
This chapter consists of two parts: first is about how to write and update XML files that store script configuration information for VManager, second is about forms of script deployment: customization and *Andre's doc*.

3.5.1 XML File How To

3.5.1.1 XML files and VManager

VManager uses the XML files to determine how to display the scripts in the user interface and how to execute the scripts. There are two kinds of XML files: one that gathers information about install and config scripts and second with test scripts' information.

During execution, VManager will look for either of these XML files in the location specified (system drive:\scripts) and display the information. The images below are examples of these XML files that represent single script entry for each XML file. To add more than one script to VManager, simply create multiple entries in the XML files.

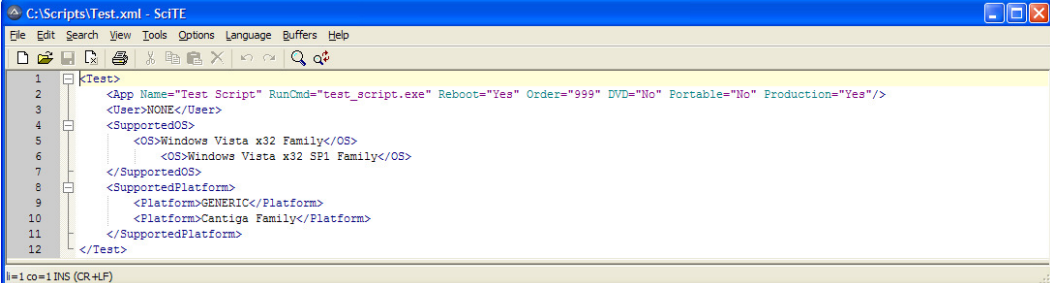


```

1 <SectionName>
2 <App Name="Install Script" RunCmd="install_script.exe" Reboot="No" Order="999" DVD="No" Portable="No" Production="Yes" Tab="Install" />
3 <User>NONE</User>
4 <SupportedOS>
5 <OS>Windows Vista x32 Family</OS>
6 <OS>Windows Vista x32 SP1 Family</OS>
7 </SupportedOS>
8 <SupportedPlatform>
9 <Platform>GENERIC</Platform>
10 <Platform>Cantiga Family</Platform>
11 </SupportedPlatform>
12 <OSDefault>
13 <OS>NONE</OS>
14 </OSDefault>
15 <PlatformDefault>
16 <Platform>NONE</Platform>
17 </PlatformDefault>
18 </SectionName>

```

Above: XML file format for install and config scripts – single entry



```

1 <Test>
2 <App Name="Test Script" RunCmd="test_script.exe" Reboot="Yes" Order="999" DVD="No" Portable="No" Production="Yes"/>
3 <User>NONE</User>
4 <SupportedOS>
5 <OS>Windows Vista x32 Family</OS>
6 <OS>Windows Vista x32 SP1 Family</OS>
7 </SupportedOS>
8 <SupportedPlatform>
9 <Platform>GENERIC</Platform>
10 <Platform>Cantiga Family</Platform>
11 </SupportedPlatform>
12 </Test>

```

Above: XML file format for test scripts – single entry



3.5.1.2 Defining a Script Entry in a XML file

VManager uses XML files to define what, when, and how to show and execute scripts.

3.5.1.3 Example of a Install/Config Script Entry

Below is an example of what a script entry in the xml install/config would look like for adding a script called "Install Script". This script will be displayed on the Install / Config tab of the VManager in a section of the tab called SectionName.

```
<SectionName>

<App Name="Install Script" RunCmd="install_script.exe" Reboot="No" Order="999"
DVD="No" Portable="No" Production="Yes" Tab="Install" />

<User>NONE</User>

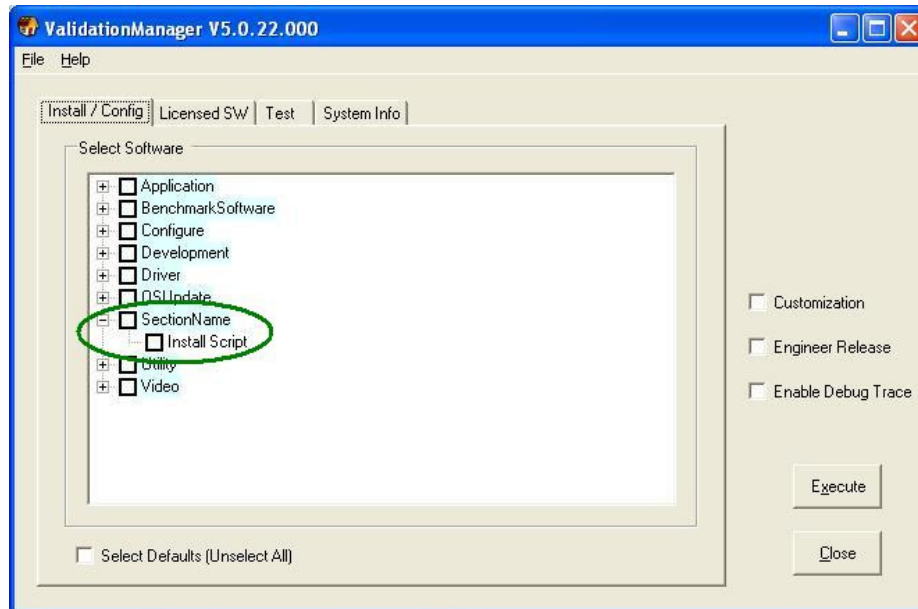
  <SupportedOS>
    <OS>Windows Vista x32 Family</OS>
    <OS>Windows Vista x32 SP1 Family</OS>
  </SupportedOS>

  <SupportedPlatform>
    <Platform>Generic</Platform>
    <Platform>Cantiga Family</Platform>
  </SupportedPlatform>

  <OSDefault>
    <OS>NONE</OS>
  </OSDefault>

  <PlatformDefault>
    <Platform>NONE</Platform>
  </PlatformDefault>

</SectionName>
```



