Configuring SNMP on VMWare ESX Server 4.0

Dell Technical White Paper

By *Sankara Gara*
Dell | Product Group - Enterprise

Revision History

Version	Date	Description	Author(s)
1.0	20 May 2009	First Version	Sankara Gara

CONTENTS

Introduction

On ESX4.0 server, the VMWare SNMP provider is now its own daemon (**vmware-hostd**) and does not register with the system default SNMP daemon (**snmpd**), requiring a management console to request VMWare SNMP data on another port than what OMSA registers with (**snmpd**). This white paper describes a procedure to configure the VMWare SNMP daemon and the system default SNMP daemon so that they both can provide SNMP data on one port – the **snmpd** one.

Configuring SNMP Service

The objective of this configuration is to allow ESX4.0 server to be managed through a single default port 161 using SNMP protocol. To do this, **snmpd** is configured to use the default port 161 and **vmware-hostd** is configured to use a different (unused) port, e.g. 167. Any SNMP request on the VMWare MIB branch will be rerouted to **vmware-hostd** using the proxy feature of the **snmpd** daemon.

The VMWare SNMP configuration file can be modified manually on the ESX server or by running VMWare RCLI command **vicfg-snmp** from a remote system (Windows or Linux). The RCLI tools can be downloaded from the VMware website (***http://www.vmware.com/download/vi/drivers_tools.html)***.

Below are the required steps for the configuration. *Following these steps is a python script that automates the configuration process. You may copy and paste the contents to a file and run it on the ESX server to do the configuration*.

1. Edit the VMWare SNMP configuration file *(/etc/vmware/snmp.xml*) either manually or run the following vicfg-snmp commands to modify the SNMP configuration settings including the SNMP listening port, community string, and the trap target ipaddress/port and trap community name and then enable the VMWare SNMP service.

   ```
   vicfg-snmp.pl --server <ESX_IP_addr> --username root --password <password>
   -c <community name> -p X –t <DMC_IP_Address>@162/<community name>
   ```

   Above, *X* represents an unused port. To find an unused port, you may look at the */etc/services* file for the port assignment for defined system services. Also, to make sure that the port selected is not currently being used by any application/service, run the `netstat –a` command on the ESX server**.**

   NOTE: Multiple DMC IP addresses can also be mentioned using a comma-separated list.

   To enable VMWare SNMP service run the following command:

   ```
   vicfg-snmp.pl --server <ESX_IP_addr> --username root --password <password>
   -E
   ```

   To view the configuration settings run the following command:

```
vicfg-snmp.pl --server <ESX_IP_addr> --username root --password <password>
-s
```

After the modification, the configuration file will look like this:

```
<?xml version="1.0">
<config>
      <snmpSettings>
           <enable>true</enable>
            <communities>public</communities>
            <targets>143.166.152.248@162/public</targets>
            <port>167</port>
      </snmpSettings>
</config>
```

2. Stop the SNMP service if it is already running on your system:

   ```
   service snmpd stop
   ```

3. Add the following line at the end of the */etc/snmp/snmpd.conf*:.

   ```
   proxy -v 1 -c public udp:127.0.0.1:X .1.3.6.1.4.1.6876
   ```

   Where *X* represents the unused port specified above, while configuring SNMP.

   ```
   trapsink <DMC_IP_Address> <community_name>
   ```

   The trapsink specification is required to send traps defined in the proprietary MIBs.

4. Restart **mgmt-vmware** service with the following command:

   ```
   service mgmt-vmware restart
   ```

5. Restart the **snmpd** service with the following command:

   ```
   service snmpd start
   ```

   NOTE: If the srvadmin (OMSA) is installed and the services are already started they should be restarted as they depend on the on **snmpd** service.

6. So that the **snmpd** daemon starts upon every reboot, run following command:

   ```
   chkconfig snmpd on
   ```

7. Run the following command to ensure that the SNMP ports are open before sending traps to the management station.

   ```
   esxcfg-firewall -e snmpd
   ```

Sample Script for Configuring SNMP on VMWare ESX Server 4.0

The following sample python script as an example to configure SNMP in the VMware ESX Server 4.0 environment. This script may work well in some environments, while in other environments you may need to develop your own script entirely from the beginning.

