SNMP Training

Short introduction to SNMP

SNMP is used to transfer data from a managed device called Agent to the managing device, called Manager. The protocol permits active operations to modify and change configurations, by changing different variables. SNMP uses UDP as a transport protocol.

Since the Version 3 of the protocol the following Operations are available:

GetRequest

This Operation is used by a manager to retrieve values of variables from an agent.

GetNextRequest

A Manager-to-agent request, which returns the lexicographically next variable in the MIB. This operation is useful to walk complete MIBs or read rows of a table (see GetBulkRequest).

Response

This is a response to a query from a Manager.

SetRequest

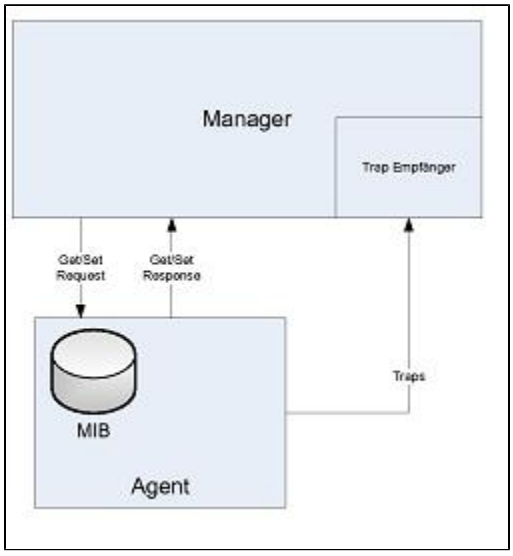
A Manager-to-agent request to change values of variables.

GetBulkRequest

This optimized version of a GetNextRequest is a request for multiple iterations of a GetNextRequest, which is useful to retrieve the rows of a table.

InformRequest

This request is an asynchronous notification from a manager to a manager.



Picture 1: SNMP Communication

Traps

Traps are asynchronous notifications from an agent to a manager.

Every SNMP instruction affects only one value. This implies that every variable has to be set separately. Before the activation of a certain protocol all important variables have to be set.

Then the protocol can be activated with the value RowStatus. SNMP uses MIBs to get access to values.

A Management Information Base (MIB) is a virtual database which is used to manage devices in a network. Objects in a MIB are defined using the Structure of management Information (SMI). The database is a structured tree and the entities are addressed using **object identifiers** (OID). This **OID** uniquely identifies a managed object and is a chain of numbers separated by a point. For Example: 1.3.6.1.2.1.1.1.

For a managed object two types exist:

- Scalar (Single Instance)
- Tabular (Multiple related instances that are grouped in a table)

With these variables the behaviour of a network protocol can be influenced. Variables can be:

ACCESS & MAX-ACCESS values	SMIv1	SMIv2
read-only	yes	yes
read-write	yes	yes
not-accessible	For table & entry objects	yes
accessible-for-notify	no	yes
read-create	no	yes
write-only	obsolete	no

Table 1: SNMP Access Control

After a SNMP query the following errors can occur:

Values	Description
noError (0)	No errors occurred during transmission
tooBig (1)	The Agent could not place the results in a single Message.
noSuchName (2)	There is no such variable

badValue (3)	Either a syntax or value error
	`$ snmpset test.net-snmp.org  snmpSetSerialNo.0 i 9999`
	`   Error in packet.`
	`   Reason: (badValue) The value given has the wrong type or length`
	`   Failed object: SNMPv2-MIB::snmpSetSerialNo.0`
	`$ snmpset test.net-snmp.org -Ir snmpSetSerialNo.0 s "How To Be Topp"`
	`   Error in packet.`
	`   Reason: (badValue) The value given has the wrong type or length`
	`   Failed object: SNMPv2-MIB::snmpSetSerialNo.0`

readOnly (4)	The variable is not allowed to change
genErr (5)	An error other than those listed here occurred
noAccess (6)	The variable is not accessible

wrongType (7)	The type of the value inside the PDU is inconsistent with the type required for the variable
	`$ snmpset -v 2c test.net-snmp.org -Ir snmpSetSerialNo.0 s "uterly wet"`
	`   Error in packet.`
	`   Reason: (wrongType) The set datatype does not match the data type the agent expects`
	`   Failed object: SNMPv2-MIB::snmpSetSerialNo.0`

