SNMP Library Documentation

Release 2.10

Lex Li

May 31, 2018

CHAPTER 1

Topics

Getting Started

Installing #SNMP Library on Windows

By Lex Li

This page shows you how to install #SNMP Library to your project on Windows.

In this article:

Install #SNMP Library via NuGet

The easiest way to get started building applications with #SNMP Library is to install via NuGet in the latest version of Visual Studio 2015 (including the free Community edition).

1. Install Visual Studio 2015.

   Be sure to specify that you include the Windows and Web Development.

2. Install latest NuGet Package Manager.

   This will install the latest NuGet tooling.

3. Open/create an empty Windows Forms project.

4. Install #SNMP Library NuGet packages following NuGet conventions.

   The latest package can be found at,

   - Main Library.
   - Platform Extensions (Required by release 10.0.0 only. Not required any more for 10.0.1 and above.)

Note: In 10.0.0 release, the platform extensions package is required for .NET Framework, Xamarin.iOS, and Xamarin.Android applications so as to use DES/AES encryption. This package is no longer needed in release 10.0.1 and above.

Install #SNMP Library via source code

#SNMP Library source code can be directly used in your project.

1. Download the source code from GitHub, or clone the repo directly.
2. Run `prepare.bat` on Windows (or `prepare.sh` on non-Windows platforms) to prepare the code base for compilation.
3. Open/create a empty Windows Forms project in a solution.
4. Add SharpSnmpLib.csproj (release 10.0 and above), and SharpSnmpLib.Full.csproj (10.0.0) in `SharpSnmpLib` directory to your solution.

Note: SharpSnmpLib.Android.csproj and SharpSnmpLib.iOS.csproj might be used to target Xamarin platforms for release 10.0.0. They are no longer needed for 10.0.1 release and above.

Related Resources

- An Introduction to #SNMP
- SNMP v3 Operations
- Command Line Tools
- Agent Development

Project History

By Lex Li

This article describes history of #SNMP.

In this article:

The Road to 1.0 Release

Microsoft introduced .NET Framework to developers in 2000. However, it lacks of official SNMP protocol support. Many third parties provide their solutions ever after to please their users.

Lex Li did an evaluation report in 2008 for his project at Cisco [1]. He was quite satisfied with some of the commercial solutions, but felt that there should be a need to have an open source implementation. Based on the code base from Malcolm Crowe [2], Lex was able to start a new open source project called #SNMP Library [3] in April 2008.

It was challenging a task to design an easy-to-use SNMP API, and also difficult a mission to study the SNMP protocol details. Remember the facts that Lex just graduated in 2007, and only had one year experience on programming serious projects. But luckily all initial problems (such as message parsing) were solved via endless experiments, and

[1] https://blog.lextudio.com/2007/12/product-review-snmp-libraries-for-net-evaluation-report/
[2] http://cis.uws.ac.uk/crow-ci0/
[3] https://sharpsnmplib.codeplex.com

even features such as MIB parsing was developed in an ugly way [4] . By using TDD approach, every major features are covered.

Near the final release of 1.0, two more products were added (the MIB browser and MIB compiler), so the project was renamed to #SNMP Suite.

The 1.0 release (code name UnicornHorn) was finally released in July 2008 [5] , four months after the project launch. It features basic SNMP v1 and v2c support.

SNMP v3 Support and Beyond

Steve Santacroce joined the project development in August 2008 [6] , soon Lex decided to leave from Cisco. Lex moved to Microsoft in October 2008. The efforts required to drive #SNMP forward were so overwhelming that Lex had to drop another open source project [7] .

The next major release of #SNMP Suite was 1.5 (code name TwinTower) in January 2009 [8] . It provides a refined API set, and many bug fixes over the initial release. Three months later, the 2.0 release (code name CrossRoad) was released in April 2009 [9] , which finished all tasks except SNMP v3 support.

SNMP v3 is a monster to conquer, because it introduces fundamental changes to the message format (and different agent side behaviors when we attempted to implement an agent prototype). Thus, many API elements have to be completely rewritten. But things became easier as another open source SNMP implementation named SNMP#NET was published by Milan Sinadinovic [10] . By reusing the encryption code, #SNMP Library soon started to support SNMP v3 packets. The 3.0 release (code name Trident) was released in August 2009 with initial SNMP v3 support.

The next major task was to implement an SNMP agent that can process incoming requests. Fortunately Lex was working on ASP.NET/IIS at Microsoft at that time, so he reused many ideas he learned from ASP.NET request pipeline and designed a similar pipeline for SNMP messages [11] . The 4.0 release (code name SquareRoot) shipped the initial result in March 2010 with SNMP v1 and v2c message support [12] . SNMP v3 message support only arrived in 5.0 release (code name CatPaw) [13] in May 2010. Mono support was also included for the first time as Lex revised DockPanel Suite [14] .

The 6.0 release (code name HoneyCell) in November 2010 was the first major release that ships no big change. #SNMP Suite had become mature then.

Later Releases and License Changes

The 7.0 release (code name BigDipper) was released in October 2011 [15] was mainly a bug fix release.

The 8.0 release (code name TritonMate) was released in April 2013 [16] . It featured a new compiler based on ANTLR [17] , which was under BSD 3 Clause. This indicated a move to more permissive licenses. The SNMP pipeline code was changed to MIT/X11 in the same release [18] . After the final release, the code base had been updated to support Xamarin's mobile platforms.

[4] https://blog.lextudio.com/2008/05/snmp-design-build-my-own-lexer/
[5] https://blog.lextudio.com/2008/07/snmp-design-unicornhorn-and-known-issues/
[6] https://blog.lextudio.com/2008/08/snmp-design-joint-forces/
[7] https://blog.lextudio.com/2008/11/candycan-opener-bad-news-at-prime-time/
[8] https://blog.lextudio.com/2009/01/snmp-design-here-comes-1-5-final-release-twintower/
[9] https://blog.lextudio.com/2009/04/snmp-design-shipping-the-package-of-crossroad/
[10] https://blog.lextudio.com/2009/05/trident-sign-another-open-source-snmp-library-via-c/
[11] https://blog.lextudio.com/2010/11/honeycell-drops-snmp-pipeline-and-our-agent-demo/
[12] https://blog.lextudio.com/2010/03/squareroot-puzzle-4-0-final-kick-off/
[13] https://blog.lextudio.com/2010/05/catpaw-rumors-release-notes-for-5-0-release/
[14] https://blog.lextudio.com/2010/05/dockpanel-suite-tip-5-we-could-go-mono/
[15] https://blog.lextudio.com/2011/10/bigdipper-light-rtm-post/
[16] https://blog.lextudio.com/2013/04/tritonmate-words-8-0-release/
[17] https://blog.lextudio.com/2012/02/tritonmate-words-upcoming-change-to-sharpsnmplib-mib-dll-license/
[18] https://blog.lextudio.com/2012/04/tritonmate-words-license-change-on-snmp-engine-support/

A serious bug in SNMP v3 was found and fixed [19] so that affected releases were updated with patches.