3.5.2 Defining the Section to be Displayed in Install/Config Tab

You can create your own Install / Config Tab section tree based on script single entries presented above. Please note that the section name cannot include any spaces or special characters.

As an example these are currently existing sections in the Install / Config Tab:

- Application
- BenchmarkSoftware
- Configure
- Development
- Driver
- OSUpdate
- Utility
- Video

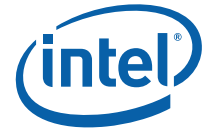
3.5.2.1 Defining App and its Attributes

The App syntax contains most of the key information used to display and execute the script.

Example of App syntax:

```
<App Name="Install Script" RunCmd="install_script.exe" Reboot="No" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" Tab="Install" />
```

Attribute	Description	Input	Default
-----------	-------------	-------	---------



			Value
Name=	Defines the friendly name of the script which will be displayed in VManager for users to select. Note: This should be a unique name not used by any other script	string	
RunCmd=	The executable name of the script file. Note: This needs to be a unique name not used by any other script	string	
Reboot=	Defines to VManager if the script will be rebooting the system during or after its execution "YES" – script will reboot the system "NO" – script will not reboot system	YES or NO	NO
Order=	Defines the order in which the scripts will be executed. The smaller the number the sooner it will run. It is recommended that the default value to be used unless there is a critical reason to increase the order of execution. If there are multiple scripts with the same value they will execute in the order they are listed in the Master XML.	Integer between 1-9999	999
DVD=	This attribute must be defined with the default value	YES	YES
Portable=	This attribute must be defined with the default value	YES	YES
Production=	This attribute must be defined with the default value	YES	YES
Tab=	This attribute must be defined with the default value	INSTALL	INSTALL

3.5.2.2 Defining User Required Feature

The <User> syntax defines if particular script requires user input or not. Based on value of this element a script will either be or not be displayed in VMLite. This element is only for VMLite purposes and doesn't affect VManager functionality.

<User>[value]</User>

Value can have one of two values: YES / NONE.

YES – user input required in script execution; script not visible in VMLite

NO – user input is not required in script execution; script visible in VMLite



3.5.2.3 Defining Supported Operating Systems

The <SupportedOS> syntax defines which Windows operating systems are supported by a script. This can be defined as all Windows operating systems, or it can be one or more families of Windows operating systems.

Supported OS entries which VManager will recognize within the XML file(s):

- ALL
- Windows 7 Family
- Windows 7 x64 Family
- Windows 7 x32 Family
- Windows Vista Family
- Windows Vista x64 Family
- Windows Vista x32 Family
- Windows Vista SP1 Family
- Windows Vista x64 SP1 Family
- Windows Vista x32 SP1 Family
- Windows XP Family
- Windows XP SP3 Family
- Windows XP Professional SP3 Family
- Windows XP Professional SP3
- Windows XP Media Center 2005 SP3
- Windows XP Tablet PC 2005 SP3
- Windows XP Home Edition SP3
- Windows XP SP2 Family
- Windows XP Professional SP2 Family
- Windows XP Professional SP2
- Windows XP Media Center 2005
- Windows XP Tablet PC 2005
- Windows XP Home Edition SP2
- Windows XP Professional SP1
- Windows XP x64 Family
- Windows XP x64 SP1
- Windows XP x64 SP2

Example 1:

```
<SupportedOS>  
  
<OS>Windows Vista x32 Family</OS>  
  
<OS>Windows Vista x32 SP1 Family</OS>  
  
</SupportedOS>
```

A script which defines its supported OS's as above will only be displayed in the 32 bit versions of the Vista operating systems that contain no service pack and in 32 bit versions of Vista operating systems with service pack 1.

Example 2:

```
<SupportedOS>  
  
<OS>Windows XP Family</OS>
```




```
<OS>Windows 7 Family</OS>
```

```
</SupportedOS>
```

A script which defines its supported OS's as above will only be displayed in all 32 bit Windows XP operating systems and all Windows 7 operating systems.

Example 3:

```
<SupportedOS>
```

```
<OS>ALL</OS>
```

```
</SupportedOS>
```

A script which defines its supported OS's as above will be displayed in all Windows operating systems.

3.5.2.4 Defining Supported Platforms

The <SupportedPlatform> syntax defines the hardware platforms that are supported by a script. This can be defined as all platforms, or one or more platform families.