wrongLength (8)	The length of the value insided the PDU inconsistent with the length required for the variable

wrongEncoding (9)	The values contains an encoding inconsistent with the required encoding

wrongValue (10)	The value of the inside the PDU inconsistent with the range required for the variable
	`$ snmpset -v 2c test.net-snmp.org snmpSetSerialNo.0 i 9999`
	`Error in packet.`
	`Reason: (wrongValue) The set value is illegal or unsupported in some way`
	`Failed object: SNMPv2-MIB::snmpSetSerialNo.0`

noCreation (11)	The value cannot be assigned to the variable.
inconsistentValue (12)	The value is inconsistent with values of other managed objects.
resourceUnavailable (13)	Assigning the value to the variable requires allocation of resources that are currently unavailable.
commitFailed (14)	No validation errors occurred, but no variables were updated.
undoFailed (15)	No validation errors occurred. Some variables were updated because it was not possible to undo their assignment.
authorizationError (16)	An authorization error occurred.
notWritable (17)	The variable exists but the agent cannot modify it.
inconsistentName (18)	The variable does not exist; the agent cannot create it because the named object instance is inconsistent with the values of other managed objects.

Table 2: SNMP Errors

Working with scalar objects you have to add a .0 after get.

wrong:

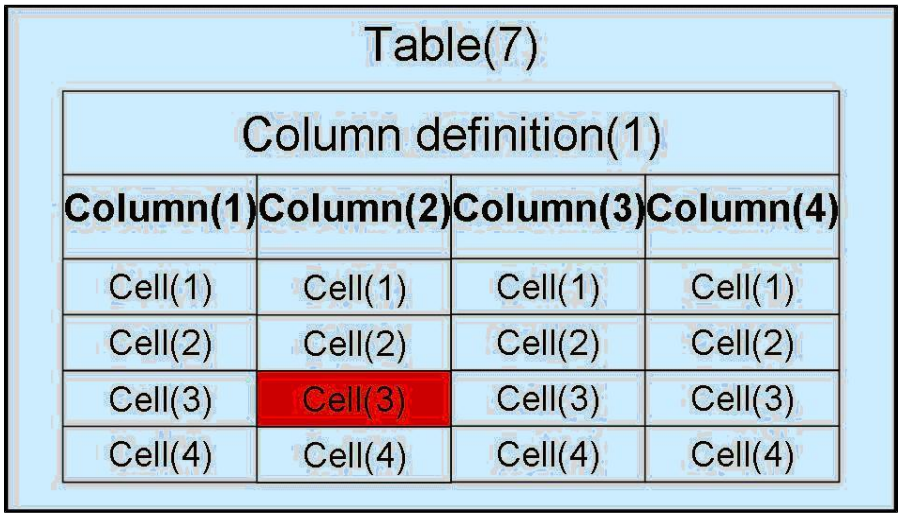
```
snmpget -v 3 -Os -l authPriv -u admin -a MD5 -A privateprivate -x DES -X
privateprivate 192.168.33.99 HM2-PLATFORM-ROUTING-
MIB::hm2AgentSwitchIpRoutingMode hm2AgentSwitchIpRoutingMode = No Such Instance
currently exists at this OID
```

right:

```
snmpget -v 3 -Os -l authPriv -u admin -a MD5 -A privateprivate -x DES -X
privateprivate 192.168.33.99 HM2-PLATFORM-ROUTING-
MIB::hm2AgentSwitchIpRoutingMode hm2AgentSwitchIpRoutingMod.0
```

RowStatus and SNMP MIB Tables

All Protocols are organized in conceptual tables. SNMP has no real tables but with OID prefixes and Indexes a table can be “simulated”. The rows in the tables can consist of one or more objects and a table has one or more indices.



Picture 2: MIB Table

If the red marked cell has to be addressed, the correct OID has to be:

BaseOID.7.1.2.3

Rows can be identified by the instance number. The Table itself has its own OID which is the root of the table. Each row has its own instance number. Another example is the ipAddrTable. The index is here the ipAdEntAddr. Other Objects of the table can be accessed via this index.

ipAddrTable is the Tablename and the column definition is the ipAddrEntry.

