IBM Software Group

IBM Presentations: z/OS SNMP Basics: Principles of Configuration and Diagnostics for SNMP Environment

David T. Britt

z/OS CommServer Level 2 Support

WebSphere® Support Technical Exchange

ON DEMAND BUSINESS

Agenda

- Introduction
- Basic SNMP Concepts
  - SNMP Component Relationships
  - The SNMPv1 Security Model
- Configuring the PW.SRC file
- Configuring the SNMPTRAP.DEST file
- Using the z/OS UNIX osnmp command to test your configuration
- Using the SNMP agent debug trace to troubleshoot configuration problems

Introduction

- This presentation is the first of a multi-part series detailing the set up process for the Simple Network Management Protocol (*SNMP*) application.
  - This first part will discuss the fundamental concepts of the SNMP environment, the SNMPv1 security model, and the basic configuration needed make the environment useful
  - The second part, to be delivered on a later date, will discuss the more advanced configuration of the SNMP environment, including the SNMPv2c and SNMPv3 security models.

Basic SNMP Components

- *Management Information Base:* Referred to as the *MIB*, this is the management data which can be exchanged between the different SNMP components. The MIB consists of individual objects, which can be referred to by their numerical Object Identifier (OID) or their textual Object Descriptors. An example of this is the system MIB, supported by the SNMP agent. The following is one of the objects within the system MIB:

  sysDescr / 1.3.6.1.2.1.1.1

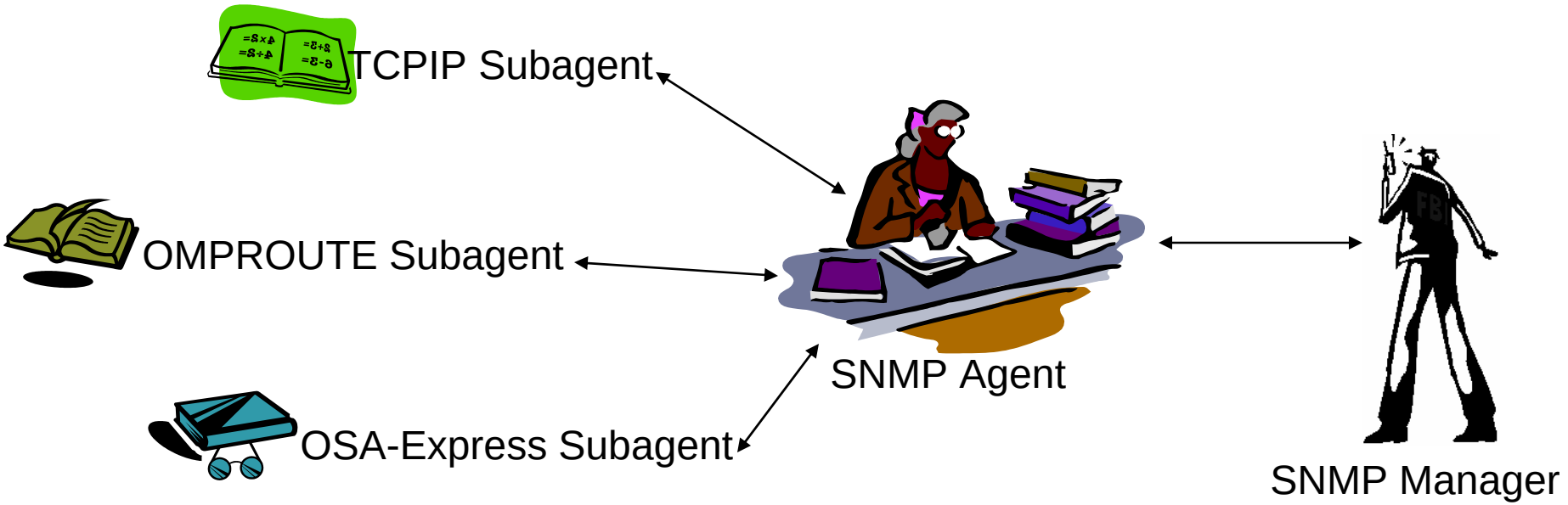
  sysDescr is the Object Descriptor, whereas *1.3.6.1.2.1.1.1* is sysDescr's OID.