Supported Platform entries that VManager will recognize within the XML file(s):

Entry	Description
ALL	Displays scripts for all platforms (specified and generic)
GENERIC	Displays scripts for platforms that are not recognized
Crestline Family	Displays scripts for platforms which are recognized as part of the Mobile Santa Rosa Program (Crestline / ICH8-M)
Cantiga Family	Displays scripts for platforms which are recognized as part of the Mobile Montevina Program (Cantiga / ICH9-M).
IbexPeak Family	Displays scripts for platforms which are recognized as part of the Mobile Calpella Program (IbexPeak)
SandyBridge-CPT Family	Displays scripts for platforms which are recognized as part of the Mobile HuronRiver Program (SandyBridge)

Example 1:

```
<SupportedPlatform>
```

```
<Platform>Cantiga Family</Platform>
```

```
</SupportedPlatform>
```



A script that defines the supported Platform as above will only be displayed for Montevina platform CRB's (for example Silver Cascade).

Example 2:

```
<SupportedPlatform>
<Platform>Generic</Platform>
<Platform>IbexPeak Family</Platform>
</SupportedPlatform>
```

A script that defines the supported Platforms as above will only be displayed for platforms that are identified as part of the IbexPeak Family (ie Red Fort CRB) and on platforms that were not recognized, which are considered generic platforms.

Example 3:

```
<SupportedPlatform>
<Platform>ALL</Platform>
</SupportedPlatform>
```

A script that defines the supported Platforms as above will be displayed on all Platforms.

3.5.2.5 Defining VManager Default Scripts

VManager has a "Select Defaults" feature that is used to automatically pre-select the scripts located in the Install / Config tab that have been identified as default scripts. The very first time VManager is run it will enable this feature. For all other VManager launches nothing will be automatically pre-selected, but at anytime the "Select Defaults" feature can be enabled by the user from the VManager interface. The definition of whether a script is a default is contained in the XML install file and is based on the values defined for the <OSDefault> and <PlatformDefault> syntax.

The possible entries for <OSDefault> or <PlatformDefault> are the same as the supported options for <SupportedOS> and <SupportedPlatform>. There is one additional value of "NONE" which is used to denote that the script is not to be selected as a default.

VManager will evaluate the contents of <OSDefault> and <PlatformDefault> and when both conditions are true it will select the script as part of the "Select Defaults" feature. If either one of these entries contains a value of "NONE" the script will not be selected as a default.

Example 1:

```
<OSDefault>
```



```
<OS>Windows XP Professional SP3</OS>
```

```
</OSDefault>
```

```
<PlatformDefault>
```

```
<Platform> IbexPeak Family</Platform>
```

```
</PlatformDefault>
```

A script that defines the default OS and Platform as above will only be selected as a default script for Windows XP Professional SP3 on IbexPeak Family Platforms.

Example 2:

```
<OSDefault>
```

```
<OS>NONE</OS>
```

```
</OSDefault>
```

```
<PlatformDefault>
```

```
<Platform>IbexPeak Family</Platform>
```

```
</PlatformDefault>
```

A script that defines the default OS and Platform as above will not be selected as a default script because the default OS has been defined as NONE.

Example 3:

```
<OSDefault>
```

```
<OS>NONE</OS>
```

```
</OSDefault>
```

```
<PlatformDefault>
```

```
<Platform>NONE</Platform>
```

```
</PlatformDefault>
```

A script that defines the default OS and Platform as above will not be selected as a default script because both values have been set to NONE.

3.5.2.6 Example of a Test Script Entry

Below is an example of what a script entry in the XML test file would look like for adding a script called "Test Script". This script will be displayed on the test tab.



<Test>

```
<App Name="Test Script" RunCmd="test_script.exe" Reboot="No" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" />
```

<User>NONE</User>

<SupportedOS>

<OS>Windows Vista x32 Family</OS>

<OS>Windows Vista x32 SP1 Family</OS>

</SupportedOS>

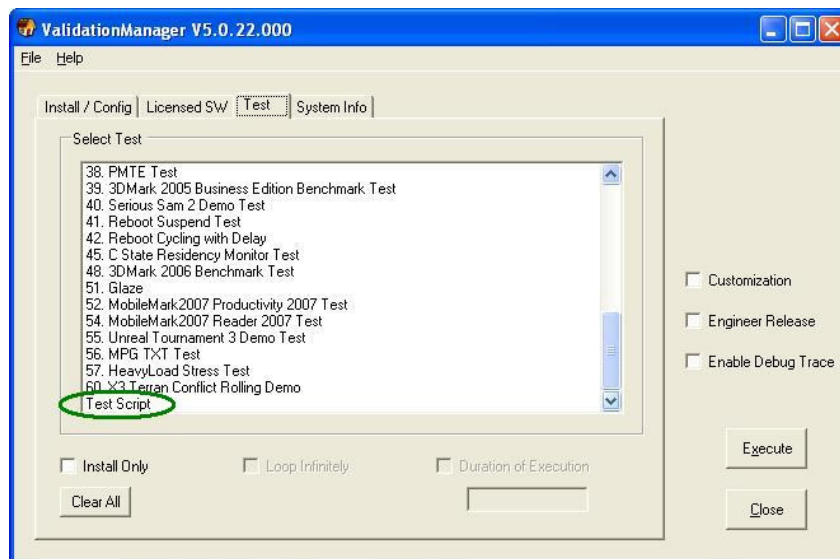
<SupportedPlatform>

<Platform>GENERIC</Platform>

<Platform>Cantiga Family</Platform>

</SupportedPlatform>

</Test>





3.5.2.7 Differences between Install/Config and Test XML Entries

An XML entry for an install/config script is almost identical to the entries for test scripts. There are a few differences which should be noted.

