**Writing Experts for Network Monitor**

By: Michael A. Hawker

Copyright 2008-2009 Microsoft

Table of Contents

[Introduction 3](#_Toc228080365)

[Network Monitor API Introduction 3](#_Toc228080366)

[What is the Network Monitor API? 3](#_Toc228080367)

[What can I do with the Network Monitor API? 3](#_Toc228080368)

[Where do I get the Network Monitor API? 3](#_Toc228080369)

[How do I work with the Network Monitor API? 3](#_Toc228080370)

[Network Monitor Expert System 4](#_Toc228080371)

[Accessing Experts in Network Monitor 4](#_Toc228080372)

[Writing Experts 4](#_Toc228080373)

[Parameters for Experts 5](#_Toc228080374)

[{NetmonOpenCaptureFile} 5](#_Toc228080375)

[{NetmonSelectedFrameRange} 5](#_Toc228080376)

[{NetmonConversationFilter} 5](#_Toc228080377)

[{NetmonDisplayFilter} 5](#_Toc228080378)

[Setting up a Development Environment 6](#_Toc228080379)

[Using the Included SDK 6](#_Toc228080380)

[Packaging Experts 7](#_Toc228080381)

[Building an MSI Package 7](#_Toc228080382)

[Sample Settings.txt File 8](#_Toc228080383)

[Building the SDK 10](#_Toc228080384)

[Saving Settings in the Registry 10](#_Toc228080385)

[Expert Portal Link 10](#_Toc228080386)

[Procedure Steps 10](#_Toc228080387)

[Network Monitor Expert Test Plan 11](#_Toc228080388)

[Configurations 11](#_Toc228080389)

[Tests 11](#_Toc228080390)

[Install 11](#_Toc228080391)

[Invoke and Parse Parameters 11](#_Toc228080392)

[Parser Setup 11](#_Toc228080393)

[Display Help 11](#_Toc228080394)

[Specific Tests 11](#_Toc228080395)

# Introduction

There are times when simple network analysis on individual network frames is not enough. Many times the data needs to be looked at in its entirety. In these cases, it becomes far too complex for a single individual to make heads or tails of a situation. This is where Experts play a vital role. Experts are independent utilities which perform complex analytical tasks on network traces.

Network Monitor 3.3 introduced the ability to launch experts on captured traces directly from the user interface. This provides a great platform for making network analysis easier. The other great thing about Network Monitor is an API is provided which can do a lot of the work for you already. The Network Monitor API allows you to load a capture file and break it down frame by frame. You can even apply filters and look at individual fields as you can with Network Monitor itself.

This document explains how it is possible to combine these two features to create an Expert which Network Monitor users can download, install, and run on their own network traces to perform analysis not possible with Network Monitor alone.

This document assumes you have an understanding of Network Monitor, Network Analysis, and intermediate programming knowledge of C++ or C# which includes using external libraries.

# Network Monitor API Introduction

## What is the Network Monitor API?

The Network Monitor API is a low-level library packaged in a DLL which allows programmers to access Network Monitor functions from within their own programs. It is installed alongside Network Monitor and much more detailed information about using it is provided in the Network Monitor Help.

## What can I do with the Network Monitor API?

The Network Monitor API provides a lot of functionality available within Network Monitor. It is possible to do such things as capturing live network data, opening capture files, and using the Network Monitor Parsers to filter and retrieve data.

## Where do I get the Network Monitor API?

The Network Monitor API is only installed with Network Monitor. It is not freely distributable at this time. Therefore, users of programs which rely on the API need to install Network Monitor first.

The C++ library and header files can be found in the API directory in the Network Monitor install directory. A C# header file is also included to allow unmanaged PInvoke access to the API from within .NET applications.

## How do I work with the Network Monitor API?

See “Setting up a Development Environment” later in this tutorial for more information.

# Network Monitor Expert System

## Accessing Experts in Network Monitor

Network Monitor Experts can currently only be launched from a previously saved capture file. If a live trace has been done, it must first be saved and then opened before an Expert can be launched.

Experts are available in two locations, one is the top-level Experts menu and the other is from the right-click menu in the Frame Summary window. By default, no Experts are installed. The user must select the Download Experts menu item to find and install Experts. This will take them to the <http://NMExperts.CodePlex.com/> page.

Experts should be packaged in an installer MSI package the user can download. Once installed, the user can open a saved trace and have the Expert available for use on that file.

Note: Newly installed Experts will not appear in the menu for any capture file currently open in Network Monitor, the tab must be closed and reopened for the Expert menu to refresh.