Basic SNMP Components, continued

- *Agent:* This is the primary SNMP component. The agent is responsible for a basic MIB (containing system and agent specific information), accepting queries from managers, forwarding queries to subagents, and sending out traps.
- *Manager:* This is the SNMP component from which requests are generated for agent and subagent MIB objects. The manager can query MIB objects, and can also set the value of some objects.
- *Subagent:* This component is typically part of another application, and supports a MIB which contains objects specific to that application.

SNMP Component Relationships

- The interaction between the manager, agent, and subagents can be better understood using the following diagram:



SNMP Component relationships, continued

- In the previous diagram, the TCPIP, OMPROUTE, and IOBSNMP subagents are used as an example.
  - The TCPIP subagent supports a MIB which consists of management data specific to the TCPIP stack (such as connection tables, link and device status, etc.).
  - The OSA-Express subagent supports a MIB consisting of management data specific to OSA cards (such as channel tables and performance tables).

SNMP Component Relationships, continued

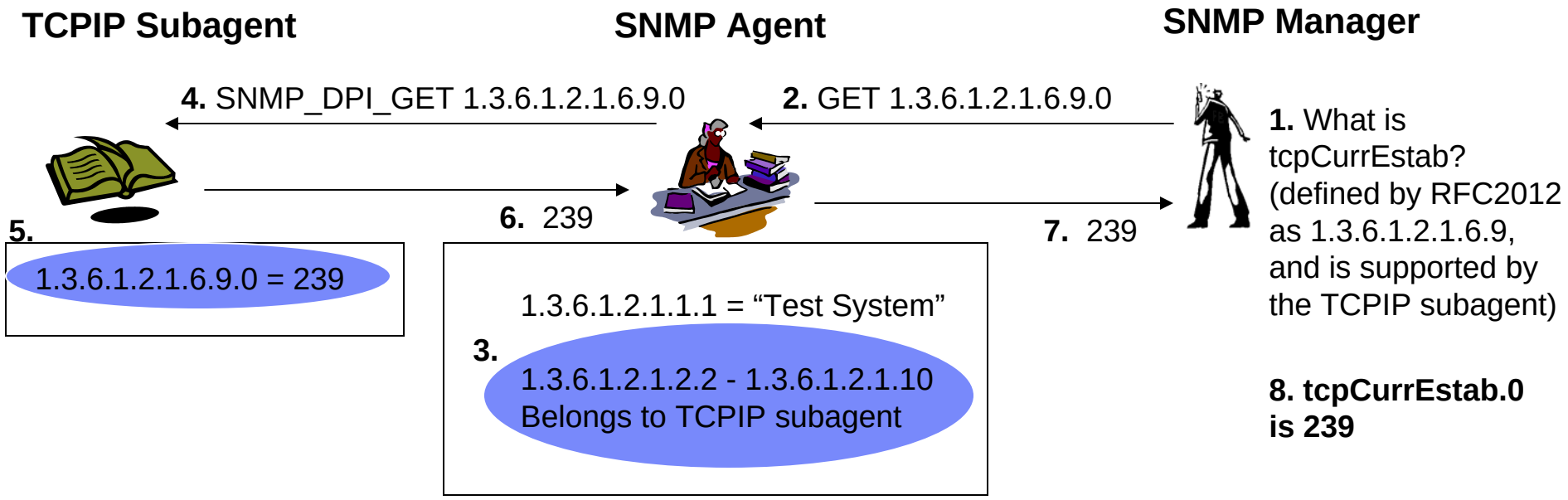
- The subagents are connected to the agent via a Distributed Protocol Interface (DPI), defined by RFC 1592
- Upon connecting to the agent, the subagents register the MIB objects they support with the agent.
- Any manager requests for these objects can then be forwarded by the agent to the appropriate subagent.

SNMP Component Relationships, continued

- As seen in the diagram, the manager communicates with the SNMP agent only. Using the User Datagram Protocol (UDP), the manager can send requests to the SNMP agent for MIB objects supported by both the agent and by the subagents.
  - Types of requests include queries of data (get, getNext, and getBulk) as well as requests to assign the value of an object (set)

SNMP Component Relationships, continued

- An example of how a typical request is processed:



The SNMPv1 Security Model

- SNMP supports three security models:
  - SNMPv1, SNMPv2c, and SNMPv3
  - SNMPv1 will be discussed in this lecture.
  - SNMPv2c and SNMPv3 will be discussed in a later, more advanced, lecture.