A significant license change was announced too, which released the Library code under MIT/X11 [20], which means all source code then was released under permissive licenses.

Following the license change, #SNMP Pro was announced, which forms a group of commercial products from LeXtudio [21] . Many issues once reported to the open source MIB compiler were finally fixed in the Pro edition by the new compiler design.

The open source project was renamed back to #SNMP Library. Its latest release 8.5 was published in February 2015 [22] . This release featured full support for Windows, OS X, Linux, and Xamarin platforms.

The 9.x release in 2016 is code named LordGate [23] .

The 10.x release will fully support .NET Standard.

Source Code License for Releases

By Lex Li

This article describes what open source licenses are used for the source code of major releases of #SNMP.

In this article:

Release 10.0 and Above

Assembly Name	Source Code License
SharpSnmpLib.dll	MIT/X11
SharpSnmpLib.Full.dll (10.0.0 only)	MIT/X11
SharpSnmpLib.Android.dll (10.0.0 only)	MIT/X11
SharpSnmpLib.iOS.dll (10.0.0 only)	MIT/X11
snmpd.exe	MIT/X11
All other samples	public domain

Release 8.5 and 9.x

Assembly Name	Source Code License
SharpSnmpLib.Portable.dll	MIT/X11
SharpSnmpLib.Full.dll	MIT/X11
SharpSnmpLib.Android.dll	MIT/X11
SharpSnmpLib.iOS.dll	MIT/X11
snmpd.exe	MIT/X11
All other samples	public domain

[19] https://blog.lextudio.com/2012/12/tritonmate-words-story-on-rfc-3414-support/
[20] https://blog.lextudio.com/2013/01/tritonmate-words-important-change-on-snmp-library-license/
[21] https://blog.lextudio.com/2013/04/tritonmate-words-the-upcoming-snmp-pro-editions/
[22] https://blog.lextudio.com/2015/02/snmp-pro-release-1-1-is-out/
[23] https://blog.lextudio.com/2015/05/tritonmate-words-snmp-9-0-plan/

Release 8.0 and Below

Assembly Name	Source Code License
SharpSnmpLib.dll	MIT/X11 (8.0), Lesser GPL (below 8.0)
SharpSnmpLib.Mib.dll	BSD 3 Clause (7.5 and above), Lesser GPL (below 7.5)
SharpSnmpLib.Optional.dll	Removed in 8.0, Lesser GPL (below 8.0)
Browser.exe	MIT/X11
Compiler.exe	MIT/X11
snmpd.exe	MIT/X11
All other samples	public domain

Dependencies

#SNMP uses many other open source projects, and their licenses are described below.

DockPanel Suite (MIT/X11) (c) Copyright 2007 Weifen Luo and other contributors http://dockpanelsuite.com

System.Tuples (MIT/X11) (c) Copyright 2010 Adis Hamzic https://systemtuples.codeplex.com

Mono.Options and Mono class library source files (MIT/X11) (c) Copyright 2008 Novell (c) Copyright 2009 Federico Di Gregorio (c) Copyright 2012 Xamarin Inc https://github.com/mono/mono

RemObjects Mono Helpers (Lesser GPL) (c) Copyright 2010-2015 RemObjects Software https://github.com/remobjects/monohelpers

Microsoft Unity (MS-PL) (c) Copyright Microsoft https://unity.codeplex.com

Apache log4net (Apache 2.0) (c) Copyright 2004-2015 The Apache Software Foundation http://logging.apache.org/log4net/

SharpDevelop TextEditor Control (Lesser GPL) (c) Copyright 2014 AlphaSierraPapa for the SharpDevelop team https://sharpdevelop.codeplex.com

ANTLR C# runtime (BSD 3 Clause) (c) Copyright 2010 Terence Parr https://www.antlr3.org

Office 2007 ToolStrip Renderer (custom license) (c) Copyright 2006 Phil. Wright http://www.codeproject.com/Articles/16666/Office-2007-ToolStrip-Renderer

Crad's Actions for Windows Forms (CPL) (c) Copyright 2006 Marco De Sanctis (c) Copyright 2012 Lex Li https://github.com/lextm/actionlistwinforms

Related Resources

- MIT/X11 License
- Lesser GPL
- MS-PL
- Apache 2.0
- BSD 3 Clause
- CPL

Export Restrictions

By Lex Li

This page shows you information about export restrictions.

Background

The #SNMP Library is open source software that contains strong encryption. Specifically, it has the ability to encrypt and decrypt SNMPv3 protocol. Its primary distribution point is in the United States, and subsequently falls under U.S. encryption export regulations.

To the best of our knowledge, #SNMP Library falls under ECCN 5D002 and qualifies for license exemption TSU under Section 734.3(b)(3) of the EAR. There are no U.S. regulations that prohibit you from downloading #SNMP Library.

References

Frank Hecker has written a detailed explanation of Mozilla's ECCN . It applies to other open source software, including #SNMP Library.

Addendum: Prior to January 7, 2011, downloading #SNMP Library in Cuba, Iran, North Korea, Libya, Sudan, and Syria was prohibited. On January 7, 2011 the U.S. Bureau of Industry and Security removed this restriction .

The information above follows Wireshark's claim . Consult a lawyer if you have any questions.

Tutorials

An Introduction to #SNMP

By Lex Li

This page shows you the basics about #SNMP. Basic SNMP operations (GET, SET and so on) are translated to #SNMP function calls.

In this article:

GET Operation

The following code shows how to send an SNMP v1 GET message to an SNMP agent located at `192.168.1.2` and query on OID `1.3.6.1.2.1.1.1.0`,

```
var result = Messenger.Get(VersionCode.V1,
                           new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161),
                           new OctetString("public"),
                           new List<Variable>{new Variable(new ObjectIdentifier("1.3.6.1.2.1.1.1.0"))
                           60000);
```

This operation will time out if no reply is received after 60 seconds (1 minute), and throw an exception (`TimeoutException`). If any error occurs, an `ErrorException` can be caught. All #SNMP exceptions are derived from `SnmpException`.