CAUTION: The sample scripts are provided as examples only and have not been tested nor are they warranted in any way by Dell; Dell disclaims any liability in connection therewith. Dell provides no technical support with regard to content herein. The script has been written to reconfigure a new non-tampered default snmp.xml and snmpd.conf files present on a ESX 4.0 System.

```
#!/usr/bin/python
#!/bin/sh
#
#     DELL INC. PROPRIETARY INFORMATION
#
#     This software is supplied under the terms of a license agreement
#      or nondisclosure agreement with Dell Inc. and may not be copied
#     or disclosed except in accordance with the terms of that agreement.
#
#     Copyright (c) 2009-10 Dell Inc. All Rights Reserved.
#     (c) 2009 Dell Inc.
#
'''
# ------------------------------------------------------------------------
---
#     SCRIPT NOTES
#     Script
#
#
#     ita_esx4_snmp_config.py is provided to configure the snmp settings and
#     enablement of snmp traps on a VM ESX 4.0 System.
#
#      (c) 2009 Dell Inc.  All rights reserved.
#    THIS SOFTWARE IS DISTRIBUTED IN THE HOPE THAT IT WILL BE USEFUL, BUT IS
#    PROVIDED  ISWITHOUT ANY WARRANTY, EXPRESS, IMPLIED OR OTHERWISE,
#    INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTY OF MERCHANTABILITY OR
#    FITNESS FOR A PARTICULAR PURPOSE OR ANY WARRANTY REGARDING TITLE OR
AGAINST
#    INFRINGEMENT.  IN NO EVENT SHALL DELL BE LIABLE FOR ANY DIRECT,
INDIRECT,
#    INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
#    NOT LIMITED TO, PROCUREMENT OF SUBSTUTUTE GOODS OR SERVICES; LOSS OF
USE,
#    DATA, OR PROFITS;OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY
#    OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
```

```
#   NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
#    EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#           The sample scripts are provided as examples for customers that
#     want to develop their own deployment process. Some customers may find
#     that the scripts work well in their environment, while some customers
#     may need to develop their own scripts entirely from scratch. The sample
#     scripts are provided as examples only and have not been tested nor are
#     they warranted in any way by Dell; Dell disclaims any liabilty in
#     connection therewith. Dell provides no technical support with regard
#     to content herein. The script has been written to reconfigure a new
#     non-tampered default snmp.xml and snmpd.conf files present on a ESX 4.0
#     System.
#
# ------------------------------------------------------------------------
---
#     LIST OF ERRORS:
#
#     "requires an argument" :-
#                 In case of value of argument being not provided.
#
#     "Script only Applicable for VMWARE ESX 4" :-
#                 Script being run on a system other than an ESX 4.0
#
#     "No modifications performed as no options given." :-
#                 User runs the scripts witout any options
#
#     "Unsupported System":-
#                 User runs the script on a non ESX 4.0
#
# ------------------------------------------------------------------------
---
#     ERROR RETURN CODES:
#
#     1: Wrong Usage
#     2: Backup Failure
#     3: Service Failure
#     4: System Mismatch
#
# ------------------------------------------------------------------------
---
#     LIST OF MODIFIED FILES :
#     /etc/vmware/snmp.xml
#     /etc/snmp/snmpd.conf
#
#           A backup of original snmpd.xml and snmpd.conf will be created
#     in their respective directories as snmp_bkup.xml and snmpd_bkup.conf.
#     Everytime the script is run, log will be generated and stored
#     in /tmp/ita_esx_log.txt.
#
# ------------------------------------------------------------------------
---
#     MODIFICATIONS PERFORMAED:
#
#           The script performs a check if the system is a VMWARE ESX 4.0
#     system or not. It proceeds only on an ESX 4.0 it proceeds, otherwise
```

```
#     it exits flagging "Unsupported System."
#
#     All the options provided by user are extracted and we exit without any
#     modifications if all are found to be NULL i.e user runs without any
#     options.
#
#           The snmpd service is stopped and for each field, namely
#     community, port and target, a check is performed to figure out if the
#     user has already configured it. If not the new supplied values are
#     added. In Case if the user has already configured some fields, new
#     values are appended in a comma seperated format.
#
#           The vmware management service and the snmpd service are
#     restarted and an exception is created for snmp service through the
#     iptables firewall.
#
# ------------------------------------------------------------------------
---