The SNMPv1 Security Model, continued

- SNMPv1 is a “community based” security environment
  - In a community based model, the SNMP agent authenticates requests based on the following:
    - A community name, which acts like a password
    - The IP address (or subnet address) associated with the community name
  - Each request generated by a manager contains a community name which must correspond with the IP address of the manager’s host

The SNMPv1 Security Model, continued

- An SNMPv1 environment for the SNMP agent shipped with the z/OS Communications Server can be configured using two files: PW.SRC and SNMPTRAP.DEST
- Note that the requests—including the community name—are sent unencrypted.
  - Anyone with a sniffer or packet trace can capture the request and determine the community name

Configuring the PW.SRC file

- The PW.SRC file is used to configure the community name and IP address mappings used by the agent to authenticate a manager's request
- The syntax for each line of this file is as follows:

  community_name IP_address subnet_mask

Configuring the PW.SRC file, continued

- The following is an example of how a PW.SRC file might be configured. We will be using this example throughout the presentation.

```
v1SubAgent    10.11.12.13    255.255.255.255
iUseSnmpV1    192.168.1.0    255.255.255.0
user1         127.0.0.1      255.255.255.255
```

Configuring the PW.SRC file, continued

- The search order for the PW.SRC file is as follows:
  1. **The name of an HFS file or an MVS file specified by the PW_SRC environment variable**
  2. **/etc/pw.src HFS file**
  3. **The data set specified on SYSPWSRC DD statement in the agent procedure**
  4. **jobname.PW.SRC, where jobname is the name of the job used to start the SNMP agent**
  5. **SYS1.TCPPARMS(PWSRC)**
  6. **hlq.PW.SRC, where hlq either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used**

Configuring the PW.SRC file, continued

- How do the PW.SRC configurations work?
  - Upon receiving a request from a manager, the SNMP agent will use the community name specified within the request to identify which PW.SRC configuration should be used during the authentication process
  - The agent then applies the network mask from the correct PW.SRC configuration to the IP address from which the request was generated (this is accomplished via a logical *AND* of the two addresses)

Configuring the PW.SRC file, continued

- The agent then takes the value resulting from the *AND* of the two addresses, and compares it to the IP address from the PW.SRC file (this is accomplished via an exclusive *OR* of the resulting value and the IP address)

- To demonstrate this process, we will use our PW.SRC example to analyzing the following three requests:
  - community name User1 from 127.0.0.1
  - community name v1SubAgent from 10.11.12.14
  - community name iUseSnmpV1 from 192.168.1.101

Configuring the PW.SRC file, continued

- A request is received from 127.0.0.1 using community name *User1*
  - The agent will compare *User1* to the three community names listed in the PW.SRC file: *v1SubAgent, iUseSnmpV1,* and *user1.*
  - Because community names are case-sensitive, it will not correlate *User1* to *user1*. Instead it will discard the request, and the manager will time out.

Configuring the PW.SRC file, continued

- A request is received from 10.11.12.14 using community name v1SubAgent
  - The agent will correlate this request to the configuration

    v1SubAgent 10.11.12.13 255.255.255.255
  - The agent performs the logical *AND*

    10.11.12.14 AND 255.255.255.255 = 10.11.12.14
  - The agent performs the exclusive *OR*

    *10.11.12.14 XOR 10.11.12.13 = **0.0.0.3***
  - Because the result is non-zero, the request is discarded

Configuring the PW.SRC file, continued

- A request is received from 192.168.1.101 using community name iUseSnmpV1
  - The agent will correlate this request to the configuration

    iUseSnmpV1 192.168.1.0 255.255.255.0
  - The agent performs the logical *AND*

    192.168.1.101 AND 255.255.255.0 = 192.168.1.0
  - The agent performs the exclusive *OR*

    *192.168.1.0 XOR 192.168.1.0 = **0.0.0.0***
  - Because the result is zero, the request is authenticated