Differences between the entries:

Test scripts don't have sections, each test is defined with the following syntax:

<Test>

...

</Test>

The App syntax is the same up until the final attribute. Tab= is not included for test scripts. Example:

```
<App Name="Custom Script 1" RunCmd="script1.exe" Reboot="No" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" />
```

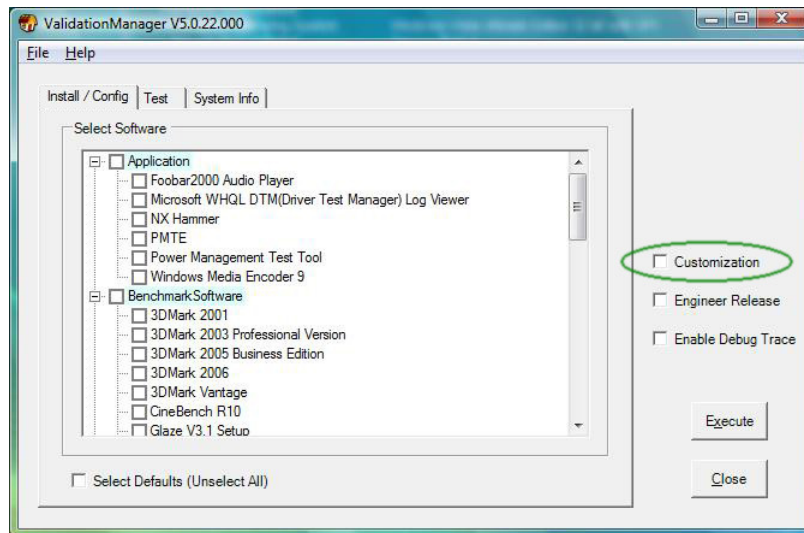
The test tab does not have any default selections, so <OSDefault> and <PlatformDefault> should not be defined for test scripts.

4 Advanced Features

4.1 VManager Customization Feature

4.1.1 Add Custom Scripts to VManager

VManager contains a feature called "Customization" which allows users to add their own scripts to VManager. When this feature is selected the user will be prompted to provide the location of the script to be added. This location must also contain the custom XML files which contain configuration information for the scripts that are to be added. VManager uses the XML files to determine how to display the scripts in the user interface and how to execute the scripts. The information contained in the custom XML files provided by the user will be appended to the Master XML files and the custom script files will be copied to the local hard drive. The necessary requirements and procedures for adding custom scripts to VManager will be outlined in this chapter.



4.1.2 Customization Feature Requirements

For VManager to display and execute custom scripts there are three key requirements that must be followed:

- A custom XML file which contains formatted entries for all scripts to be added to VManager. The custom XML files have fixed file names that must be used:
 - InstallApp.xml - adds scripts to the install/config tab of VManager.
 - TestApp.xml - adds scripts to the test tab of VManager.
- Custom scripts in an executable file format.
- The custom XML files must reside in the same directory as the custom script files.



4.1.3 Creating XML Files to Support Custom Scripts

There are two different XML files which the customization feature will recognize and use. Only the XML files appropriate for the type of scripts that are to be added to VManager needs to be provided.

InstallApp.xml – contains information about install and/or configuration scripts to be added to the Config/Install tab of VManager.

TestApp.xml – contains information about test scripts to be added to the Test tab of VManager.

During the customization process, VManager will look for either of these XML files in the location specified and append the information inside of them to the Master XML files. The images below are examples of these XML files formatted to add one script each. To add more than one script to VManager, simply create multiple entries in the custom XML files.

```

1 <CustomSection>
2 <App Name="Custom Script 1" RunCmd="script1.exe" Reboot="No" Order="999" DVD="No" Portable="Yes" Production="Yes" Tab="Install" />
3 <User>NONE</User>
4 <SupportedOS>
5 <OS>Windows Vista x32 Family</OS>
6 <OS>Windows Vista x32 SP1 Family</OS>
7 </SupportedOS>
8 <SupportedPlatform>
9 <Platform>GENERIC</Platform>
10 <Platform>Cantiga Family</Platform>
11 </SupportedPlatform>
12 <OSDefault>
13 <OS>NONE</OS>
14 </OSDefault>
15 <PlatformDefault>
16 <Platform>NONE</Platform>
17 </PlatformDefault>
18 </CustomSection>
19
20

```

```

1 <Test>
2 <App Name="Custom Script 1" RunCmd="script1.exe" Reboot="Yes" Order="999" DVD="No" Portable="No" Production="Yes"/>
3 <User>NONE</User>
4 <SupportedOS>
5 <OS>Windows Vista x32 Family</OS>
6 <OS>Windows Vista x32 SP1 Family</OS>
7 </SupportedOS>
8 <SupportedPlatform>
9 <Platform>GENERIC</Platform>
10 <Platform>Cantiga Family</Platform>
11 </SupportedPlatform>
12 </Test>

```

4.1.3.1 Defining a Script Entry in a Custom XML

VManager uses XML files to define what, when, and how to show and execute scripts. The customization feature in VManager appends the script entries in the custom XML files to its Master XML files. It is important that the script entries are created correctly for them to be properly appended.