The result returned is a list that matches the list of `Variable` objects sent. The `Variable` in this list contains the value of the OID.

SET Operation

The following code shows how to send an SNMP v1 SET message to an SNMP agent located at `192.168.1.2` and set the value of OID `1.3.6.1.2.1.1.6.0` to `"Shanghai"`,

```
var result = Messenger.Set(VersionCode.V1,
                           new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161),
                           new OctetString("public"),
                           new List<Variable>{new Variable(new ObjectIdentifier("1.3.6.1.2.1.1.6.0"),
                           60000);
```

GET-NEXT Operation

The following code shows how to send an SNMP v1 GET-NEXT message to an SNMP agent located at `192.168.1.2` and query on OID `1.3.6.1.2.1.1.1.0`,

```
GetNextRequestMessage message = new GetNextRequestMessage(0,
                                                          VersionCode.V1,
                                                          new OctetString("public"),
                                                          new List<Variable>{new Variable(new Object
ISnmpMessage response = message.GetResponse(60000, new IPEndPoint(IPAddress.Parse("192.168.1.2"), 16
if (response.Pdu().ErrorStatus.ToInt32() != 0)
{
    throw ErrorException.Create(
        "error in response",
        IPAddress.Parse("192.168.1.2"),
        response);
}

var result = response.Pdu().Variables;
```

GET-BULK Operation

The following code shows how to send an SNMP v2 GET-BULK message to an SNMP agent located at `192.168.1.2` and query on OID `1.3.6.1.2.1.1.1.0`,

```
GetBulkRequestMessage message = new GetBulkRequestMessage(0,
                                                          VersionCode.V2,
                                                          new OctetString("public"),
                                                          0,
                                                          10,
                                                          new List<Variable>{new Variable(new Object
ISnmpMessage response = message.GetResponse(60000, new IPEndPoint(IPAddress.Parse("192.168.1.2"), 16
if (response.Pdu().ErrorStatus.ToInt32() != 0)
{
    throw ErrorException.Create(
        "error in response",
        IPAddress.Parse("192.168.1.2"),
        response);
}

var result = response.Pdu().Variables;
```

Walk Operation

Walk is not an atomic operation. That means, it utilizes several GET-NEXT (SNMP v1 walk) or GET-BULK (v2 and above). The following code shows how to perform walk on an SNMP agent located at `192.168.1.2` starting at `1.3.6.1.2.1.1`,

```
var result = new List<Variable>();
Messenger.Walk(VersionCode.V1,
               new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161),
               new OctetString("public"),
               new ObjectIdentifier("1.3.6.1.2.1.1"),
               result,
               60000,
               WalkMode.WithinSubtree);
```

The result returned contains a list of all available OIDs (as `Variable`) in this SNMP agent that under tree node of `1.3.6.1.2.1.1`.

#SNMP supports two walk modes, `Default` and `WithinSubtree`. The former ends the WALK operation at the end of MIB view, while the latter ends at the end of the subtree of initial OID.

`Messenger.Walk` is built upon GET-NEXT operations. Note that `Messenger.BulkWalk` should be used if the device supports SNMP v2, as it is built upon GET-BULK operations and provide better performance.

```
var result = new List<Variable>();
Messenger.BulkWalk(VersionCode.V2,
                   new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161),
                   new OctetString("public"),
                   new ObjectIdentifier("1.3.6.1.2.1.1"),
                   result,
                   60000,
                   10,
                   WalkMode.WithinSubtree,
                   null,
                   null);
```

TRAP Operation

It is usually an SNMP agent that sends out TRAP messages. The following code shows how to send an empty SNMP v1 TRAP message from `192.168.1.2` to an SNMP manager located at `192.168.1.3`,

```
Messenger.SendTrapV1(new IPEndPoint(IPAddress.Parse("192.168.1.3"), 162),
                     IPAddress.Parse("192.168.1.2"),
                     new OctetString("public"),
                     new ObjectIdentifier("1.3.6.1.2.1.1"),
                     GenericCode.ColdStart,
                     0,
                     0,
                     new List<Variable>();
```

SNMP v2 and above introduces a simplified TRAP v2 message,

```
Messenger.SendTrapV2(0,
                        VersionCode.V2,
                        new IPEndPoint(IPAddress.Parse("192.168.1.3"), 162),
                        new OctetString("public"),
                        new ObjectIdentifier("1.3.6.1.2.1.1"),
                        0,
                        new List<Variable>());
```

INFORM Operation

It is usually an SNMP agent that sends out INFORM messages. The following code shows how to send an empty INFORM message to an SNMP manager located at `192.168.1.3`,

```
Messenger.SendInform(0,
                        VersionCode.V2,
                        new IPEndPoint(IPAddress.Parse("192.168.1.3"), 162),
                        new OctetString("public"),
                        new ObjectIdentifier("1.3.6.1.2.1.1"),
                        0,
                        new List<Variable>(),
                        2000,
                        null,
                        null);
```

The manager should send back a reply to this INFORM message. Otherwise, a `TimeoutException` occurs.

Note: To help you understand how to use the API provided by #SNMP Library, there are more sample projects you can find under Samples folder in source code package. Both C# and VB.NET samples are available.

Related Resources

- Command Line Tools
- SNMP v3 Operations
- The API Reference

SNMP v3 Operations

By Lex Li

This page shows you how to perform SNMP v3 operations. Basic SNMP v3 operations (GET, SET and so on) are translated to #SNMP function calls.

In this article:

Background

SNMP v1 and v2c were designed to be simple, but one significant issue of them is security. It is very hard to hide community strings from sniffers, and also difficult to set access control on entities.

IETF recognized SNMP v3 RFC documents in 2004, which uses a new design to address the security concerns. The changes were so huge, so #SNMP has to expose a different styles of APIs for developers to perform v3 operations.

Discovery Process

SNMP v3 defines a discovery process in RFC 3414, that before an SNMP agent responds to a manager the two must interchange information such as time stamp, engine ID and so on.

```
Discovery discovery = Messenger.GetNextDiscovery(SnmpType.GetRequestPdu);
ReportMessage report = discovery.GetResponse(60000, new IPEndPoint(IPAddress.Parse("192.168.1.2"), 16
```

Above we perform such a discovery process by sending out a `GetRequestPdu` type of discovery message, and the agent replies with a `ReportMessage` which we can reuse later to construct a valid request.