Configuring the PW.SRC file, continued

- Additional notes concerning configuring the PW.SRC file for subagents:
  - As noted earlier, SNMP subagents communicate with the SNMP agent using a DPI connection
  - Before a subagent can begin using the DPI connection, it must obtain information about the DPI socket from the agent
  - This requires that the subagent query the agent using UDP, just as a manger might do. As such, the PW.SRC must contain an entry which will allow the subagent's queries to be authenticated

Configuring the SNMPTRAP.DEST file

- In addition to handling requests from managers, the SNMP agent is also capable of sending *traps* to managers
  - traps are asynchronous notifications of events
  - traps can be generated by the agent to mark things such as the agent's initialization or an authentication failure
  - traps can also be generated by subagents for application-specific events. For example, the TCPIP subagent can generate a trap if a link goes up or down

Configuring the SNMPTRAP.DEST file, cont.

- In order for the traps to be delivered by the agent, the agent must be configured with the destinations of the traps, and the transport that will be used in sending the traps
  - the destinations are typically hosts at which managers are actively listening for traps on port 162
  - UDP is the only transport supported by the SNMP agent shipped with the z/OS Communications Server product

Configuring the SNMPTRAP.DEST file, cont.

- The syntax for each line of this file is:

  destination_host_or_IP transport

  - as noted previously, because SNMP only supports traps over UDP, *transport* will always be *UDP*
- The following is an example of how an SNMPDTRAP.DEST file might be configured. We will refer back to this example later

  10.11.12.13 UDP

  192.168.1.101 UDP

Configuring the SNMPTRAP.DEST file, cont.

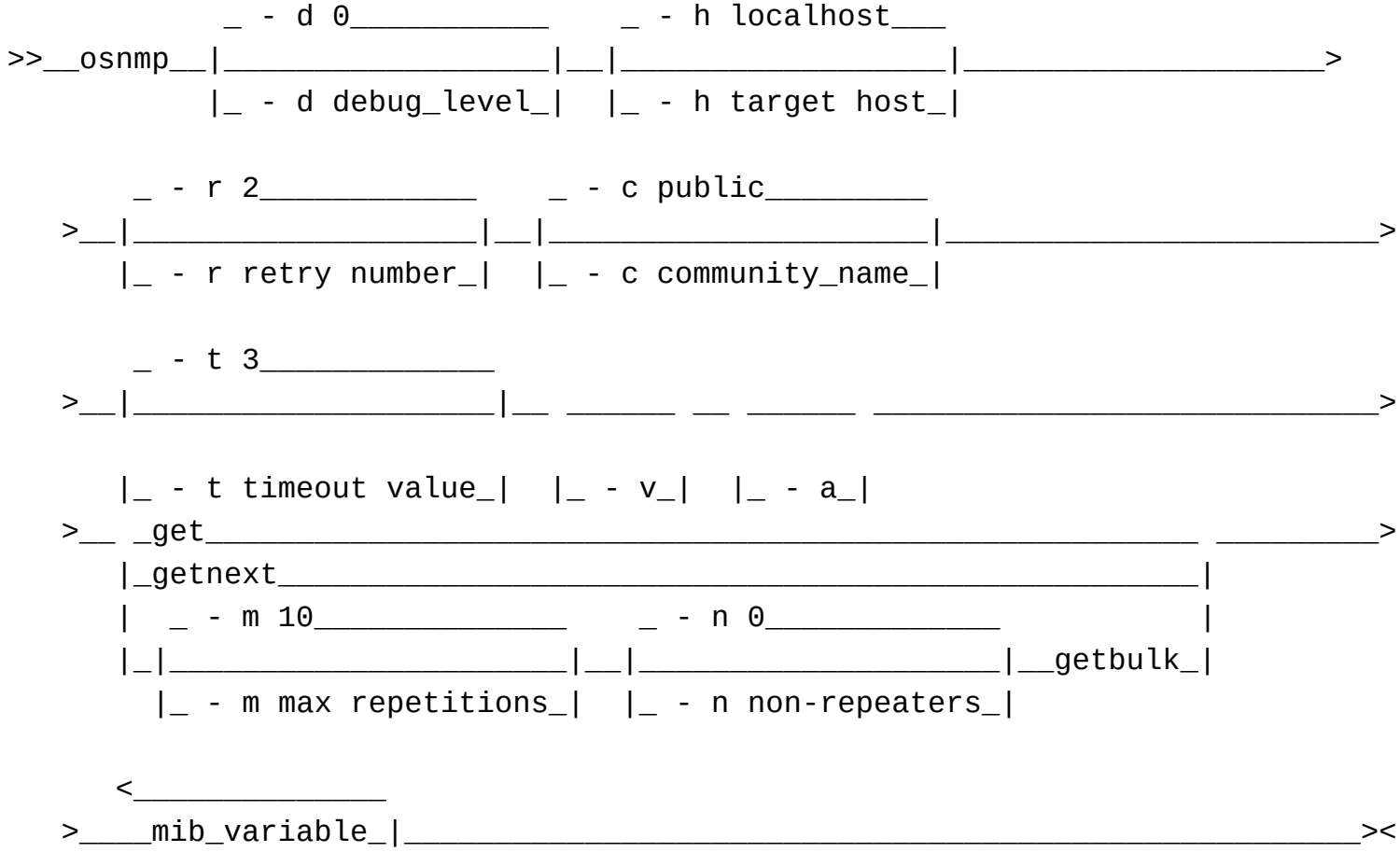
- The search order for the SNMPTRAP.DEST file is as follows:
  1. **The name of an HFS file or an MVS file specified by the SNMPTRAP_DEST environment variable**
  2. **/etc/snmptrap.dest HFS file**
  3. **The data set specified on SNMPTRAP DD statement in the agent procedure**
  4. **jobname.SNMPTRAP.DEST, where jobname is the name of the job used to start the SNMP agent**
  5. **SYS1.TCPPARMS(SNMPTRAP)**
  6. **hlq.SNMPTRAP.DEST, where hlq either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used**

