Projektdoku: SNMP-Tool

# 1. Aufgabenstellung

Zu implementieren gilt eine Software, genauer gesagt eine Konsolenapplikation in der Programmiersprache C#, die Funktionalitäten des Simple Network Management Protokolls anbietet. Diese umfassen neben dem Absenden von Get-Befehlen auch das Hören nach Trap-Meldungen. Dabei wird ausschließlich das SNMP-Protokoll v2c verwendet, da dies das Beliebteste ist.

# 2. Server aufsetzen

Im Rahmen des Projekts wurde zu Testzwecken ein Windows Server mit dem Betriebssystem Windows Server 2012 aufgesetzt. Dieser bietet in seinen Diensten das SNMP-Protokoll an.

* **Infos über den Server**
  + User: Administrator
  + Passwort: W16Abc4711
  + IP-Adresse: 10.60.45.40 (kann sich geringfügig durch DHCP ändern)

## 2.1. SNMP-Dienst aktivieren

Da der SNMP-Dienst von Haus aus nicht aktiv ist, muss dieser zunächst konfiguriert werden.

**Anleitung:**

Server Manager 🡪 Dienste hinzufügen 🡪 SNMP-Dienst 🡪 Häkchen setzen

Community String: Dieser ist standardmäßig entweder auf private oder public gesetzt. Damit unser Server nur Meldungen von unserer Anwendung annimmt, ändern wir diesen auf „ntma\_snmp“. Was dabei zu beachten ist, dass der Community-String zwischen 8 und 10 Zeichen lang sein muss.

Somit ist unser Server einsatzbereit.

# 3. C# Konsolenanwendung

Wireshark-Dump eines Referenzpakets, versendet & erstellt mittels SnmpSharpNet.dll:

Set-Request:

OID: 1.3.6.1.2.1.1.6.0

Value: „Some other value“

0000 90 2b 34 bd b9 da 00 0f fe a1 0e 8b 08 00 45 00 .+4..... ......E.

0010 00 5a 1a c7 00 00 80 11 00 00 0a 3c 2d 22 0a 3c .Z...... ...<-".<

0020 2d 23 d3 29 00 a1 00 46 6f 14 30 3c 02 01 01 04 -#.)...F o.0<....

0030 09 6e 74 6d 61 5f 73 6e 6d 70 a3 2c 02 04 61 e4 .ntma\_sn mp.,..a.

0040 5f ed 02 01 00 02 01 00 30 1e 30 1c 06 08 2b 06 \_....... 0.0...+.

0050 01 02 01 01 06 00 04 10 53 6f 6d 65 20 6f 74 68 ........ Some oth

0060 65 72 20 76 61 6c 75 65 er value

SNMP-Paket

Dieselbe SNMP-Request mit der selbst programmierten C#-Applikation:

0000 90 2b 34 bd b9 da 00 0f fe a1 0e 8b 08 00 45 00 .+4..... ......E.

0010 00 58 4a 7b 00 00 80 11 00 00 0a 3c 2d 22 0a 3c .XJ{.... ...<-".<

0020 2d 23 e5 d5 00 a1 00 44 6f 12 00 00 00 01 6e 74 -#.....D o.....nt

0030 6d 61 5f 73 6e 6d 70 00 00 00 03 c5 71 3e 55 00 ma\_snmp. ....q>U.

0040 00 00 00 00 00 00 00 31 2e 33 2e 36 2e 31 2e 32 .......1 .3.6.1.2

0050 2e 31 2e 31 2e 36 2e 30 53 6f 6d 65 4f 74 68 65 .1.1.6.0 SomeOthe

0060 72 56 61 6c 75 65 rValue

SNMP-Paket

Obwohl laut Spezifikation die ersten 4 Bytes eines SNMP-Pakets für das Versionsfeld reserviert sind („01“ bedeutet Version 2) beschreibt die Referenzsoftware das Versionsfeld anders. Weitere Abweichungen liegen in der Konvertierung der OID vor (ASN.1 Kodierung).

Get-Request erstellt und versendet via SnmpSharpNet.dll:

OID: 1.3.6.1.2.1.1.6.0

0000 90 2b 34 bd b9 da 00 0f fe a1 0e 8b 08 00 45 00 .+4..... ......E.

0010 00 4a 22 f8 00 00 80 11 00 00 0a 3c 2d 22 0a 3c .J"..... ...<-".<

0020 2d 23 e1 c7 00 a1 00 36 6f 04 30 2c 02 01 01 04 -#.....6 o.0,....