User Credentials

SNMP v3 requires user credentials to be passed on in each requests, so we need to prepare that information ahead of time.

```
var auth = new SHA1AuthenticationProvider(new OctetString("myauthenticationpassword"));
var priv = new DESPrivacyProvider(new OctetString("myprivacypassword"), auth);
```

Assume the agent requires both authentication (SHA-1) and privacy (DES) protection, then above we create a single provider that embeds both passwords.

GET Operation

The following code shows how to send an SNMP v3 GET message to an agent located at `192.168.1.2` and query on OID `1.3.6.1.2.1.1.1.0`,

```
GetRequestMessage request = new GetRequestMessage(VersionCode.V3, Messenger.NextMessageId, Messenger
ISnmpMessage reply = request.GetResponse(60000, new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161));
if (reply.Pdu().ErrorStatus.ToInt32() != 0) // != ErrorCode.NoError
{
    throw ErrorException.Create(
        "error in response",
        IPAddress.Parse("192.168.1.2"),
        reply);
}
```

The reply object usually contains the response message we expect. By accessing `reply.Pdu().Variables` we can see what data are returned.

SET Operation

The following code shows how to send an SNMP v3 SET message to an SNMP agent located at `192.168.1.2` and set the value of OID `1.3.6.1.2.1.1.6.0` to `"Shanghai"`,

```
SetRequestMessage request = new GetRequestMessage(VersionCode.V3, Messenger.NextMessageId, Messenger
ISnmpMessage reply = request.GetResponse(60000, new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161));
if (reply.Pdu().ErrorStatus.ToInt32() != 0) // != ErrorCode.NoError
{
    throw ErrorException.Create(
        "error in response",
        IPAddress.Parse("192.168.1.2"),
        reply);
}
```

GET-BULK Operation

The following code shows how to send an SNMP v3 GET-BULK message to an SNMP agent located at `192.168.1.2` and query on OID `1.3.6.1.2.1.1.1.0`,

```
GetBulkRequestMessage request = new GetBulkRequestMessage(VersionCode.V3, Messenger.NextMessageId, Me
ISnmpMessage reply = request.GetResponse(60000, new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161));
if (reply.Pdu().ErrorStatus.ToInt32() != 0) // != ErrorCode.NoError
{
    throw ErrorException.Create(
        "error in response",
        IPAddress.Parse("192.168.1.2"),
        reply);
}

var result = reply.Pdu().Variables;
```

WALK Operation

The following code shows how to perform v3 WALK on an SNMP agent located at `192.168.1.2` starting at `1.3.6.1.2.1.1`,

```
var result = new List<variable>();
Messenger.BulkWalk(VersionCode.V3,
                  new IPEndPoint(IPAddress.Parse("192.168.1.2"), 161),
                  new OctetString("public"),
                  new ObjectIdentifier("1.3.6.1.2.1.1"),
                  result,
                  60000,
                  10,
                  WalkMode.WithinSubtree,
                  priv,
                  report);
```

TRAP v2 Operation

Note: TRAP v1 message format is obsolete by TRAP v2. Thus, for SNMP v3 TRAP operations, you can only use TRAP v2 message format.

The following code shows how to send a TRAP v2 message to an SNMP manager/trap listener located at `192.168.1.2`,

```
var trap = new TrapV2Message(
  VersionCode.V3,
  528732060,
  1905687779,
  new OctetString("neither"),
  new ObjectIdentifier("1.3.6"),
  0,
  new List<Variable>(),
  DefaultPrivacyProvider.DefaultPair,
  0x10000,
  new OctetString(ByteTool.Convert("80001F8880E9630000D61FF449")),
  0,
  0);
trap.Send(new IPEndPoint(IPAddress.Parse("192.168.1.2"), 162));
```

INFORM Operation

Note: Comparing to sending TRAP v2 messages, it is common to send INFORM messages.

The following code shows how to send an INFORM message to an SNMP manager/trap listener located at `192.168.1.2`,

```
Messenger.SendInform(
  0,
  VersionCode.V3,
  new IPEndPoint(IPAddress.Parse("192.168.1.2"), 162),
  new OctetString("neither"),
  new ObjectIdentifier(new uint[] { 1, 3, 6 }),
  0,
  new List<Variable>(),
  2000,
  DefaultPrivacyProvider.DefaultPair,
  report);
```

Note: To help you understand how to use the API provided by #SNMP Library, there are more sample projects you can find under Samples folder in source code package. Both C# and VB.NET samples are available.

Related Resources

- Command Line Tools

- An Introduction to #SNMP
- The API Reference

SNMP Device Discovery

By Lex Li

This page shows you how SNMP device might be discovered.

In this article:

Background

When SNMP was defined and published, there was no official way to discover SNMP enabled devices in the network. This might be caused by the well known security issues of SNMP itself, as if you know there is such a device in the network, you can somehow sniff the wire to get community names. Thus, if you are a device administrator, make sure you hide the devices if they should not be discovered.

However, some SNMP products, such as SNMP MIB browsers, might use an unofficial way to detect devices. Our sample, snmpdiscover, implements one common approach.

Attention: The following device discovery approaches rely on UDP broadcasting. THerefore, they are valid only for IPv4. IPv6 does not support broadcasting.

Simple Device Discovery for v1 and v2c (IPv4)

UDP allows broadcast, so if we broadcast an SNMP GET request with OID `1.3.6.1.2.1.1.1.0` using community name `"public"`, some devices will reply with their device information. In this way we know both the device IP address and type from the replies.

Note: You should avoid using “public” as community name, as it is so well known.

Discovery for v3 (IPv4)

RFC 3414 defines a discovery process for SNMP v3. This gives us a chance to discover all v3 enabled devices in the same network by broadcasting a simple discovery request without any credentials.

As in this way the device IP address is revealed, make sure your devices don’t use a common user name and passwords.

Related Resources

- Command Line Tools
- An Introduction to #SNMP

- SNMP v3 Operations

OCTET STRING and Encoding

By Lex Li

This page shows you how `OCTET STRING` type works internally.

In this article:

Background

If you notice, both SNMP data type `OCTET STRING` and `Opaque` can be viewed as byte array type in .NET. Maybe you wonder why `OCTET STRING` is not viewed as `System.String`, the reason is simple. `System.String` does have an important property that distinguishes itself from a byte array – encoding. It seems when SNMP was designed, it does not provide a default encoding for `OCTET STRING` explicitly (and sometimes it is even used just like a byte array when you store MAC address bytes in it).