4.1.3.1.1 Example of a Install/Config Script Entry

Below is an example of what a script entry in the InstallApp.xml would look like for adding a script called "Custom Script 1". It is possible to have the scripts displayed in a user specified section or in an existing section. This script will be displayed on the Install / Config tab of the VManager in a section of the tab called CustomSection.

```
<CustomSection>

<App Name="Custom Script 1" RunCmd="script1.exe" Reboot="Yes" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" Tab="Install" />

<User>NONE</User>

<SupportedOS>

<OS>Windows Vista x32 Family</OS>

<OS>Windows Vista x32 SP1 Family</OS>

</SupportedOS>

<SupportedPlatform>

<Platform>GENERIC</Platform>

<Platform>Cantiga Family</Platform>

</SupportedPlatform>

<OSDefault>

<OS>NONE</OS>

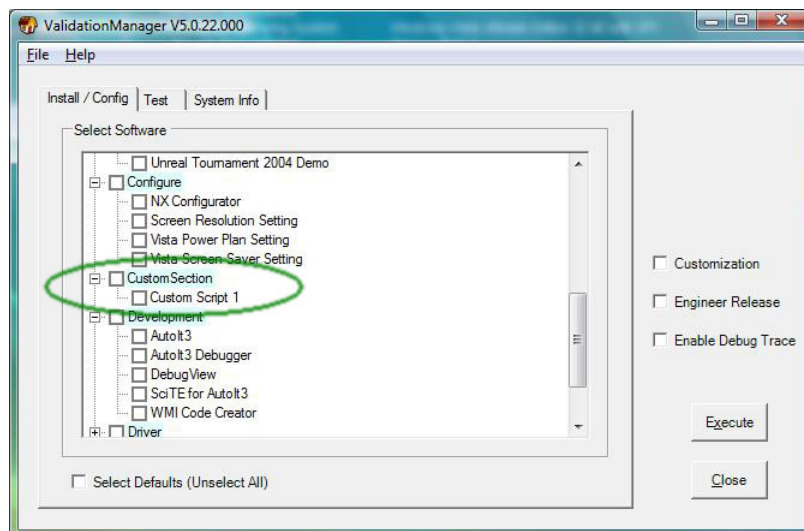
</OSDefault>

<PlatformDefault>

<Platform>NONE</Platform>

</PlatformDefault>

</CustomSection>
```

4.1.3.1.2 Defining the Section to be Displayed in Install/Config Tab

Install and Config scripts can be displayed in existing sections in VManager or in one specified in the InstallApp.xml. The section is the first item to define when creating an entry in the InstallApp.xml file. Please note that the section name cannot include any spaces or special characters.

Existing sections in the Install / Config Tab:

- Application
- BenchmarkSoftware
- Configure
- Development
- Driver
- OSUpdate
- Utility
- Video

For example to have a section called CustomSection to list the custom scripts on the Install/Config tab, the script definition should be enclosed with the section keywords:

```
<CustomSection>
```

```
...
```

```
</CustomSection>
```

4.1.3.1.3 Defining App and its Attributes

The App syntax contains most of the key information used to display and execute the script.

Example of App syntax:

```
<App Name="Custom Script 1" RunCmd="script1.exe" Reboot="No" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" Tab="Install" />
```

Attribute	Description	Input	Default Value
Name=	Defines the friendly name of the script which will be displayed in VManager for users to select. Note: This should be a unique name not used by any other script	string	
RunCmd=	The executable name of the script file. Note: This needs to be a unique name not used by any other script	string	
Reboot=	Defines to VManager if the script will be rebooting the system during or after its execution "YES" – script will reboot the system "NO" – script will not reboot system	YES or NO	NO
Order=	Defines the order in which the scripts will be executed. The smaller the number the sooner it will run. It is recommended that the default value be used unless there is a critical reason to increase the order of execution. If there are multiple scripts with the same value they will execute in the order they are listed in the Master XML.	Integer between 1-9999	999
DVD=	This attribute must be defined with the default value	YES	YES
Portable=	This attribute must be defined with the default value	YES	YES
Production=	This attribute must be defined with the default value	YES	YES
Tab=	This attribute must be defined with the default value	INSTALL	INSTALL

4.1.3.1.4 Defining Supported Operating Systems

The <SupportedOS> syntax defines which Windows operating systems are supported by the custom script. This can be defined as all Windows operating systems, or it can be one or more families of Windows operating systems.