Using the z/OS UNIX osnmp command

- Any SNMP manager can be used to test your configuration, but the z/OS Communications Server product is shipped with a built in manager called the z/OS UNIX osnmp command
  - this command acts like a simple manager application, allowing a user to issue requests for MIB objects such as get, getNext, getBulk, walk, bulkWalk, or set
  - the command is not intended for network management, and is more suited for testing

Using the z/OS UNIX osnmp command, cont.

- The syntax of the osnmp command is as follows:



Using the z/OS UNIX osnmp command, cont.

- For our testing purposes, the following syntax will be sufficient:

  osnmp –v –h *hostname* –c *communityName* getnext *MIBvariable*

- Successful queries of MIB variables using each community name configured in the PW.SRC file confirm that the SNMP agent is configured properly.
  - In each test, we will query the sysDescr.0 MIB value. We expect the same response (“Level 2 Test System”) in each case.
  - For our examples, we will assume that the SNMP agent is running on host 10.11.12.13

Using the z/OS UNIX osnmp command, cont.

- Test the v1SubAgent community:
  - This must be issued from the 10.11.12.13 host as the PW.SRC file is configured to associate the v1SubAgent community name with that IP address. Because the agent also runs on this host, no *–h* parameter is needed. By default the osnmp command redirects the request to the local host.

```
/:> osnmp –v –c v1SubAgent getnext sysDescr
SysDescr.0 = Level 2 Test System
```

Using the z/OS UNIX osnmp command, cont.

- Test the iUseSnmpV1 community:
  - This must be issued from a host on the 192.168.1.0 subnet as the PW.SRC file is configured to associate the iUseSnmpV1 community name with that subnet address. Because the agent is running on a different host, the –h parameter must be used to direct the request to the correct destination.

```
/:> osnmp –v –h 10.11.12.13 –c iUseSnmpV1 getnext sysDescr
SysDescr.0 = Level 2 Test System
```

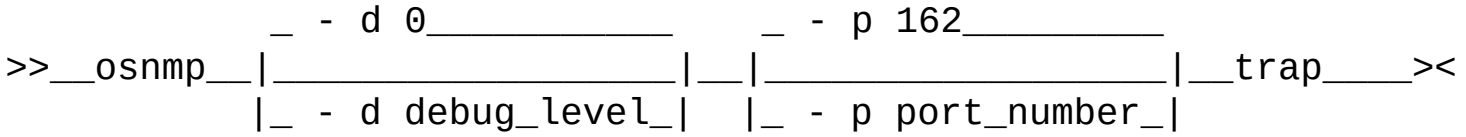
Using the z/OS UNIX osnmp command, cont.