Therefore, a problem has been there for a long time, "which encoding should be used there if an OctetString instance comes". In very old revision of #SNMP Library, the answer was ASCII and there was no way in them to change it to Unicode or any other encodings. Note that it is recommended you use ASCII if possible.

In more recent builds of #SNMP Library you can explicitly state which encoding you prefer.

Important Methods

As a reference at first, there are new methods and properties in `OctetString` class you should pay attention to,

1. The overloading constructor `public OctetString(string str, Encoding encoding)`. You may provide a specific `Encoding` so that the internal bytes are what you expect after initialization.
2. The overloading method `public string ToString(Encoding encoding)`. According to `Encoding` provided, the internal bytes are decoded to a correct string.
3. The instance property `public Encoding Encoding`. This is a read only property that illustrates what encoding is assigned to this `OctetString` instance.
4. The static property `public static Encoding DefaultEncoding`. This is a property to define default encoding used for all `OctetString` instance if no encoding is assigned via the constructor described in item 1. The default value of this property is `Encoding.ASCII`.
5. The overloading constructor `public OctetString(string str)`. `OctetString.DefaultEncoding` is used to generate internal bytes from `str`.
6. The overloading method `public string ToString()`. `OctetString.Encoding` is used to generate a string from internal bytes.

Scenarios

Now let's review a few common scenarios and discuss what you may do to customize `OctetString`'s behaviors.

Default ASCII Way

If you don't touch the properties and methods related to `Encoding`, then you are in a pure ASCII way. It should behavior the same as old releases of #SNMP (0.5, 1.0 and 1.1).

Default Encoding.* Way

If you change `DefaultEncoding` to any other `Encoding` types, and don't touch the properties and methods related to `Encoding` later, you are in a pure `Encoding.*` way.

Basic Hybrid Way

If some of your devices require ASCII while others require Unicode (aka UTF-16), then you may find it a little bit difficult. My suggestions are,

> Whenever you need to construct an `OctetString` object, specify its encoding explicitly. Whenever you need to use an `OctetString` object, check its `Encoding` before calling `ToString()`. If the `Encoding` is not correct, call `ToString(Encoding)` explicitly.

Last Words

I understand the current implementation is not perfect yet. When #SNMP parses incoming packets, it always use `DefaultEncoding` to construct `OctetString` objects from raw bytes. This is not optimal for hybrid cases. Hope a better way can be found soon.

Related Resources

Troubleshooting Guide

By Lex Li

This page shows you how to troubleshoot common problems.

In this article:

General Approach

Troubleshooting #SNMP related issues require basic knowledge of the following

- Computer networking (TCP/IP)
- UDP and SNMP specific

- .NET and C#

The steps below can often help determine what might be the cause of an error.

1. Does this problem reproducible? If it is reproducible, move on.
2. Do other libraries behave the same? Please test with Net-SNMP which is also open source. If not, move on.
3. Does the latest build in the repository work? If not, move on.
4. Is this a known issue? If not, move on.
5. Prepare an error report (with all information you can think of that is related) and extra information
6. SNMP packets related: please provide network packet captures if possible following *How to Capture Network Packets* .
7. .NET exceptions: please provide stack trace information.

When the cause cannot be determined at this stage, open a bug report with the log file or packet capture attached following “Reporting a Bug” section in :doc:/contribute/github .

How to Capture Network Packets

Wireshark

Use Wireshark you can capture network traffic and analyze the packets.

Note: network capture can contain confidential information. Before sharing it with others, make sure you strip out such critical data.

System.NET Tracing

System.Net tracing is another useful way to log network traffic.

Note: tracing log can contain confidential information. Before sharing it with others, make sure you strip out such critical data.

Windows ETW

Microsoft adds to latest Windows releases a built-in feature to capture network packets via ETW, which can be enabled easily following this guide .

Note: tracing log can contain confidential information. Before sharing it with others, make sure you strip out such critical data.

Beginners’s Guide on MIB Documents

By Lex Li

This page shows you which roles MIB documents play in SNMP communication.

Important: It is highly recommended that you buy a book such as Understanding SNMP MIBs to learn from the masters of this art.

In this article:

Background

MIB documents can be found from SNMP RFC documents, product manuals, and Internet downloads. So why do they exist and do I need them? The answer varies for each roles you play.

Device Vendors

If you are going to sell an SNMP enabled device, you need to prepare user manuals and also provide the MIB documents to your customers. Your developers have implemented management objects in the agent software, which is invisible to everyone else.

The customers can only learn how to manage the device via the MIB documents you provide.

Device Administrators

If you bought an SNMP enabled device from a vendor, make sure you grab the MIB documents and learn carefully which objects are exposed.

You can import the MIB documents to a monitor software, or write your own software to monitor the objects.

SNMP OID, Data Type, and Description

From the MIB documents you know the identifiers (OID) of the management objects, their data types, and descriptions. That information can then be put down to papers.

You can use #SNMP Library with MIB documents, as only OID is required during request construction in most cases. Of course data type is needed when performing SET operations. The descriptions help you understand the meaning of the data you get from the agent.

Related Resources

- #SNMP MIB Compiler Pro
- SharpSnmpPro.Mib Assembly

Samples

Command Line Tools

By Lex Li

This page shows you what are the command line tools shipped.

In this article

How to Acquire The Command Line Tools

Please follow "Install #SNMP Library via source code" section in Installing #SNMP Library on Windows to acquire the source code. Then launch Visual Studio 2015 to open `SharpSnmpLib.Classic.sln`.

The sample projects are under the "Samples" folder and devided into two languages, C# and VB.NET. Compile them and then they can be executed.

Typical Commands

Below describes typical commands for #SNMP command line tools. For more information on each tools, you may refer to the source code.