Supported OS entries which VManager will recognize within the XML file(s):



ALL
Windows 7 Family
Windows 7 x64 Family
Windows 7 x32 Family
Windows Vista Family
Windows Vista x64 Family
Windows Vista x32 Family
Windows Vista SP1 Family
Windows Vista x64 SP1 Family
Windows Vista x32 SP1 Family
Windows XP Family
Windows XP SP3 Family
Windows XP Professional SP3 Family
Windows XP Professional SP3
Windows XP Media Center 2005 SP3
Windows XP Tablet PC 2005 SP3
Windows XP Home Edition SP3
Windows XP SP2 Family
Windows XP Professional SP2 Family
Windows XP Professional SP2
Windows XP Media Center 2005
Windows XP Tablet PC 2005
Windows XP Home Edition SP2
Windows XP Professional SP1
Windows XP x64 Family
Windows XP x64 SP1
Windows XP x64 SP2

Example 1:

```
<SupportedOS>  
<OS>Windows Vista x32 Family</OS>  
<OS>Windows Vista x32 SP1 Family</OS>  
</SupportedOS>
```

A script which defines its supported OS's as above will only be displayed in the 32 bit versions of the Vista operating systems that contain no service pack and in 32 bit versions of Vista operating systems with service pack 1.

Example 2:

```
<SupportedOS>  
<OS>Windows XP Family</OS>  
<OS>Windows 7 Family</OS>  
</SupportedOS>
```

A script which defines its supported OS's as above will only be displayed in all 32 bit Windows XP operating systems and all Windows 7 operating systems.

Example 3

<SupportedOS>

<OS>ALL</OS>

</SupportedOS>

A script which defines its supported OS's as above will be displayed in all Windows operating systems.

4.1.3.1.5 Defining Supported Platforms

The <SupportedPlatform> syntax defines the hardware platforms that are supported by the custom script. This can be defined as all platforms, or one or more platform families.

Supported Platform entries that VManager will recognize within the XML file(s):

Entry	Description
ALL	Displays scripts for all platforms (specified and generic)
GENERIC	Displays scripts for platforms that are not recognized
Calistoga Family	Displays scripts for platforms which are recognized as part of the Mobile Napa Program (Calistoga / ICH7-M)
Crestline Family	Displays scripts for platforms which are recognized as part of the Mobile Santa Rosa Program (Crestline / ICH8-M)
Cantiga Family	Displays scripts for platforms which are recognized as part of the Mobile Montevina Program (Cantiga / ICH9-M).
IbexPeak Family	Displays scripts for platforms which are recognized as part of the Mobile Calpella Program (IbexPeak)

Example 1:

<SupportedPlatform>

<Platform>Cantiga Family</Platform>

</SupportedPlatform>

A script that defines the supported Platform as above will only be displayed for Montevina platform CRB's (for example Silver Cascade).

Example 2:

<SupportedPlatform>

<Platform>Generic</Platform>

<Platform>IbexPeak Family</Platform>



</SupportedPlatform>

A script that defines the supported Platforms as above will only be displayed for platforms that are identified as part of the IbexPeak Family (ie Red Fort CRB) and on platforms that were not recognized, which are considered generic platforms.

Example 3:

<SupportedPlatform>

<Platform>ALL</Platform>

</SupportedPlatform>

A script that defines the supported Platforms as above will be displayed on all Platforms.

4.1.3.1.6 Defining VManager Default Scripts

VManager has a "Select Defaults" feature that is used to automatically pre-select the scripts located in the Install / Config tab that have been identified as default scripts. The very first time VManager is run it will enable this feature. For all other VManager launches nothing will be automatically pre-selected, but at anytime the "Select Defaults" feature can be enabled by the user from the VManager interface. The definition of whether a script is a default is contained in the InstallApp.xml file and is based on the values defined for the <OSDefault> and <PlatformDefault> syntax.

The possible entries for <OSDefault> or <PlatformDefault> are the same as the supported options for <SupportedOS> and <SupportedPlatform>. There is one additional value of "NONE" which is used to denote that the script is not to be selected as a default.

VManager will evaluate the contents of <OSDefault> and <PlatformDefault> and when both conditions are true it will select the script as part of the "Select Defaults" feature. If either one of these entries contains a value of "NONE" the script will not be selected as a default.

Example 1:

<OSDefault>

<OS>Windows XP Professional SP3</OS>

</OSDefault>

<PlatformDefault>

<Platform> IbexPeak Family</Platform>

</PlatformDefault>

A script that defines the default OS and Platform as above will only be selected as a default script for Windows XP Professional SP3 on IbexPeak Family Platforms.

Example 2:

```
<OSDefault>
<OS>NONE</OS>
</OSDefault>
<PlatformDefault>
<Platform>IbexPeak Family</Platform>
</PlatformDefault>
```

A script that defines the default OS and Platform as above will not be selected as a default script because the default OS has been defined as NONE.

Example 3:

```
<OSDefault>
<OS>NONE</OS>
</OSDefault>
<PlatformDefault>
<Platform>NONE</Platform>
</PlatformDefault>
```

A script that defines the default OS and Platform as above will not be selected as a default script because both values have been set to NONE.