An entire table can be retrieved by performing repeated SNMP GETNEXT or GETBULK Operations. Also the row data can be retrieved with a given index value.

Polling of a table is an important feature, as all values will be updated automatically. Polling is a request-response interaction between the SNMP Manager and Agent. It is also possible to update only a specific value in a table, with a known row and column index.

```
ipAddrTable.ipAddrEntry.ipAdEntAddr.127.0.0.1                = 127.0.0.1

ipAddrTable.ipAddrEntry.ipAdEntAddr.10.1.1.54                = 10.1.1.54

ipAddrTable.ipAddrEntry.ipAdEntIfIndex.127.0.0.1              = 1
ipAddrTable.ipAddrEntry.ipAdEntIfIndex.10.1.1.54              = 2
ipAddrTable.ipAddrEntry.ipAdEntNetMask.127.0.0.1              = 255.0.0.0

ipAddrTable.ipAddrEntry. ipAdEntNetMask .10.1.1.54            = 255.0.0.0

….
```

ipAddrTable is the Tablename and the column definition is the ipAddrEntry.

An entire table can be retrieved by performing repeated SNMP GETNEXT or GETBULK Operations. Also the row data can be retrieved with a given index value.Polling of a table is an important feature, as all values will be updated automatically. Polling is a request-response interaction between the SNMP Manager and Agent. It is also possible to update only a specific value in a table, with a known row and column index.

To create or delete a row the RowStatus is used. The RowStatus has six defined values:

- **active (1)** The row is available for usage
- **notInService (2)** The row exists but is not available for usage
- **notReady (3)** The row exists but one or more variables are not instantiated
- **createAndGo (4)** Creates a row which is available for usage after the creation

- **createAndWait (5)** Creates a row which is not available for usage
- **destroy (6)** Destroys the row with all instances associated

An existing row can be in the states “active”, “notInService”, “notReady”. If all values of a row are filled and the manager wishes to create a new row, the State should be “createAndGo”. But if not all values are given, the State should be “createAndWait”. After “createAndWait” the Row is “notReady” until all values are filled in. After all values are set, the State changes to “active”. It is possible to give default values to objects. If a Row will be created and the values are not filled in by the Management Application, these default values will be assigned to the objects. Not all objects are suited for a default value (e.g. Ringports).

The RowStatus defines the state of the database entry of a certain protocol. If the value returns “active” the protocol is running correctly. The Objects cannot be modified once the RowState is “active” except it is specified.

The RowStatus is a way to predefine an example set of values and commit them all at once. Only if all values are configured correct, the configuration can be activated.

Dependencies between tables are also possible. One Table is the base and the other is the augmenting Table. An example is a MIB which imports another MIB and shares the same table.

Summary RowStatus and SNMP MIB Tables

- Each table contains zero or more rows and each row contains one or more objects.
- Dynamic row creation should use a RowStatus
- Object changes are always allowed if the RowStatus is not active and may be allowed if the RowStatus is active
- If not all non-defaultable values are filled correctly the RowStatus is not allowed to be active
- There must be a StorageType Value or a DESCRIPTION Clause which has to specify what happens after a restart
- There is no need for a RowStatus if the base row of an conceptual row that AUGMENTS has already the RowStatus
- The DESCRIPTION clause has to specify whether the RowStatus has to be active to modify objects or not. It is useful to forbid any changes while the RowStatus is *active* to minimize the side-effects on the client-side.
- The RowStatus allows a separate activation of a configuration.