SNMPGET

For SNMP v1 and v2c, typical commands are

```
snmpget -c=public -v=1 localhost 1.3.6.1.2.1.1.1.0
snmpget -c=public -v=2 localhost 1.3.6.1.2.1.1.1.0
```

For SNMP v3, typical commands are

```
snmpget -v=3 -l=noAuthNoPriv -u=neither localhost 1.3.6.1.2.1.1.1.0
snmpget -v=3 -l=authNoPriv -a=MD5 -A=authentication -u=authen localhost 1.3.6.1.2.1.1.1.0
snmpget -v=3 -l=authPriv -a=MD5 -A=authentication -x=DES -X=privacyphrase -u=privacy localhost 1.3.6
```

SNMPSET

For SNMP v1 and v2c, typical commands are

```
snmpset -c=public -v=1 localhost 1.3.6.1.2.1.1.6.0 s Shanghai
snmpset -c=public -v=2 localhost 1.3.6.1.2.1.1.6.0 s Shanghai
```

For SNMP v3, typical commands are

```
snmpset -v=3 -l=noAuthNoPriv -u=neither localhost 1.3.6.1.2.1.1.6.0 s Shanghai
snmpset -v=3 -l=authNoPriv -a=MD5 -A=authentication -u=authen localhost 1.3.6.1.2.1.1.6.0 s Shanghai
snmpset -v=3 -l=authPriv -a=MD5 -A=authentication -x=DES -X=privacyphrase -u=privacy localhost 1.3.6
```

SNMPBULKGET

For SNMP v2c, typical command is

```
snmpbulkget -v=2 -c=public -Cr=10 localhost 1.3.6.1.2.1.1.1.0
```

For SNMP v3, typical commands are

```
snmpbulkget -v=3 -l=noAuthNoPriv -u=neither -Cr=10 localhost 1.3.6.1.2.1.1.1.0
snmpbulkget -v=3 -l=authNoPriv -a=MD5 -A=authentication -u=authen -Cr=10 localhost 1.3.6.1.2.1.1.1.0
snmpbulkget -v=3 -l=authPriv -a=MD5 -A=authentication -x=DES -X=privacyphrase -u=privacy -Cr=10 local
```

SNMPGETNEXT

For SNMP v1 and v2c, typical commands are

```
snmpgetnext -c=public -v=1 localhost 1.3.6.1.2.1.1.1.0
snmpgetnext -c=public -v=2 localhost 1.3.6.1.2.1.1.1.0
```

For SNMP v3, typical commands are

```
snmpgetnext -v=3 -l=noAuthNoPriv -u=neither localhost 1.3.6.1.2.1.1.1.0
snmpgetnext -v=3 -l=authNoPriv -a=MD5 -A=authentication -u=authen localhost 1.3.6.1.2.1.1.1.0
snmpgetnext -v=3 -l=authPriv -a=MD5 -A=authentication -x=DES -X=privacyphrase -u=privacy localhost 1
```

SNMPWALK

For SNMP v1, typical command is

```
snmpwalk -c=public -v=1 -m=subtree localhost 1.3.6.1.2.1.1
```

For SNMP v2c, typical command is

```
snmpwalk -v=2 -c=public -Cr=10 -m=subtree localhost 1.3.6.1.2.1.1
```

For SNMP v3, typical commands are

```
snmpwalk -v=3 -l=noAuthNoPriv -u=neither -m=subtree -Cr=10 localhost 1.3.6.1.2.1.1
snmpwalk -v=3 -l=authNoPriv -a=MD5 -A=authentication -u=authen -m=subtree -Cr=10 localhost 1.3.6.1.2
snmpwalk -v=3 -l=authPriv -a=MD5 -A=authentication -x=DES -X=privacyphrase -u=privacy -m=subtree -Cr=
```

Built-in community names and users for snmpd

SNMP v1 and v2c

Get community name	Set community name
public	public

SNMP v3

Community names are obsolete in SNMP v3, so snmpd.exe supports three users (to match three modes).

User name	Security mode	Authentication provider	Authentication phrase	Privacy provider	Privacy phrase
neither	noAuthNoPriv	default	N/A	default	N/A
authen	authNoPriv	MD5	authentication	default	N/A
privacy	authPriv	MD5	authentication	DES	privacyphrase

Checkout

The samples can be used to carry out basic SNMP operations. So if you are going to learn the basics, you should follow them.

Pairing the SNMP test agent and the manager side tools

Run snmpd.exe as administrator and click "Start listening" button without modifying any settings. This allows the test SNMP agent to hook to port 161 on all network interfaces (IP v4 and v6).

Information about the agent builtin community names and user accounts can be found above in *Built-in community names and users for snmpd* .

Then the command line utilities can be run at command prompt with *Typical Commands* . The agent will respond with correct packets.

Pairing the SNMP test agent and the trap listener tool

Run snmptrapd.exe at command prompt as administrator. It will hook to port 162 and monitor incoming trap messages.

Click the "Sent Trap v1", "Send Trap v2", "Send Inform v2" and "Send Inform v3" buttons on the test agent panel. See those messages are captured by the trap listener tool.

Above setup assumes that all tools are running on the same machine. If you want to test out agent and manager sides each on a dedicate machine, make sure you open the firewall ports and allow SNMP packets to go through.

Related Resources

- An Introduction to #SNMP
- SNMP v3 Operations
- Source Code License for Releases

Agent Development

By Lex Li

This page shows you how the agent sample works.

In this article:

Dependencies

When you open snmpd.csproj in Visual Studio (or another IDE), you should first check what are its references.

The `SharpSnmpLib` NuGet package contains our #SNMP Library components.

Note: This package targets multiple platforms.

Facade and Pipeline

We no longer need to construct an `Agent` object, because that class is too old to be used. We try to construct an `SnmpEngine` object instead.

`SnmpEngine` is the facade that you should put on top. It is a very complex class that replaces our old Agent class. To construct it, we need: an `EngineGroup` object, which contains all engine global objects; A `Listener` object, which monitors incoming SNMP message; An `SnmpApplicationFactory`, who manages the pipelines.

`SnmpApplicationFactory` creates new pipelines when its pipeline pool is full of busy pipelines. The objects we pass to its constructor finally goes into each pipeline as they are shared among them (at this moment).

A pipeline is in fact an `SnmpApplication` object. It has several processing phases and each phase utilizes a few helper classes, which you can extend.

An SNMP message captured by `Listener` will be packaged as an `SnmpContext` object. This context object is passed to the pipeline and processed in each phase to generate a correct response message.

About how to construct instances of each classes, you can refer to snmpd sample source code.

Pipeline in Details

Currently we only have four phases in the pipeline,

- Authentication
- Request handler mapping
- Request handler executing
- Logging request

Authentication

By checking the sample code you can see how the primary membership provider (`ComposedMembershipProvider`) is created and added into the pipeline. The request is checked by the provider and dropped if it does not contain a proper community name.