# Writing Experts

There is a lot involved in writing an Expert. This tutorial is focused more on the process involved after an Expert has been created. Creating Experts is a complex task because specific domain knowledge is needed for their creation; however, the shell of an Expert is really just a program which reads in a network trace and provides some output based upon its contents.

The first step in writing an Expert is finding a need which can’t be satisfied with the basic analysis tools and Experts already available for Network Monitor. The second is figuring out how you can use the captured data available to solve the problem.

An Expert needs to be able to rely only on the captured data in the trace and a few auxiliary pieces of input. It needs to do most of the work for the user without making their life more difficult. The best place to start is by looking at the currently available Experts to see how they work.

Once you have a clear idea for an Expert, feel free to read the documentation provided within Network Monitor for more information on the Network Monitor API. Many examples are provided which can guide you in the programmatic development of an Expert.

## Parameters for Experts

Network Monitor can pass a number of different values as arguments to an Expert. These arguments should be read in by the Expert to perform its analysis. When creating the installer package, you can specify how these arguments should be passed to the Expert (more information is available in the Packaging Experts section). By default, the Conversation and Display filter parameters have their quotes escaped on the commandline. This functionality can be changed for custom parameter parsing in the MSI installation settings (see the following section).

### {NetmonOpenCaptureFile}

Specifies the full path and file name of the capture file to run the Expert on.

### {NetmonSelectedFrameRange}

Specifies the frames the user currently had selected in the UI before launching the Expert.

It is formatted as a string with no spaces containing only whole numbers in increasing numerical order, dashes and commas. Commas are used to separate groupings of individual frames or ranges of frames. An individual frame will simply be a whole number. A range is indicated with a dash between two numbers and represents an inclusive set between the indicated numbers. A comma is used to distinguish these elements.

e.g. 1-5,7,9-200,350-450,567,789

**NOT** 1-5,7-11,6,400-300,127

The numbers represent the frame numbers indexed starting at 1 as they are displayed in the UI. The Network Monitor API uses a zero-based index.

An explicit C# code example can be found and used in the CommandLineArguments.cs file included with the SDK.

### {NetmonConversationFilter}

The currently used (and applied) conversation filter.

### {NetmonDisplayFilter}

The currently used (and applied) display filter. (Generally, the capture filter is saved in the NetworkInfo frame of a trace.) To obtain similar results to what the user is viewing in the UI, first run the frame through the conversation filter, and then the display filter. See the included sample code.

## Setting up a Development Environment

This section will show you how to create a simple project in Microsoft Visual Studio 2008 which can access the Network Monitor API.

1. Open Microsoft Visual Studio 2008
2. On the **File** menu, select **New**, and then **Project**.
3. Select a Windows Forms Application or Console Application under Visual C# -> Windows depending on your Expert’s needs. We’ll assume you’re creating a GUI driven expert, and work from “Windows Forms Application”
4. Give your project a name and location; we’ll call ours “NetmonTestExpert.”
5. Click **OK**.
6. In Solution Explorer, right-click on your Project name (NetmonTestExpert).
7. Select Add, then Existing Item.
8. Navigate to the install directory of Network Monitor (usually Program Files\Network Monitor 3).
9. Go to the API directory and select NetmonAPI.cs.
10. Click **Add**.
11. To use the Network Monitor API, to the top of you C# file add the line:  
     using Microsoft.NetworkMonitor;
12. You can now reference Network Monitor Native API calls with the NetmonAPI object in your code. Refer to the SDK documentation under the Network Monitor Help menu for more information on specific functions and code examples.
13. In some instances, you may need to allow unsafe code to run when obtaining certain values from the API. In these cases you need to check the “Allow unsafe code” box in the **Build** tab under your project settings.

## Using the Included SDK

A Visual Studio 2008 project has been included. Be sure to extract the SDK to a location on your harddrive. The ExpertExample.sln file included should allow you to open the project contained in the ExpertExample directory. The ExpertMSI directory contains the MSI builder tool explained in the next section.

The project contains three code files. They are NetmonAPI.cs, CommandLineArguments.cs, and Program.cs:

* **NetmonAPI.cs** is the C# header file included with Network Monitor.
* **CommandLineArguments.cs** is a simple commandline argument processor specifically for the parameters passed by Network Monitor. It is portable and can be reused in other projects.  
  It provides a **Parse** method which accepts the **Main** arguments for your program. It will then populate properties such as CaptureFileName, DisplayFilter, and ConversationFilter. The SelectedFrames property provided is a special enumerator which can be used in loops to obtain the specific frame numbers the user has selected. However, the SelectedFramesString property still contains the raw string.
* **Program.cs** shows how to load a file, filter it to the Network Monitor view, and display details on individual frames using the Network Monitor API.