Rules for creating MIB Tables

1. It is possible to transfer multiple SNMP Instructions in one frame. It has to be ensured that the RowStatus update is the last instruction. Otherwise the RowStatus will be set before all variables are updated. If a table has more objects that fit in one PDU the RowStatus update has to be in the last PDU.
2. Also the range of a specific value has to be checked before the activation of a row. If a variable only accepts the values “1” or “2” but is set to “3” the RowStatus can’t be set to “active”. The implementation has to return an error and set the RowStatus to “notReady”.
3. Another important issue is fate sharing. This means that the state of a table depends on the state of another table. If one table is deleted, the other will be also deleted. If we have two tables that hold information that belong together, it might be the case that one has to be put in the “active” state before the other. Only if the first table is “active” the other can be activated too. If both tables are “active” the transaction can be seen as committed. This behaviour should be explicitly defined with the SMI index qualifier *AUGMENT* if the relationship is one-to-one or *INDEX* if the relationship is sparse. Otherwise the description should define what happens if one row is deleted or activated.
4. If an object needs to be set before creating a row instance an agent capabilities statement should be used with a CREATION-REQUIRES clause.
5. The state “notReady” should be returned if the row has not yet set one or more non-defaultable instance values for the row.
6. During the time when not all of the required non-defaultable objects have been set, an Agent should return a noSuchInstance error for a GET on these objects.
7. The RowStatus should not be used to activate or deactivate a configuration. There is no guarantee that the agent won’t discard the configuration. A global admin state is a better solution, which prevents the table from being aged out and deleted.
8. An Implementation has to consider how many rows, which are “notInService” or “notReady” can be created such that the resources are not exhausted.

State Diagram

The following state diagram shows the dependencies between the different states in which the RowStatus could be. The Transitions are also shown.

State	**A** **RowStatus does not exist**	**B** **RowStatus is *not ready***	**C** **RowStatus is *not*** ***inService***	**D** **RowStatus is *active***

CreateAndGo	noError ? D or inconsistent Value	inconsistent Value	inconsistent Value	inconsistent Value
CreateAndWait	noError or wrongValue (1)	inconsistent Value	inconsistent Value	inconsistent Value
active	inconsistent Value	inconsistent Value or noError ? D (2)	noError ? D (6)	noError ? D
notInService	inconsistent Value	inconsistent Value or ? C (3)	no Error ? C	no Error ? C
destroy	noError ? A	noError ? A	noError ? A	noError ? A
set any column	(4)	noError (1)	noError ? C	noError ? D (5)

Table 3: RowStatus State Diagram

Explanation:

(1) B or C depending on the information available on the agent

(2) If the PDU provides the missing values then go to D

(3) If the PDU provides the missing values then go to C

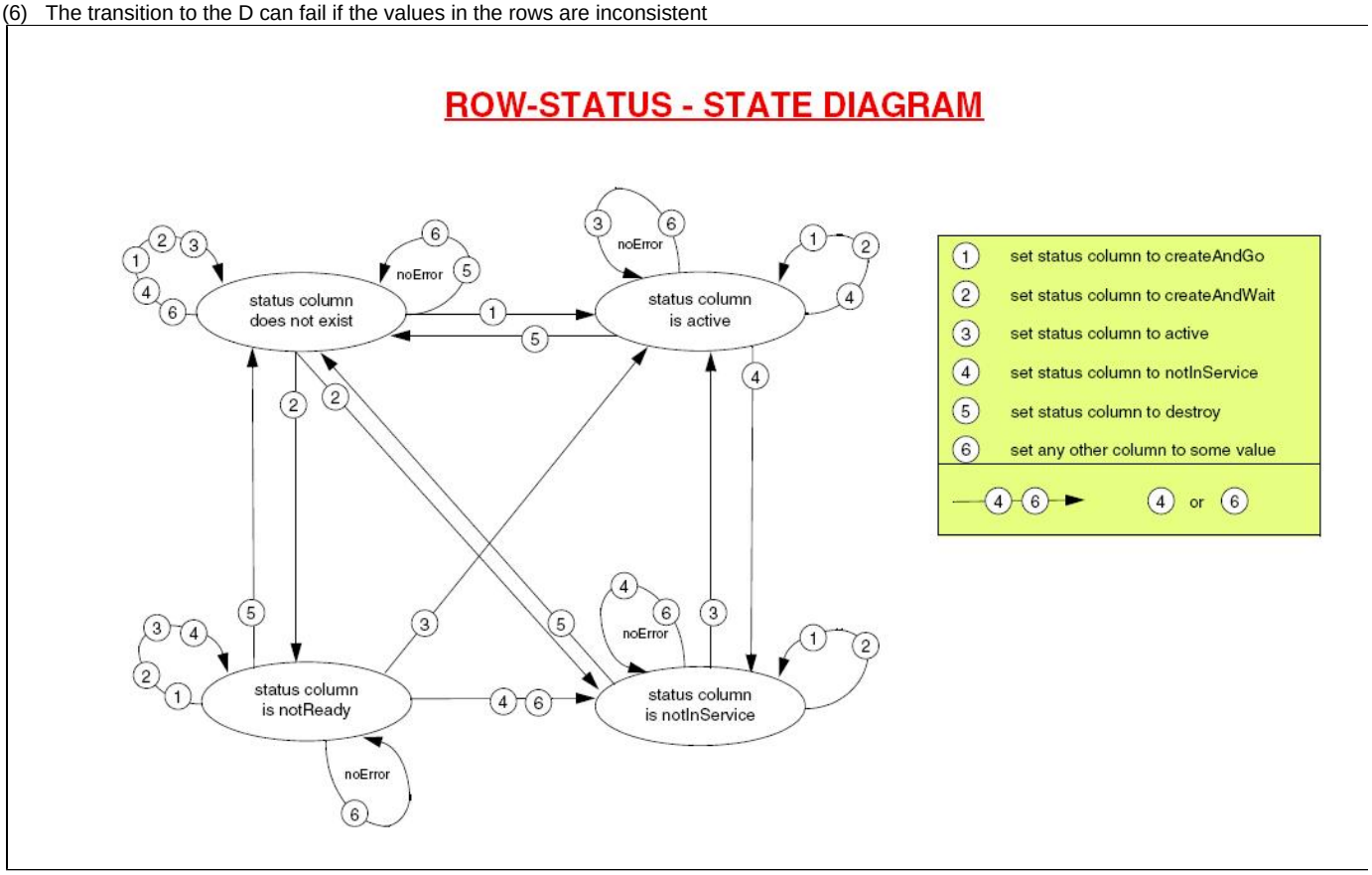
(4) Three Return Values may occur:

1. Inconsistent Name
2. Inconsistent Value
3. NoError if the agent creates the instance

If NoError is returned the RowStatus has to be changed. The State is now B or C depending on the delivered values. If inconsistentName/inconsistentValue is returned, the RowStatus remains in A

(5) Depending on the MIB definition either noError or inconsistentValue may be returned

(6) The transition to the D can fail if the values in the rows are inconsistent



Picture 3: RowStauts State Diagram (http://www.simpleweb.org/w/images/9/91/Tutorial_Slides_Smi.pdf)

MIB Design

Please refer to the following books which show how to design SNMP MIBs. The “SNMP MIB Handbook” by [Larry Walsh] shows step by step how

to build and design MIBs.

Another excellent book is “Understanding SNMP MIBs” by [David Perkins]. This book is a reference and describes all the interns of creating MIBs.

References

Books covering SNMP:

- SNMP, SNMPv2, SNMPv3 and RMON 1 and 2 0-201-48534-6 Addison-Wesley William Stallings Third Edition, 1999
- Total SNMP, Exploring the Simple Network Management Protocol 0-13-646994-9 Prentice Hall PTR Sean Harnedy Second Edition, 1998

Books covering MIBs:

- Understanding SNMP MIBs 0-13-437708-7 Prentice Hall PTR David Perkins, Evan McGinnis 1997
- SNMPv 3 and Network Management (A practical guide to) 0-13-021453-1 Prentice Hall PTR David Zeltserman Ausg. 99

Further Information:

[1] RFC2578 - Structure of Management Information Version 2 (SMIv2)

http://www.ietf.org/rfc/rfc2578

[2] RFC2579 - Textual Conventions for SMIv2

http://www.ietf.org/rfc/rfc2579

[3] RFC3512 - Configuring Networks and Devices with Simple Network Management Protocol (SNMP), Chapter 3 Designing a MIB Module

http://www.ietf.org/rfc/rfc3512

[4] RFC4181 - Guidelines for Authors and Reviewers of MIB Documents

http://www.ietf.org/rfc/rfc4181.txt

[5] A Consolidated Overview of Version 3 of the Simple Network Management

Protocol (SNMPv3)

http://tools.ietf.org/html/draft-perkins-snmpv3-overview-00

[6] Larry Walsh: SNMP MIB Handbook. Wyndham Press. 2008.

[7] David T. Perkins: Understanding SNMP MIBs. Prentice Hall. 1996

See also: http://www.dsc.ufcg.edu.br/~jacques/cursos/gr/recursos/outros/perkins.pdf