`ComposedMembershipProvider` is a special membership provider, who allows you to support different SNMP versions. If you only target a specific version, you can use the version specific provider as primary one.

To customize authentication, make use of the existing providers or write your own.

Request handler mapping

For authenticated message, in this phase it is verified again and mapped to a message handler.

Message handlers are injected to the pipeline, too. So you can analyze the sample code to know how many handlers are there already, and how each registers its interested message (SNMP version and verb).

For example, `GetV1MessageHandler` is only interested in v1 GET messages, while `GetMessageHandler` in v2 and v3 GET messages.

Carefully configure the existing handlers, you can achieve different SNMP engine configurations, so as to meet different requirements. You can write your own handlers to further customize the pipeline.

Request handler executing

Once a message handler is found for the message, in this phase the handler performs the requested operation and generate a response message.

You should notice that `*V1MessageHandler` classes follow RFC 1157 specification to handle v1 messages, while other handler classes follow RFC 3416 to handle v2 and v3 messages.

Logging request

In this phase the response message is sent back, while the logger logs the processing into log files.

After phase 4, the pipeline is reused by `SnmpApplicationFactory` for future messages.

We may add an authorization phase to achieve user based authorization, but it is not yet designed and implemented.

ObjectStore and ISnmpObject

When a handler tries to do a typical SNMP operation, it looks into the `ObjectStore` object to locate the specified object.

Currently we only have a few sample objects created, to test out the pipeline. You can find them under `Lextm.SharpSnmpLib.Objects` namespace.

If you want to write more objects, you can follow our sample ones.

`ObjectStore` is not yet thread safe, which will be improved in the future.

Performance Tuning

The SNMP engine is multi-threading by nature. Both the `ListenerBinding` and `SnmpApplication` instances utilize the default thread pool to handle requests asynchronously. Thus, it is a must to make sure the thread pool is optimized before requests come in.

Before calling `SnmpEngine.Start`, it is recommended that the below code to be executed, which sets the minimal worker thread count to a suitable value.

```
int minWorker, minIOC;
// Get the current settings.
ThreadPool.GetMinThreads(out minWorker, out minIOC);
var threads = engine.Listener.Bindings.Count;
ThreadPool.SetMinThreads(threads + 1, minIOC);
```

If not tuned, the very first request to this agent will cost extra time (noticeably several seconds if there are too many bindings), as the operating system needs to create new threads before putting them into the thread pool.

Limitation Explained

You should take a look at `MainForm.cs` and read what extra lines are required to configure the `SnmpEngine` object, how to start and stop it. SNMP tables can be quite complex, while this sample only shows simple tables such as IfTable for simplicity.

As the sample is released under MIT/X11 license. The snmptrapd sample also uses the pipeline to handle trap messages, and once you are familiar with snmpd, you can switch to it to learn how to construct a browser side pipeline accordingly.

This sample is provided to demonstrate how the library might be used. If you want to build a full feature SNMP agent based on this sample, then many changes (mult-threading and security related) are mandate.

Related Resources

- An Introduction to #SNMP
- Command Line Tools
- Source Code License for Releases

Contribute

Note: We reuse the ASP.NET docs style guide.

Contributing Guide

GitHub Guide

By Lex Li

This page shows you how to use #SNMP Library repo on GitHub to collaborate.

In this article:

Submitting a Patch

An issue might be opened to discuss with the maintainers before creating the patch. This helps the maintainers track your progress, and provide guidance if needed.

1. Learn about GitHub via http://help.github.com/.
2. Create your own fork from the main repo at https://github.com/lextm/sharpsnmplib.
3. Create a new branch with a meaningful name.
4. Make the changes on this new branch in your fork, and test it fully.
5. Create a pull request back (link the new branch in your fork to the master branch of main repo).
6. Document all your changes in details as part of the pull request, which is critical to better communicate with maintainers.

Patches will be reviewed and merged as early as possible.

Note: If the patch is large, it might take several weeks to review and merge.

Note: To see what makes a good pull request, please follow *Suggestions on Creating Pull Requests* section.

Reviewing a Patch

Maintainers should respond to new pull requests as early as possible by commenting like this,

- "Acknowledged. Will start review."

which gives the contributors a hint that the process has begun.

Further responses can be like,

- "Comments have been left at **. Please revise your patch like this,"
- "Reviewed. ** will be merged, while ** will not. The reasons are,"

which will keep all discussions public to reveal all necessary technical information for future reference.

Reporting a Bug

Note: If you already have a patch for the bug, please follow *Submitting a Patch* section.

Note: If you are not sure whether it is a bug, please follow *Asking a Question* section.

1. Learn about GitHub via http://help.github.com/.
2. Review existing issues at https://github.com/lextm/sharpsnmplib/issues that are marked as bug or enhancement to avoid duplicate.
3. Create a new issue on https://github.com/lextm/sharpsnmplib/issues and provide all information in details.

You are welcome to provide step by step instructions, as that can help other reproduce and investigate the issue. If you are willing to share a sample project, please use a service such as DropBox or OneDrive.

Reviewing a Bug

Maintainers should respond to bug reports as early as possible by commenting like this,

- "Acknowledged. Will start review."

which gives the contributors a hint that the process has begun.

Further responses can be like,

- "Could not reproduce it. Please provide more information to assist investigation such as **,"
- "Reviewed. ** is a bug that can be reproduced. Will perform further investigation on how to resolve it. This may take a long time."

which will keep all discussions public to reveal all necessary technical information for future reference.

Asking a Question

Asking a Question on StackOverflow (Recommended)

1. Log into StackOverflow.
2. Before starting asking a new question, please review all questions under tag `sharpsnmplib` in case yours has already been answered.
3. Click Ask Question to create a new question.
4. Add tag `sharpsnmplib` to this question, and also include all information in details.

Then you can wait till users reply to your question.

Asking a Question on GitHub

1. Learn about GitHub via http://help.github.com/.
2. Before creating the issue, please review all existing issues especially our FAQ in case the issue has already been reported and resolved.
3. Create a new issue on https://github.com/lextm/sharpsnmplib/issues and provide all information in details.

Answering a Question

Maintainers might join StackOverflow and monitor discussions under `sharpsnmplib` tag.

Maintainers should respond to questions on GitHub as early as possible by commenting like this,

- "Acknowledged. Will start review."

which gives the contributors a hint that the process has begun.