0030 09 6e 74 6d 61 5f 73 6e 6d 70 a0 1c 02 04 01 4d .ntma\_sn mp.....M

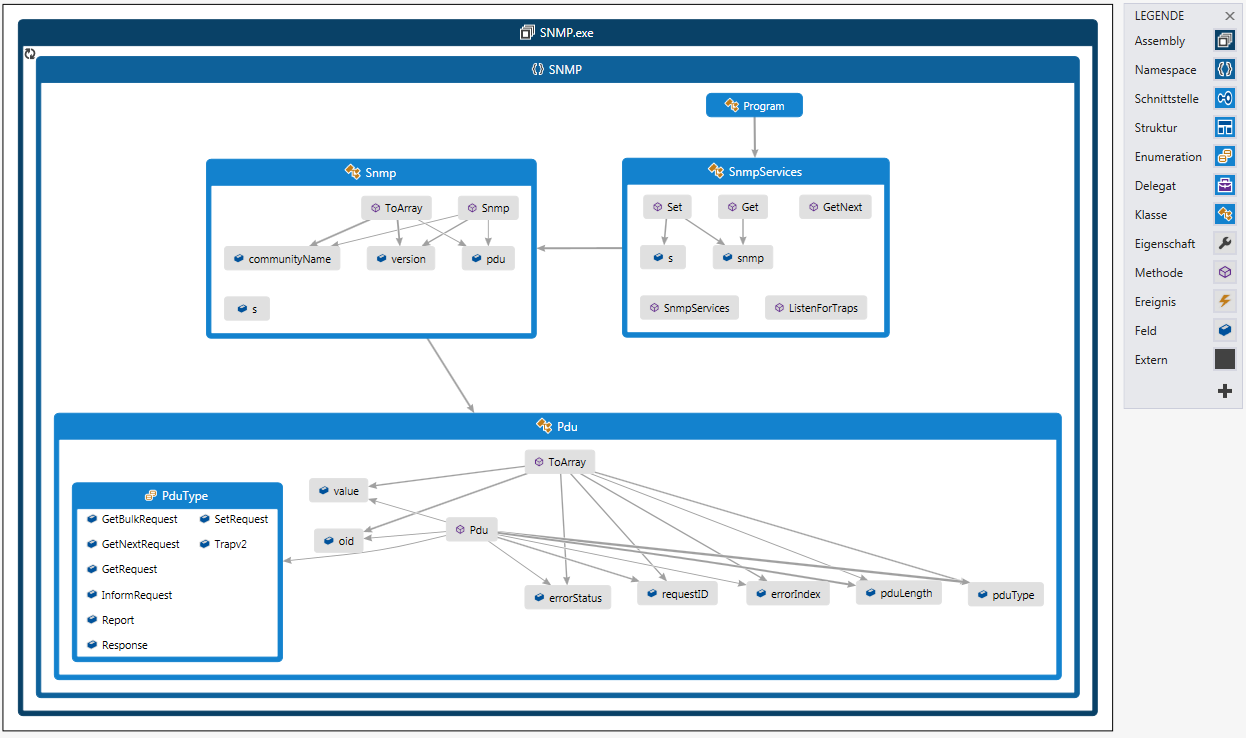
0040 1a 83 02 01 00 02 01 00 30 0e 30 0c 06 08 2b 06 ........ 0.0...+.

0050 01 02 01 01 06 00 05 00 ........

ASN.1 Kodierte OID 1.3.6.1.2.1.1.6.0

Wie an der OID der Get-Request zu sehen ist wurde diese ASN.1 kodiert.

# Klassendiagramm



Wie aus dem Klassendiagramm hervorgeht wurde versucht durch verschiedene Klassen einen modularen Aufbau der Applikation zu erreichen. Durch die Unterstützung verschiedener PDU-Formate (Protocol Data Unit) können über die Pdu-Klasse verschiedene PDUs automatisiert erstellt werden. Die PDU bildet in Kombination mit Community-Name und SNMP Version ein SNMP-Paket (Abbildung 1). Je nach Art der SNMP Anfrage kann die Länge der PDU variieren. Eine SNMP Set-Request beispielsweise enthält eine OID plus den zu setzenden Wert. Eine etwas kürzere Get-Request hingegen nur eine OID zur Abfrage eines Objekts von einem SNMP-Server

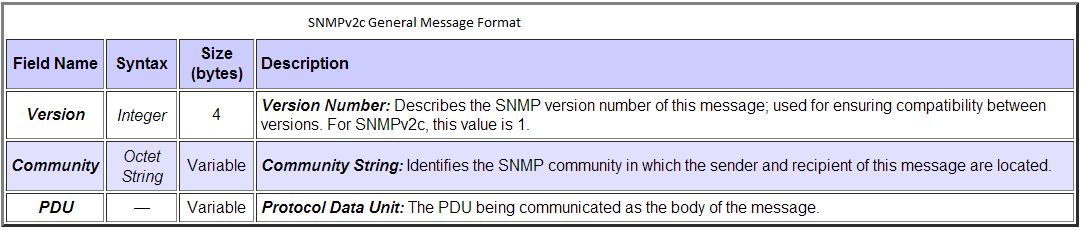


Abbildung - SNMP-Paketformat Quelle: tcpipguide.com