#
#     Assumptions:
#           The script assumes that the snmp.xml and snmpd.conf files are
#     present.The user running the script has root privelages.
#
# ------------------------------------------------------------------------
---
'''
#check if the system is a Vmware ESX 4.0 or not
import sys
import getopt
import xml.dom.minidom
import string
import os
import shutil
import time

log_fd = None
# Set defaults

error=0
warning=1
informational=2

modified_line1='## Begin Modified by ita.py ##\n'
modified_line3='\n## End Modified by ita.py ##\n'

system_check='vmware -v | grep -q "ESX 4"'

error_wrong_usage=1
error_backup_failure=2
error_service_failure=3
error_system_mismatch=4

snmp_xml_path= '/etc/vmware/snmp.xml'
snmpd_conf_path= '/etc/snmp/snmpd.conf'
snmpd_conf_bkup_path='/tmp/snmpd.conf.bak'
snmp_xml_bkup_path='/tmp/snmp.xml.bak'
proxy_line_index=None
```

```
config = { 'communities': None,
            'communities_conf': None,
            'port': None,
            'targets': None,
            'force': False,
            'debug': False,
            'verbose': 0,
            'logfile': '/tmp/ita_esx4_snmp_config.log',
            }

def display_usage():
      print'''
      Usage:
      ./ita.py -c -p -t
      arguments:
      -c| --community:  Community name
      -p| --port:             Port number
      -t| --targets:          Targets value for sending traps
      -f| --force:            Overwrite an already configured port
      -d| --debug:            Debug Mode
      -l| --logFile:          Logfile Name with complete path
      -v| --verbose:          level(minimal=0,default=1,maximum=2)
      -h| --help:             Display the help
      -o| --document          Display the Script Comments
'''

def parse_arguments(args, config):
      '''
      parse_arguments()
      The functions parses all the arguments provided
      by the user and check for any missing arguments
      or errors.
      '''
      global log_fd
      try:
            opts, rem_args = getopt.gnu_getopt(args, 'hc:p:t:fdl:vo',
['help', 'communities=', 'port=', 'targets=', 'force', 'debug', 'logfile=',
'verbose', 'document'])
      except getopt.GetoptError:
            raise
      #fetch all optional arguments passed by user
      for arg, val in opts:
            if arg in ( '-h', '--help'):
                  display_usage()
                  sys.exit(0)
            elif arg in ('-c', '--communities'):
                  config['communities'] = val
            elif arg in ('-t', '--targets'):
                  config['targets'] = val
            elif arg in ('-p', '--port'):
                  try: config['port'] = int(val)
                  except ValueError: raise 'Port must be an integer'
            elif arg in ('-f', '--force'):
                  config['force']=True
            elif arg in ('-d', '--debug'):
                  config['debug']=True
```

```
            elif arg in ('-l', '--logfile'):
                  config['logfile']=val
            elif arg in ('-v', '--verbose'):
                  config['verbose']=config['verbose']+1
            elif arg in ('-o', '--document'):
                  display_usage()
                  print __doc__
                  sys.exit(0)

      if config['verbose'] > 2: config['verbose'] = 2

      #open the user specefied logfile
      log_fd = open(config['logfile'], 'a')
      #communities field is mandatory
      if config['communities'] is None:
            print 'Communities field is mandatory'
            sys.exit(error_wrong_usage)

      #if no options are given just exit.
      if config['communities'] is None and config['targets'] is None and
config['port'] is None:
            print 'No options are given. Exiting without modifications'
            sys.exit(error_wrong_usage)

def get_tag_text(dom, tag_name):
      '''
            get_tag_text()
            The function fetches the present values of
            the speceified tag in the existing snmp.xml.
      '''
      text = ''
      for node in dom.getElementsByTagName(tag_name)[0].childNodes:
            if node.nodeType == dom.TEXT_NODE:
                  text = text + node.nodeValue
                  if len(text) == 0: text = None
            return text


def get_snmp_values_from_xml(dom):
      '''
            get_snmp_values_from_xml()
            The function fetches the values of all
            the existing fields which needs to be
            modified from the exisiting snmp.xml.
      '''
      try: snmp_communities =  get_tag_text(dom, 'communities').split(',')
      except: snmp_communities = []

      try: snmp_enable =  get_tag_text(dom, 'enable')
      except: snmp_enable = None

      try: snmp_port =  int(get_tag_text(dom, 'port'))
      except: snmp_port = None

      try: snmp_targets =  get_tag_text(dom, 'targets').split(',')
      except: snmp_targets = []
```

```
	return snmp_communities, snmp_enable, snmp_port, snmp_targets


def replace_text(dom, node_name, text, parent_node_name = None):
	'''
		replace_text()
		The function replaces the existing values
		present in the xml with the new ones
		specified by the user.
	'''
	try:
		node = dom.getElementsByTagName(node_name)[0]
	except IndexError:
		# Try creating the node @ parent
		parent_node = dom.getElementsByTagName(parent_node_name)[0]
		node = dom.createElement(node_name)
		parent_node.appendChild(node)
	# Remove old text
	for n in node.childNodes:
		if n.nodeType == dom.TEXT_NODE:
			node.removeChild(n)
	# Add new text
	node.appendChild( dom.createTextNode( text ) )

def log(msg, loglevel):
	global log_fd
	if loglevel > config['verbose']: return
	log_entry = '%s :%d:%d: %s' % (time.ctime(), os.getpid(), loglevel,
msg)
	if config['debug'] is True: # Onto stdout
		print log_entry
	log_fd.write(log_entry+'\n')


def modify_snmp_xml(config):
	'''
		modify_snmp_xml()
		The is the main function which does
		conditional modifications in the snmp.xml
		based on the opitonal argguments provided
		by the user.
	'''
	#parsing existing snmp.xml
	log(msg='Parsing existing snmp.xml',loglevel=informational)
	dom=xml.dom.minidom.parse(snmp_xml_path)
	snmp_communities, snmp_enable, snmp_port, snmp_targets =
get_snmp_values_from_xml(dom)

	#Add communites only if it does'nt already exists
	if config['communities'] not in snmp_communities:
		snmp_communities.append(config['communities'])

	#Add targets only if it does'nt already exists
	if config['targets'] is not None and config['targets'] not in
snmp_targets:
		snmp_targets.append(config['targets'])
```

```
      #Configure port if it does'nt exist. Else replace earlier port if -f
option os specified
      if config['port'] is not None and ( snmp_port is None or
config['force'] is True):
            snmp_port = config['port']

      log(msg='Updating communites field',loglevel=informational)
      replace_text(dom=dom, parent_node_name='snmpSettings',
node_name='communities', text=string.join(snmp_communities, ','))
      config['community_conf']=string.join(snmp_communities, ',')

      log(msg='Updating targets field',loglevel=informational)
      replace_text(dom=dom, parent_node_name='snmpSettings',
node_name='targets', text=string.join(snmp_targets, ',') )

      log(msg='Updating port field',loglevel=informational)
      if config['port'] is not None:
            replace_text(dom=dom, parent_node_name='snmpSettings',
node_name='port', text=str(snmp_port))

      #Add enable field if absent.If already present, set its value to True
      log(msg='Updating enable field',loglevel=informational)
      replace_text(dom=dom, parent_node_name='snmpSettings',
node_name='enable', text='true')
      return dom


def revert_changes(error_msg, error_code):
      '''
            revert_changes()
            In case of a failure to perform
            the modifications, this function
            reverts the system back to the
            original stste.
      '''
      shutil.move(snmp_xml_bkup_path, snmp_xml_path)
      shutil.move(snmpd_conf_bkup_path, snmpd_conf_path)
      log(msg=error_msg,loglevel=error)
      sys.exit(error_code)


def main():


      parse_arguments(sys.argv[1:], config=config)
      log(msg='Parsed Arguments',loglevel=informational)

      result=raw_input('Please read the notes first by executing the script
using option -o or --document. Are You Sure You wamt to Continue? (Y/N) : ')
      if result is not 'Y' and result is not 'y':
            sys.exit(0)

      log(msg='Checking System OS',loglevel=informational)
      ret=os.system(system_check)
      if ret is not 0:
            log(msg='Unsupported System. Script executes only on ESX
4.0',loglevel=error)
```

```
            sys.exit(error_system_mismatch)

      log(msg='Modifying XML',loglevel=informational)
      dom = modify_snmp_xml(config)

      log(msg='Checking SNMP Status',loglevel=informational)
      ret_val=os.system('pidof snmpd  > /dev/null')
      if ret_val==0 :
            log(msg='Stopping SNMP service',loglevel=warning)
            os.system('service snmpd stop')
            time.sleep(5)
            ret_val=os.system('pidof snmpd  > /dev/null')
            if ret_val==0:
                  revert_changes(error_msg='Error: Could not Stop SNMP
Service. Terminating without modifications',error_code=error_service_failure)

      log(msg='Checking vmware-hostd Status',loglevel=informational)
      ret_val=os.system('pidof vmware-hostd  > /dev/null')
      if ret_val==0 :
            log(msg='Stopping mgmt-vmware vmware',loglevel=warning)
            os.system('service mgmt-vmware stop')
            time.sleep(5)
            ret_val=os.system('pidof vmware-hostd  > /dev/null')
            if ret_val==0:
                  revert_changes(error_msg='Error: Could not Stop vmware-
hostd Service. Terminating without
modifications',error_code=error_service_failure)

      log(msg='Creating Backup',loglevel=informational)
      shutil.copy(snmp_xml_path, snmp_xml_bkup_path)

      log(msg='Updating the snmp.xml file',loglevel=warning)
      snmp_xml = open(snmp_xml_path,'w')
      dom.writexml(snmp_xml)
      snmp_xml.close()

      shutil.copy(snmpd_conf_path, snmpd_conf_bkup_path)

      snmpd_conf = open(snmpd_conf_path).readlines()
      config['proxy_line_index'] = None

      try: port_conf =  get_tag_text(dom, 'port')
        except:
            revert_changes(error_msg="Error: No port Value
Specified",error_code=error_wrong_usage)

      proxy_entry = 'proxy -v 1 -c %s udp:127.0.0.1:%s .1.3.6.1.4.1.6876\n'
%(config['communities'],port_conf)

      try:
            proxy_line_index = [ i for i,x in enumerate(snmpd_conf) if
x.startswith(modified_line1) ][0]
            # Entry already present. Needs to be modified
            snmpd_conf[proxy_line_index+1] = proxy_entry
      # New entry needs to be created
      except IndexError:
            snmpd_conf.extend(modified_line1)
```

```
            snmpd_conf.extend(proxy_entry)
            snmpd_conf.extend(modified_line3)

      log(msg='Updating the snmpd.conf file',loglevel=warning)
      open(snmpd_conf_path, 'w').writelines(snmpd_conf)

      log(msg='Starting SNMP service',loglevel=warning)
      os.system('service snmpd start')
      time.sleep(5)
      ret=os.system('pidof snmpd > /dev/null')
      if ret is not 0:
            revert_changes(error_msg="Error: Could not start snmpd
service",error_code=error_service_failure)

      log(msg='Starting mgmt-vmware vmware',loglevel=warning)
      os.system('service mgmt-vmware start')
      time.sleep(7)
      ret=os.system('pidof vmware-hostd > /dev/null')
      if ret is not 0:
            revert_changes(error_msg="Error: Could not start mgmt-vmware
service",error_code=error_service_failure)

      log(msg='Unblocking SNMP service from Firewall',loglevel=warning)
      os.system('esxcfg-firewall --e snmpd')
      ret=os.system('esxcfg-firewall --q snmpd')
      if ret is not 0:
            revert_changes(error_msg="Error: Could not unblock snmpd service
from firewall",error_code=error_service_failure)

      log(msg='SNMP modifications completed
Successfully',loglevel=informational)
      print('SNMP modifications completed Successfully')
if __name__ == '__main__':
      main()
sys.exit(0)
```

Testing the Configuration

Basic testing of the configuration involves walking the MIB-2 system branch, the Dell 10892 MIB branch and the VMWare MIB branch using port 161. The MIB walks will be successful (no timeout or no null data) if the configuration is correct. Also send test traps from the ESX4.0 server and check if the traps are received in DMC Event Console.

1) Using any SNMP MIB browser or **snmpwalk** command to walk the following OIDs.

   System - **.1.3.6.1.2.1.1**

   Dell (if OMSA installed and started) **- .1.3.6.1.4.1.674.10892.1**

   VMWare - **.1.3.6.1.4.1.6876**

2) Run the following command from a remote system (Windows or Linux) to send a test traps from the ESX4.0 server.

```
vicfg-snmp.pl --server <ESX_IP_addr> --username root --password <password>
-T
```