Other files included:

* The **msi** directory contains two pre-built MSIs, which can be used to quickly install and test the example Expert.
* **Expert.ico** can be replaced with your own icon file to be the icon which appears under Programs and Feaures in Windows Vista for your Expert once it is installed.
* **My Expert Help File.rtf** is the help file for your Expert displayed in Network Monitor. It is automatically copied and included in the MSI built by the project. If you want to rename it, be sure to update the settings.txt file with the new name.
* **Settings.txt** contains settings for building the MSI. It is a partial set of the full MSI parameters described in the previous section as some are handled automatically for you by the VS project provided. Before releasing an Expert build on this SDK be sure to update the parameters here to display the correct company information and naming for your Expert in the installer.
* **Writing Experts for Network Monitor.docx** this file.

# Packaging Experts

## Building an MSI Package

Building an MSI package for your Expert isn’t as hard as it sounds. It can also register your Expert with Network Monitor so users will have easy access to your tool.

Our MSI creation package is available as part of this project and is built automatically when the project is built in a Release configuration. See more information on building the SDK in the next section.

The MSI script tool is found in the ExpertMSI directory and contains a usage.txt file for more advanced usages of the tool. We have provided example usages below which can be modified for simple setup creation. However, as mentioned, the default Visual Studio 2008 project included automatically builds an MSI, if you set your target to a Release build. *Be sure to change the UPGRADECODE in settings.txt and your executable GUID (Properties->Application->Assembly Information) before releasing an Expert MSI.*

Before you can run the tool above, you need to install two things, the WiX toolkit and a Perl interpreter.

The WiX Toolkit binary zip is available here: <http://sourceforge.net/project/showfiles.php?group_id=105970&package_id=114109>

It is assumed it is installed to C:\Program Files\WiX\ in our example.

Strawberry Perl for Windows is available here: <http://strawberryperl.com/releases.html> and assumed to be in the default path.

You also need a GUID Generator, but a simple one is included (it requires the .NET Framework). The source has also been provided in the ExpertMSI directory.

## Sample Settings.txt File

Note the fields mentioned below are for running the tool stand-alone. The default project included handles some properties automatically.

TargetFile=C:\Some Path\Some More Path\SetupExpertName.msi

ExpertName=My Expert Name Here

Manufacturer=My Company Name

ExpertVersion=1.0.0.0

TargetNetmonVersion=any

PlatformID=amd64

BinaryPath=C:\Path to your Executable

ExpertCommand=ExpertExecutableName.exe

ExpertCommandLineArgs="{NetmonOpenCaptureFile}"

ExpertHelpFile=HelpFileName.rtf

WixToolPath=C:\Program Files\WiX\

UPGRADECODE=UNIQUE GUID HERE

GuidGenerator=C:\Some Path\Some More Path\QGUID.exe

**TargetFile**

The Target file where you want the MSI file to be created.

**ExpertName**

This is the name of your Expert as it will appear in Add/Remove Programs and the Network Monitor Experts menu

**Manufacturer**

This should be your company name.

**ExpertVersion**

This is the full version number of the Expert. Don’t forget to change it if this is an upgrade to an existing expert.

**TargetNetmonVersion**

The full version number ex. 03.03.1444.0001 of a Network Monitor build which is the required version for your Expert to run or **any** for all Network Monitor versions. Build 03.03.1444.0001 is the first version with Expert functionality and should be the minimum value you specify here.

**PlatformID**

The platform target for your Expert (either **amd64**, **x86**, or **ia64**). Remember that Network Monitor runs on each of these three platforms. It would be best to cross-compile your Expert on a 64bit machine to obtain both 32 and 64 bit binaries. Although amd64 is mentioned here, this also targets the 64 bit Intel platform, as well.

**BinaryPath**

The full path to where your Expert executable and dependent files reside. This should not be the same directory as where you’ve installed the Expert MSI tool. Otherwise, all the Expert MSI tool files will be included in your own MSI. You should create a separate sub-directory which only contains your Expert related files to point to with this path.

Do not include files installed by Network Monitor (like nmapi.dll). Also, the user can configure the install path, so don’t hard-code the default Program Files directory into your Expert.

