developerWorks®

Using the SNMP GetBulk request for data retrieval

Learn about IBM i SNMP performance and usability improvements

Clair M. Wood

March 19, 2015
(First published March 19, 2015)

Simple Network Management Protocol (SNMP) provides a system or network management application with the ability to gather information about network devices and to monitor them. To accomplish this, SNMP defines a set of operations for retrieving and setting data as well as monitoring for conditions being reported by the managed devices. One of these operations is GetBulk, which provides an application with the ability to easily retrieve a large amount of data with a single request. This can be particularly useful when retrieving information from the standard SNMP tables. This article describes how to use a new GetBulk API for retrieving data from a table and other new functions recently added to the SNMP support on IBM i.

Introduction

The IBM i 7.1 and IBM i 7.2 releases have recently been enhanced to provide new capabilities for SNMP-related functions. These new capabilities offer IBM i customers improved performance and enhanced usability. In this article, I will discuss these new enhancements and provide a programming example that makes use of several of them. The primary focus of this article is on using the `GetBulk` operation to easily retrieve data from one of the standard SNMP tables.

Sending SNMP responses through port 161

In the past, IBM i received SNMP requests through UDP port 161 and then sent a response through a random ephemeral port. This behavior made it difficult for administrators to configure firewalls and still allow SNMP traffic to flow between an IBM i SNMP agent and an SNMP-based remote system or network management application. The SNMP agent has been changed so that responses to SNMP requests will now be sent through UDP port 161. SNMP trap messages that originate on an IBM i system will also now be sent through port 161. This change does not affect the IBM i trap manager, which will continue to receive trap messages through UDP port 162. Traps being forwarded by the IBM i trap manager will also be sent through port 161. This change also does not affect the native IBM i SNMP manager APIs which will continue to send requests through a random ephemeral port.

Configuring storage pools and disk block sizes

As the storage sizes for both storage pools and individual disk units have increased, the limits of the original SNMP design have been reached or in some cases exceeded. For example, the size that SNMP returns for an 8 TB disk unit on an IBM i system might show up as a negative number in some systems management applications, or it could be flagged as invalid. The latest enhancements to SNMP allow the configuration of block sizes to be used for returning storage size information for both storage pools and disk units. One thing to note is that using this support for disk units can cause IBM i to be non-compliant with the *RFC 1514* standard, which defines a disk unit block size as 1024 bytes. A larger block size can be configured using the new block size (BLKSIZE) parameter on the `Change SNMP Attributes (CHGSNMPA)` command. By using an appropriate block size, you can avoid confusion related to the incorrect storage size value or a disk unit being flagged as invalid by a systems management application.

GetBulk operation support

The SNMP *GetBulk* operation was introduced in SNMP version 2 (SNMPv2) and provides a method to easily get a relatively large amount of data with a single SNMP request. Although IBM i does not support SNMPv2, it does support SNMP version 3, which provides improved security and privacy for SNMP messages. The IBM i 7.1 and IBM i 7.2 agents now fully supports the GetBulk requests for SNMPv3. Additionally, in IBM i 7.2, there is a new SNMP manager API, `snmpGetbulk_v3`, that can be used to send a GetBulk request to an SNMPv3 agent.

Retrieving table information using GetBulk

Now, let's take a look at an example of a C program for an SNMP manager running on an IBM i system. The GetBulk operation is used to retrieve information about storage pools from other systems. This example assumes that SNMPv3 is already enabled and configured on both the *SNMP manager (source of the SNMP requests)* and *agent (target of the SNMP requests)*. Refer to the GetBulk source code for this example.

The following steps highlight the key operations in this example program.

1. For each agent, the program must perform SNMPv3 agent engine ID discovery between the SNMP manager program and the agent. This is done by calling the `snmpDiscover_v3()` API. If the API call is successful, SNMPv3 operations between the manager and agent can proceed. The `snmpDiscover_v3()` API returns a control block that is used by other SNMPv3 APIs for time synchronization and authentication. The control block returned is specific to an agent. Because of this, the example program issues the `snmpDiscover_v3()` API once for each agent.

Listing 1. Performing SNMPv3 agent engine ID discovery

```
host = hostList[currentHost].host;              /* Set the host name. */
cb = NULL;                                      /* Set authentication CB to NULL. */
rc = snmpDiscover_v3(host, timeout, &cb);       /* Perform SNMP Agent Discovery */
if (rc != API_RC_OK) {                          /* Check whether discovery failed. */
    printf("Discovery failed with rc=%d\n", rc);   /* Output reason code. */
    goto Cleanup;                               /* Cleanup and end. */
} else;
```

2. The program must determine the number of storage pools that are configured for the SNMP agent. To do this, the `snmpGetnext_v3()`API is used to examine the host resources' storage table, also known by the standard name of `hrStorageTable`. The first entry in the table, `hrStorageIndex`, provides the program with the information needed to read selected information from the table. The `snmpGetnext_v3()` API is used in a loop until all the table entries corresponding to `hrStorageIndex` have been accessed. Counting these entries provides the number of rows in `hrStorageTable`.