This erased part has moved moved to chapter 2: Writing Scripts For VManager – please refer to subchapter 2.5

4.1.3.1.7 Example of a Test Script Entry

Below is an example of what a script entry in the TestApp.xml would look like for adding a script called "Custom Script 1". This script will be displayed on the test tab.

```
<Test>
```

Advanced Features



```
<App Name="Custom Script 1" RunCmd="script1.exe" Reboot="No" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" />
```

```
<User>NONE</User>
```

```
<SupportedOS>
```

```
<OS>Windows Vista x32 Family</OS>
```

```
<OS>Windows Vista x32 SP1 Family</OS>
```

```
</SupportedOS>
```

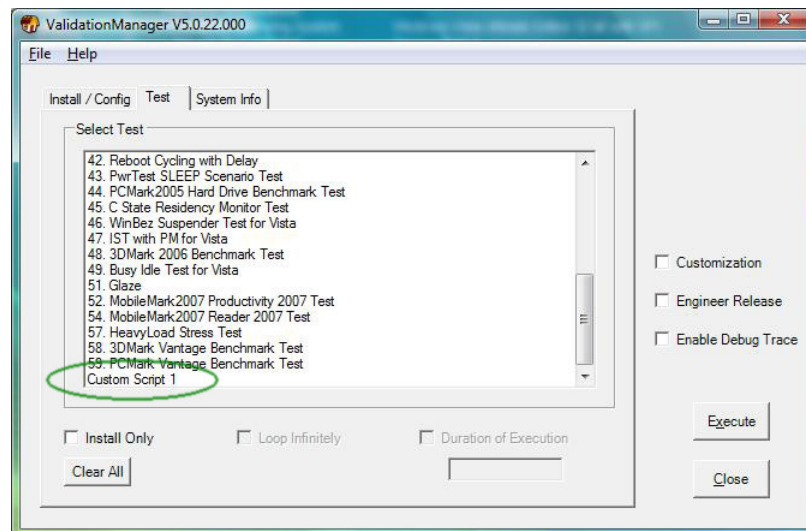
```
<SupportedPlatform>
```

```
<Platform>GENERIC</Platform>
```

```
<Platform>Cantiga Family</Platform>
```

```
</SupportedPlatform>
```

```
</Test>
```



4.1.3.1.8 Differences between Install/Config and Test XML Entries

An XML entry for an install/config script is almost identical to the entries for test scripts. There are a few differences which should be noted.

Differences between the entries:

Test scripts don't have sections, each test is defined with the following syntax:

```
<Test>
```

```
...
```

```
</Test>
```

The App syntax is the same up until the final attribute. Tab= is not included for test scripts. Example:

```
<App Name="Custom Script 1" RunCmd="script1.exe" Reboot="No" Order="999"
DVD="Yes" Portable="Yes" Production="Yes" />
```

The test tab does not have any default selections, so <OSDefault> and <PlatformDefault> should not be defined for test scripts.



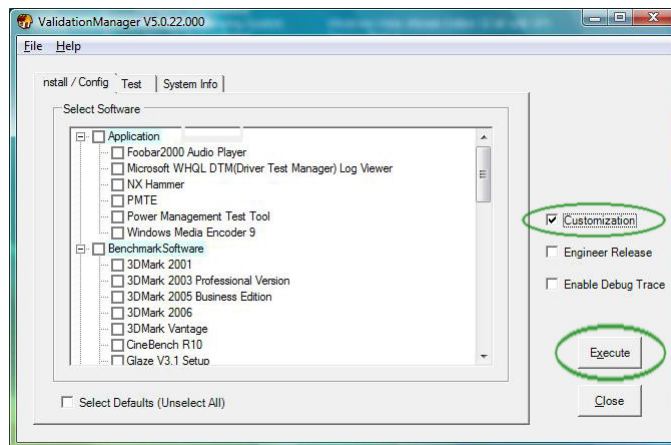
4.1.4 Procedure for Adding Custom Scripts to VManager

A step by step walk through for adding custom scripts to VManager:

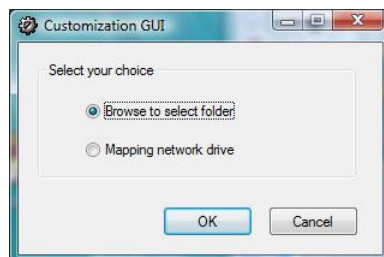
Launch VManager by double clicking on the icon located on the Windows desktop of the test system.



Click the "Customization" check box and then click "Execute".



Navigate to the location of the files to be added to VManager
"Browse to select folder"
Select "Browse to select folder" and click "OK".



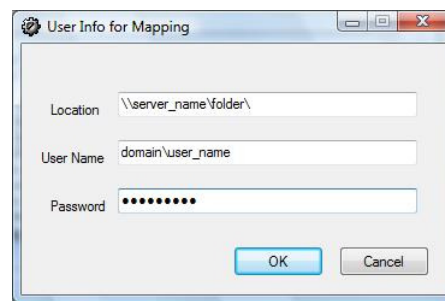
Navigate to the directory where the script and XML files are located then click "OK".