Further responses can be like,

- "Could not reproduce it. Please provide more information to assist investigation such as **,"
- "Reviewed. ** is a bug that can be reproduced. Will perform further investigation on how to resolve it. This may take a long time."

which will keep all discussions public to reveal all necessary technical information for future reference.

Tag such an issue with question tag.

Close such issues once a meaningful answer is given.

Mark an issue as `faq candidate` if it should be considered as an FAQ.

Overview of Issue Tags

Maintainers should use the tags as early as possible so as to help each other to easily track the progress. The decoration tags are most useful for items which are not yet assigned to milestones.

Tags for Item Categories

The following are used to assign an item to a specific category,

- bug This item was reported as a bug of this product. The reporter expects a fix.
- enhancement This item was reported as an enhancement request. The reporter expects a certain feature to be enhanced or a new feature to be implemented.
- task This item was reported as a task. The reporter expects a maintainer to perform a piece of work (usually not development).
- idea This item was reported as a new idea. The reporter expects some discussion on a feature request. Once discussed, this item might be upgraded to an enhancement.
- question This item was reported as a question. The reporter expects some discussion on a problem met about this product. Once discussed, this item might be upgraded to a bug, an enhancement, or an idea.
- tech debt This item was reported as bad smells detected in the code base. The reporter expects changes in the code base to remove the bad smells.
- pull request This item was used to handle a pull request.

Tags for Decoration

The following are used to decorate an item so as to make it easy to see its status and required actions,

- dependency bug This only applies to bug items. It means the bug was caused by a bug of one of the dependencies (such as bugs of .NET Framework/Mono bugs, or bugs of the operating systems).
- not an issue This means after discussion, there is nothing to be done further (usually for false positives).

- wontfix This means the item (usually bugs) won't be fixed due to a strong justification. An agreement must be achieved among the maintainers.
- duplicate This means the item is exactly the same as another existing item. The maintainers should explicitly point out which item will be the focus and mark all the rest as duplicate.
- tentative This means based on the provided information it is not likely to move on. The reporter should provide more information and drive the discussion.
- soon to close This means there is little left to do on the item. The maintainers are going to close the item after a few more days (usually applied to tentative and cannot reproduce items).
- cannot reproduce This means the maintainers failed to reproduce the symptoms described in a bug report. The reporter should provide more information (process dumps, sample projects, screen shots, video clips and so on) and drive the investigation.
- in progress This means the item has been actively investigated by the maintainers.
- up for grabs This means community contribution is welcome.

Suggestions on Creating Pull Requests

All pull requests are appreciated (even if some we cannot merge). The following can make the pull requests simpler for reviewers, so hope you can follow them.

- If possible, send multiple pull requests for individual tasks and avoid a pull request for multiple tasks. Properly isolating changes to meaningful batches makes it quicker to analyze and assert the changes.
- Fork and create a new branch with a meaningful name first before making the changes.
- Squash all commits on this new branch to only one or two before sending the pull request.
- Wait for comments from the reviewers. It usually takes weeks as the reviewers might not be able to finish quickly. Don't make further changes at this stage to avoid changes of this pull request.
- Revise the code based on feedbacks, and then make a second commit with necessary changes and push to the branch in your fork, where GitHub automatically appends it to the pull request for further review.

Then the reviewers will decide whether to accept or reject the pull request based on code quality.

One important notice is that some pull requests might not be accepted, but they are still valuable to the community,

- It contains a nice-to-have feature (such as options to enable/disable part of a theme, or a visual element) for some users but not all.
- It introduces a feature (such as new visual elements) that goes beyond Visual Studio look and feel.

Such pull requests are of great value of course. But since the primary goal of DPS is to simulate Visual Studio look and feel, and the code base is already huge to maintain, we try to avoid bringing in non-core features.

Suggestions on Reviewing Pull Requests

Please leave a message that you are going to review a pull request. That should let the submitter know it's been reviewed.

Leave all comments at a time, so that the submitter can revise them altogether to form a new commit.

Decide carefully whether to accept or reject a pull request. Leave explanation for future reference.

Support

Release Notes

By Lex Li

This page documents major releases of #SNMP Library.

In this article:

Releases

Detailed release notes have been moved and can be found at GitHub .

Product Lifecycle

Due to the introduction of #SNMP Pro products, the support lifecycle will match the corresponding Pro products.

#SNMP Pro current release (1.2.0) requires #SNMP Library 9.0.5 and above, so release 9.0.5 and above are all supported.

Any release older than 9.0.5 is obsolete and no longer supported.

Related Resources

- Support Services

Supported Platforms

By Lex Li

This page describes for major releases of #SNMP, which version(s) of .NET Framework and Mono are supported.

In this article:

.NET Framework

.NET 1.0/.NET 1.1

These .NET Framework releases are end of life.

#SNMP does not support them.

.NET 2.0 RTM/.NET 2.0 SP1/.NET 2.0 SP2/.NET 3.0 RTM/.NET 3.0 SP1/.NET 3.0 SP2/.NET 3.5 RTM

These .NET Framework releases are end of life.

#SNMP does not support them.

.NET 3.5 SP1

According to Microsoft support policies , this .NET Framework release is still supported.

#SNMP 8.0 release (code name TritonMate) was the last release for this platform.

It is recommended to upgrade to .NET 4.5.2/.NET 4.6/.NET 4.6.1.

.NET 4/.NET 4.5/.NET 4.5.1

These .NET Framework releases are end of life.

#SNMP does not support them.

.NET 4.5.2/.NET 4.6/.NET 4.6.1

According to Microsoft support policies, these releases are still supported.

#SNMP 8.5 release (code name TritonMate Refresh) and above can be used on these platforms.

.NET Compact Framework

Support for .NET Compact Framework 3.5 and above is still experimental.

Mono

#SNMP 8.5 release (code name TritonMate Refresh) and above can be used on Mono.

The following operating systems are tested,

- OS X/macOS
- Ubuntu
- Fedora Core

It is recommended to use latest Mono release, such as 4.2.1 and above.

.NET Core

#SNMP 10.0 release supports this platform officially via .NET Standard 1.3 compliance.

Related Resources

- Support Services

CHAPTER 2

Pro Edition

MIB Compiler and the related library is released as #SNMP Pro edition.

More information can be found at its own documentation site .

CHAPTER 3

Contribute

We accept pull requests! But you're more likely to have yours accepted if you follow these guidelines:

1. Read the Contributing Guide.
2. Follow the ASP.NET 5 Docs Style Guide.