- Test the user1 community:
  - This must be issued from the 10.11.12.13 host as the PW.SRC file is configured to associate the user1 community name with the loopback address. Because the osnmp command will use the local host's address if no destination is specified, *-h* must be used to specify the loopback address

```
/:> osnmp –v –h 127.0.0.1 –c user1 getnext sysDescr
SysDescr.0 = Level 2 Test System
```

Using the z/OS UNIX osnmp command, cont.

- The osnmp command can also be used to listen for traps, and therefore to test the SNMPTRAP.DEST configuration:



- This command can be used to test the trap configuration by running it on any z/OS system which has been configured as a destination in the SNMPTRAP.DEST file
- The output is written to standard out (STDOUT)

Using the z/OS UNIX osnmp command, cont.

- An example of using the *osnmp* trap command:

```
# osnmp -p 162 trap &
[1]     50331666
#
Display of SNMPv1 trap:
community:        public
enterprise oid:  1.3.6.1.4.1.2.3.13
          .....    ibmTcpIpMvs
agentAddress:    10.11.12.13
generic-trap:    coldStart ('00000000'h)
specific-trap:   0 ('00000000'h)
time-stamp:      0 - 0.00 seconds
```

Using the SNMP agent trace

- Three common errors encountered when configuring the SNMP agent are:
  - EZZ3310I timeout after 3 seconds
  - *OID* = noSuchName
  - Traps are not generated to the specified destinations
- All of these problems can be diagnosed using the SNMP agent trace

Using the SNMP agent trace, continued

- The SNMP agent trace has multiple levels of tracing, defined as follows:

1	Trace SNMP requests
2	Trace SNMP responses
4	Trace SNMP traps
8	Trace DPI packets level 1
16	Trace DPI internals level 2
32	Internal trace (debug level 1)
64	Extended trace (debug level 2)
128	Trace DPI internals level 2

Using the SNMP agent trace, continued

- Multiple levels of tracing can be specified by specifying the sum of the desired trace levels.
  - if trace levels 1, 2, and 4 are desired, the trace should be activated at level 7 (1 + 2 + 4)
- The trace can be started dynamically from the console:
  - /MODIFY snmp_jobname,TRACE,LEVEL=#
- However, for our testing purposes, we will start the trace at the agent's initialization
  - This can be done by adding *–d* to the *PARM* line of the JCL used to start OSNMPD, or by invoking OSNMPD with the *–d parameter*

Using the SNMP agent trace, continued

- Sample JCL to start the agent with debugging:

```
//OSNMPD PROC
//OSNMPD EXEC PGM=EZASNMPD,REGION=0M,TIME=NOLIMIT,
//  PARM='POSIX(ON) ALL31(ON)/ -d 255‘
…..
```

- An example of starting OSNMPD from the z/OS UNIX shell using the *–d* parameter:

```
osnmpd –d 255 –p 161 –s /tmp/dpi_socket
```

Using the SNMP agent trace, continued

- Timeout errors are not only the most common, but require the most debugging. The following list details the 5 steps for quickly troubleshooting such an error.
  1. The timeout could have occurred because the manager's request never reached the SNMP agent. After recreating the error with the trace running, search the trace output for the following string:

     "Received an SNMP packet on"

Using the SNMP agent trace, continued

- This message is indicative of the SNMP agent's receipt of a request, and contains the IP address information of the request's originator:

```
Received an SNMP packet on fd=3 from
   UDP 192.168.2.101 port 50009
```

- If you cannot find an instance which specifies the IP address of your manager, then the request never reached the agent.
- To resolve this, investigation must be done on the network segment between the agent and manager systems

Using the SNMP agent trace, continued

2. If the manager's request is found in the trace, then it may be that the wrong community name is being used. If this is the case, the following message will be found following the receipt of the request:

```
IDSTMOE.CZ708N04.SOURCE.S@AGV123(1939):
   rc=-14 (SNMP_RC_NOT_AUTHENTICATED) from
   snmp_process_message()
```

Using the SNMP agent trace, continued

- Ensure that the community name being used is correctly specified in the PW.SRC file and on the manager request. Remember, the community names are case-sensitive
- Ensure that the IP address and subnet mask are specified correctly

Using the SNMP agent trace, continued

3. It is possible that the manager's source IP address is not within the range assigned to that community name
   - Such occurrences would also cause an authentication failure

```
IDSTMOE.CZ708N04.SOURCE.S@AGV123(1939):
   rc=-14 (SNMP_RC_NOT_AUTHENTICATED)
   from snmp_process_message()
```

   - Check the source IP address listed in the message, and compare it to the range assigned to the community name in the PW.SRC file.