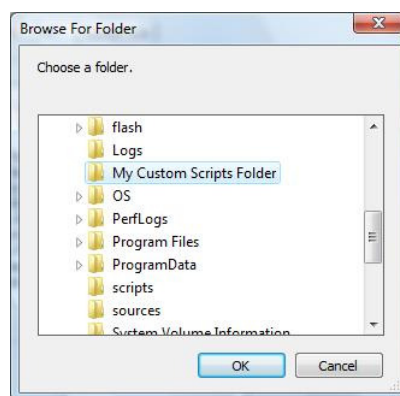
Mapping Network Drive

Select "Mapping Network Drive" and click "OK".

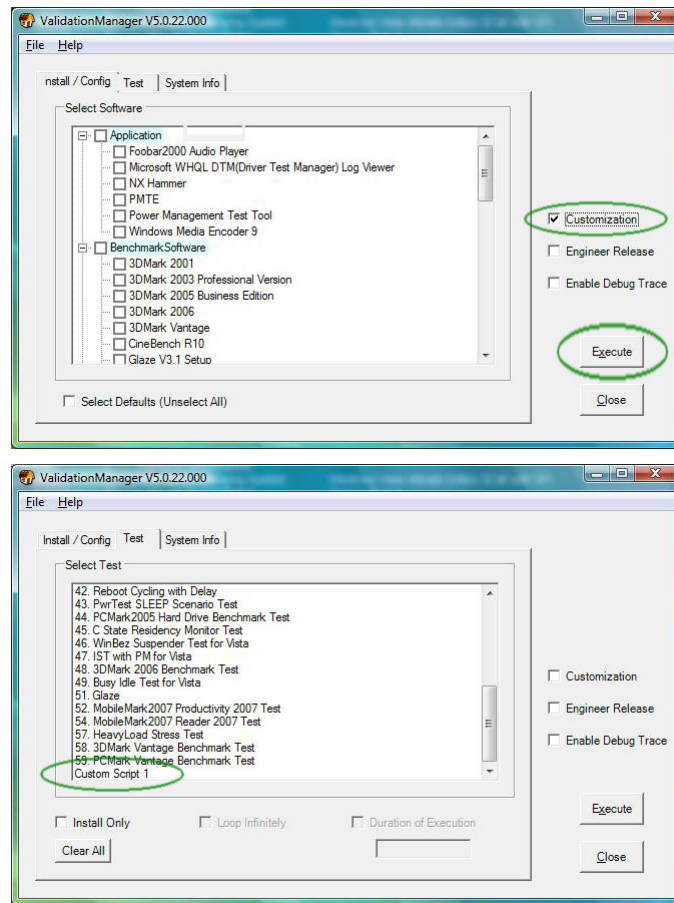
Enter network location, username, and password and click "OK".



Navigate to the directory where the script and XML files are located then click "OK".



VManager will copy the files and append the XML files.
VManager should now display the custom scripts.



4.2 Optional Reboot

<>

4.3 Install Only

<>

5 Appendix

5.1 Script Writing BMKs for VManager

The XML files can be used to do the OS and platform support detection for the scripts, so that it does not have to be done in the scripts.

Scripts should be able to handle checking for any files or application dependencies they will reply upon.

If network usage is needed, the scripts should be able to properly map network drives.

Scripts should clean up changes made to the system. For instance, temp files, directories, or registry keys.

Log script execution information to a log file. It is recommended that the log file be placed in the (local drive):\Logs directory which is automatically created by VManager.

5.2 VMLite and Script Writing for VManager

VMLite is VManager like program that provides an automated means to configure a Windows operating system, install drivers, install applications, and select test to be run in a Windows OS, conveniently during Preload phase from Windows PE. The VMLite supports install, config and test scripts that do not require user input at any time of execution.

VMLite displays all the scripts defined for use in and for VManager as long as XML <User> element is set to NONE. VMLite is simply a link allowing more efficient automation. If satisfying script selection is made in VMLite there is no need to execute VManager in Windows OS. In this scenario selected scripts will start executing after Preload phase on a test platform.

Customization is not supported by VMLite. To add custom scripts user must use VManager.

5.3 Troubleshooting

Problem	Suggestion
My network drives keep disappearing.	VManager will remove all network mappings each time it is launched. If the custom scripts need to use the network, please be sure to check for it first then re-map it if it is missing.



Something went wrong with my custom script.	Check the log file generated by the script in the (local drive):\logs directory.
Where is my custom script located on the local system?	All script executable files are copied to the (local drive):\scripts directory.
I do not see any scripts listed in VManager.	The XML file used to import the scripts into VManager was not formatted correctly. Please review the FAQ section on the MPG Validation Automation Services website for contact information.
I used customization to add some custom scripts, but they do not appear in VManager.	<p>Verify that the correct directory was selected for import.</p> <p>Verify that the scripts and XML files were stored in the same source directory at the time of import.</p> <p>Verify the XML file(s) was correctly formatted</p> <p>Check to make sure the test system (OS and Platform) is supported for the scripts that were added.</p>

5.4 Contact Information

For additional information visit <http://msw.co.intel.com/AutomationServices>