Listing 2. Determining the number of table entries

```
maxReps = 0;                                         /* Initialize maximum repetitions. */
do {                                                 /* Loop until finished. */
    pdu->varbind->val_len = API_MAX_VALUE_SIZE + 1;
    rc = snmpGetnext_v3(pdu, host, timeout, user, &cb); /* Perform GetNext operation. */
    maxReps++;                                       /* Incremment maximum repetitions.*/
} while ((memcmp(tblIndexOID, pdu->varbind->oid, strlen(tblIndexOID)) == 0) &&
    (rc == API_RC_OK));                              /* Check whether we are still */
                                                     /* processing index entries. */
```

3. To prepare for performing the `GetBulk` operation, the program sets up the necessary *protocol data unit (PDU)* structures and *variable bindings (varbinds)*. The program gets several pieces of general system information as well as specific information from each row of `hrStorageTable`. For this example, it adds the `varbinds` to get the system name (`sysName`) and the network management system (SNMP server) up-time (`sysUpTime`). The `GetBulk` request that the program builds will result in each of these varbinds only being processed once. These make up the *non-repeaters* section of the `GetBulk` PDU.
4. Next, the program sets up the `varbinds` for retrieving the storage pool information from `hrStorageTable`. It will add varbinds for the description of the storage pool (`hrStorageDesc`), the size in bytes of an allocation unit (`hrStorageAllocationUnits`), the storage size in allocation units (`hrStorageSize`), and the amount of storage used in allocation units (`hrStorageUsed`). These `varbinds` make up the *maximum-repetitions* portion of the `GetBulk` PDU.
5. Next, the program calculates the number of `varbinds` that are expected to be returned in the response to the `GetBulk` request. The program then needs to allocate all the varbind structures necessary for the `GetBulk` response. For a single GetBulk request, IBM i supports returning up to 512 varbinds. The IBM i system also restricts the size of the response data packet to 32 KB. On a large system, a table such as the host resources' device table (`hrDeviceTable`) can contain more than 512 entries. Multiple `GetBulk` requests would be necessary to retrieve an entire table in some cases.
6. All the information for `hrStorageTable` will then be retrieved with a single `GetBulk` request sent by the `snmpGetbulk_v3()` API.

Listing 3. Setting up varbinds and performing GetBulk

```
/* Add a varbind to retrieve the system name. This will only be retrieved once
   and is part of the "non-repeaters" varbind section of the GetBulk input PDU.
*/

varBNbr = AddVarbind((snmppdu **) &bulkpdu, "1.3.6.1.2.1.1.5");
```

```
/* Add a varbind to retrieve the network management up-time. This will be
   retrieved once and is part of the "non-repeaters" varbind section of the GetBulk input PDU.
*/

varBNbr = AddVarbind((snmppdu **) &bulkpdu, "1.3.6.1.2.1.1.3");
bulkpdu->non_repeaters = varBNbr;                                /* Set number of non-repeaters. */

/* These statements add the OIDs to the GetBulk PDU which will allow us to
   retrieve the information from the host resources storage table (hrStorageTable) for the
   system's storage pools. These will be retrieved repeatedly and make up the
   "maximum-repetitions" section of the PDU
*/

varBNbr = AddVarbind((snmppdu **) &bulkpdu, "1.3.6.1.2.1.25.2.3.1.3"); /* Storage Description */
varBNbr = AddVarbind((snmppdu **) &bulkpdu, "1.3.6.1.2.1.25.2.3.1.4"); /* Allocation units */
varBNbr = AddVarbind((snmppdu **) &bulkpdu, "1.3.6.1.2.1.25.2.3.1.5"); /* Storage size */
varBNbr = AddVarbind((snmppdu **) &bulkpdu, "1.3.6.1.2.1.25.2.3.1.6"); /* Storage used */
bulkpdu->pdu_type = GETBULK_PDU_TYPE;                              /* Initialize the PDU type */
bulkpdu->maximum_repetitions = maxReps;                            /* Set maximum repetitions */
respNbr = ((varBNbr - bulkpdu->non_repeaters) * bulkpdu->maximum_repetitions) +
    bulkpdu->non_repeaters;                                        /* Calculate the number of varbinds
                                                                       in the response PDU. */
MakeResponsePDUSpace(&pdu, respNbr);                              /* Make a response PDU space with
                                                                       calcuated number of varbinds. */
rc = snmpGetbulk_v3(bulkpdu, host, timeout, user, &cb, pdu);      /* Perform the GetBulk. */
```

7. After a successful call to `snmpGetbulk_v3()`, the program walks through the varbinds in the response PDU and displays the information that was returned. It also uses the information returned to calculate and display the storage size in bytes and the storage used in bytes.
8. The program then cleans up all the storage it allocated and the authentication control block that was allocated for processing the current agent. The `snmpFreeAuthCB_v3()` API must be used to free the authentication control block.
9. The program then processes the next agent in its list.

Example 1 shows the output of the program.