Using the SNMP agent trace, continued

- If this is the case, either correct the manager such that it uses the correct IP address, or add the IP address being used (along with the appropriate community name) to the PW.SRC file

Using the SNMP agent trace, continued

4. If you find that the correct IP address and community name pair is being used, it may be that the PW.SRC has a syntax or configuration error. This will be indicated at the beginning of the agent trace by a message such as

   EZZ6209I Line number 3 in the /etc/pw.src file has a network address:0.0 which is not in the correct format

   - Upon identifying this error, the PW.SRC file can then be corrected and the configuration re-tested

Using the SNMP agent trace, continued

5. Lastly, ensure that the correct PW.SRC file is being used.
   - The beginning of the agent trace will contain a message such as

     EZZ6266I Using file /etc/pw.src for community names configuration
   - If this is not the intended PW.SRC file, then ensure that the file being used precedes the file listed by the EZZ6266I message in the PW.SRC search order
   - Alternately, the correct configuration can be moved to the file listed in the EZZ6266I message

Using the SNMP agent trace, continued

- *noSuchName* messages indicate that a manager has requested a MIB variable which the agent does not recognize. This most commonly occurs when a manager generates a request for a MIB object supported by a subagent that is not connected to the agent.
  - This could be because the subagent has not been started
  - This could also indicate that the subagent is unable to connect to the agent, perhaps due to authentication failures when the subagent queries the agent for DPI information.

Using the SNMP agent trace, continued

- You can troubleshoot this in the same way that you would a timeout error
  - search the agent trace for the receipt of a packet that is denied with *SNMP_RC_NOT_AUTHENTICATED* or *rc=-14*
  - if this is found, ensure that the community name and IP address used by the subagent is specified correctly both in the PW.SRC file and in the subagent's configuration
  - check to see if any configuration errors were encountered in the PW.SRC file, and if the correct PW.SRC file is being used

Using the SNMP agent trace, continued

- There are three common reasons traps may not be generated to the specified destinations;
  1. If the trap not being delivered is an authenticationFailure (indicating that someone attempted to query the agent with invalid credentials), it may be that the MIB object snmpEnableAuthenTraps.0 is set to “2” instead of “1”
     - This value can be set using the following osnmp command:

       osnmp –c *community* –h *host* set snmpEnableAuthenTraps.0 1

Using the SNMP agent trace, continued

2. There is a configuration error in the SNMPTRAP.DEST file
   - Similar to the PW.SRC file, syntax or configuration errors in the SNMPTRAP.DEST will be reported at the beginning of the SNMP agent trace by a message such as

     EZZ6216I Line number 3 in the /etc/snmptrap.dest file failed host_lookup for ip address: FAKE.HOST.IBM.COM
   - Upon identifying this error, the SNMPTRAP.DEST file can then be corrected and the configuration re-tested

Using the SNMP agent trace, continued

3. The wrong SNMPTRAP.DEST file is being used
   - Similar to the PW.SRC file, the SNMPTRAP.DEST file being used will be listed at the beginning of the agent trace:

     EZZ6214I Using /etc/snmptrap.dest for trap destination file
   - Again, if this is not the correct file, the intended file can be renamed such that it is earlier in the SNMPTRAP.DEST file than the file listed in the EZZ6214I message
   - Or the correct configuration can be moved to the file listed

Additional WebSphere Product Resources

- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: www.ibm.com/developerworks/websphere/community/
- Learn about other upcoming webcasts, conferences and events: www.ibm.com/software/websphere/events_1.html
- Join the Global WebSphere User Group Community: www.websphere.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant: ibm.com/software/info/education/assistant
- Learn about the Electronic Service Request (ESR) tool for submitting problems electronically: www.ibm.com/software/support/viewlet/probsub/ESR_Overview_viewlet_swf.html
- Sign up to receive weekly technical My support emails: www.ibm.com/software/support/einfo.html

Questions and Answers