Example 1. Storage pool information from GetBulk

```
System Name: LOCALHOST
Up-time (in seconds): 62939

Pool           Block Size    Size         Size           Size Used
Description    in Bytes      in Blocks    in Bytes       in Bytes
System ASP     4096          39141480     160323502080   46772875264
RAM            4096          145441       595726336      576901120
RAM            4096          1135368      4650467328     3070193664
RAM            4096          15728        64421888       12288
RAM            4096          276327       1131835392     144236544

System Name: BIGSYSTEM
Up-time (in seconds): 62900

Pool             Block Size    Size         Size             Size Used
Description      in Bytes      in Blocks    in Bytes         in Bytes
System ASP       8192          1228421942   10063232548864   6161258889216
User ASP         8192          9321270      76359843840      5750784
User ASP         8192          335544320    2748779069440    351518720
Independent ASP  8192          120193024    984621252608     8866537472
Independent ASP  8192          51511296     421980536832     124952576
RAM              8192          6442790      52779335680      34723086336
RAM              8192          96346898     789273788416     380127264768
RAM             8192          25787520     211251363840    880140288
RAM             8192          1298759      10639433728     119259136
```

One thing to note about the output is that each system has a different unit or block size retrieved for `hrStorageAllocationUnits.` The `CHGSNMPA BLKSIZE(8192 *DFT)` command was run on the system BIGSYSTEM before running the `GetBulk` operation. This caused 8192 to be returned for `hrStorageAllocationUnits` instead of letting the system determine the block size. Before running the `CHGSNMPA` command, the output from the example program produced invalid results for `hrStorageSize` because of the large size of the system auxiliary storage pool (ASP) on BIGSYSTEM. Even with a 4096 block size, the total number of blocks might not fit in a 4-byte integer. Changing the block size allowed valid information to be retrieved for `hrStorageSize` which the program could then use to calculate the actual storage size in bytes. One last thing to note is that the SNMP server must be ended and restarted in order for changes to the block size to become effective. Example 2 shows the output before changing the storage pool block size. The size both in blocks and bytes show incorrectly as a negative number:

Example 2. Partial GetBulk results with block size too small

```
System Name: BIGSYSTEM
Up-time (in seconds): 11

Pool           Block Size    Size         Size            Size Used
Description    in Bytes      in Blocks    in Bytes        in Bytes
System ASP     4096          -1838123412  -7528953495552  6167716663296
```

Additional information

Required PTFs

The following PTFs enable the enhancements described in this article:

- IBM i 7.1 - PTF SI55745.
- IBM i 7.2 - PTFs SI55537, SI55539, SI55766, SI55787, and SI55966.

Setting up SNMPv3 on an IBM i system

You need to perform the following steps to set up SNMPv3 on an IBM i system.

1. Check whether SNMPv3 is already enabled on your system by entering the `CHGSNMPA` command on the IBM i command line. Locate the Allow SNMPv3 Support (`ALWSNMPV3`) parameter and verify that it is set to `*YES`.
2. If not, change it to *`YES` and also change the SNMP engine identifier (`SNMPENGID`) to `*SYSGEN`.
3. End the SNMP server by running the `ENDTCPSVR *SNMP` command.
4. Configure the SNMPv3 users by running the Add User for SNMP (`ADDUSRSNMP`) command. User names and passwords that you configure with ADDUSRSNMP are case sensitive and must match exactly between the SNMP agent and manager. In addition, the authentication and privacy protocols must also match exactly.
5. If you are using the IBM i SNMPv3 manager APIs, you must specify a key type (`KEYTYPE`) parameter value of `*NONLOCALIZED`. This is necessary because the SNMP manager uses the

agent's SNMP engine ID to *localize* the authentication and privacy keys when performing encryption and decryption operations.
6. After adding all your SNMPv3 users, you can start the SNMP server by running the `STRTCPSVR *SNMP` command.

Due to differences in SNMPv3 manager implementations, it might be necessary to add an environment variable to make a change in the way the IBM i agent performs validation checks during the initial communication with an SNMPv3 manager. If an SNMPv3 manager application is timing out or reporting a time synchronization error while attempting to establish the initial SNMPv3 communications with the IBM i agent, add this environment variable using the following command:

```
ADDENVVAR ENVVAR(QIBM_SNMPV3_AUTH) VALUE('1') LEVEL(*SYS)
```

After running this command, the SNMP server must be ended and then restarted. In addition, it might be necessary to end and then restart the SNMP manager application.

Summary

The IBM i system now offers new capabilities that include the ability to more easily configure firewalls by sending responses through UDP port 161, to eliminate SNMP manager reported errors by configuring storage pool and disk unit block sizes, and to improve SNMP manager performance through the use of the `GetBulk` request.

References

- For information about the IBM i SNMPv3 Manager APIs, see the Simple Network Management Protocol (SNMP) Manager APIs topic in the IBM i Knowledge Center.
- The Host Resources MIB is described by RFC1514 - Host Resources MIB
- The GetBulk operation is described by RFC1448 - Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)