**ExpertCommand**

The file name of your Expert executable to run when a user selects your Expert in Network Monitor

**ExpertCommandLineArgs**

The arguments for your Expert, including acceptable values which are automatically replaced by Network Monitor before passing to your Expert (See the Parameters for Experts section above). Expert parameters are passed in {} Ex. {NetmonOpenCaptureFile}. Be sure to use "" as in the example above to accept paths for files with spaces in them.

e.g. to get everything use the following text:

"{NetmonOpenCaptureFile}" /d "{NetmonDisplayFilter}" /c "{NetmonConversationFilter}" /f {NetmonSelectedFrames}

**ExpertDontEscapeFilterArgs**

Normally, we escape quotes in the NetmonDisplayFilter and NetmonConversationFilter parameters so applications will correctly receive the filter string to pass into the NMAPI. If you would like the raw string for command-line programs or for alternative commandline parsing methods, you can set this optional flag to 1.

**ExpertHelpFile**

The relative location from the binary path to the name of the RTF file which provides help on how to use your Expert. It is also a good location to store author and Web site information for your Expert. See the provided template for more details.

**WixToolPath**

The installation location of the WiX Toolkit on your computer

**UPGRADECODE**

Identifer GUID value for your expert.

You should use the same value for future versions of your expert. That way, Windows will automatically upgrade any existing version of the expert that the user might have on their system.

**GuidGenerator**

Path and executable name of a GUID generator program which outputs GUIDs to stdout. If not provided, you’ll be prompted to input GUIDs.

## Building the SDK

The SDK included shows a simple example which will print information about frames as they are displayed in Network Monitor. It uses all the parameters Network Monitor provides.

The SDK has two directories, the ExpertSDK directory contains all the source to build the Expert. The ExpertMSI directory contains the MSI builder tool described above.

The Expert SDK is preconfigured to automatically build an MSI when the project is built in a Release configuration with a specified platform (x86 or x64). It is important to change some fields before releasing your own Expert, such as the UPGRADECODE and your project GUID *(Properties->Application->Assembly Information)*.

## Saving Settings in the Registry

Although this Expert doesn’t save any settings, it is recommended that you save registry settings to the following registry path:

HKEY\_CURRENT\_USER\Software\Microsoft\Netmon3\Experts\UPGRADECODE

Where UPGRADECODE is the GUID in the XXX-XXX format used for the MSI build. This registry path will automatically be removed when the Expert is uninstalled. It will remain untouched during upgrades and reinstallation.

# Expert Portal Link

If you’re interested in releasing your Expert on [CodePlex](http://nmexperts.codeplex.com) or the Web and would like a link to your project, we’d be happy to hear from you. Before contacting us though be sure you’ve adhered to our guidelines below:

## Procedure Steps

1. Run [FxCop](http://msdn.microsoft.com/en-us/library/bb429476(VS.80).aspx) & [StyleCop](http://code.msdn.microsoft.com/sourceanalysis)(C#) or [PREfast](http://msdn.microsoft.com/en-us/library/ms182025(VS.80).aspx)(C++) on Expert.
   1. Fix any Issues related to the Expert.
   2. Any exceptions must be discussed.
2. Have Documentation & Landing Page Text.
3. Create MSI Package.
4. Verify Test Procedures (below).

# Network Monitor Expert Test Plan

This section outlines tests to be performed on Network Monitor Experts. The actual functional testing of an expert will vary greatly depending on the function of the expert and is not addressed here. This section only covers the common functions needed to plug an expert into Network Monitor.

## Configurations

Although experts may be used with Network Monitor 3.2, UI integration with experts is only available Network Monitor 3.3 and higher. For these tests, Network Monitor 3.3 is assumed.

Network Monitor 3.3 runs on a variety of versions of Windows, and on different hardware architectures. The Network Monitor Expert interfaces are not sensitive to the different platforms, but these tests may be run on all supported combinations of hardware/software.

## Tests

### Install

* The Expert installs on the target platform using normal installation techniques (double-click .msi).
* The Expert registers successfully in Network Monitor and is visible from both the Experts menu and from the frame summary right-click menu.
* The Expert can be uninstalled from the “Uninstall or Change a program” list in Programs and Features (or from comparable applications in Windows other than Vista).

### Invoke and Parse Parameters

The Expert can be invoked in different ways with different parameters:

* Invoke from Experts Menu using >Launch Expert.
* Invoke from Frame Summary right-click menu>Launch Expert.
* Invoke with a range of frames selected.
* Invoke with a conversation filter applied.
* Invoke with a display filter applied.

Not all parameters may be used by the Expert, but all parameters and combinations may be passed from Network Monitor, so they should be handled gracefully. If inappropriate selections are made in the Network Monitor UI, a usable error message should be presented.

### Parser Setup

The Expert may be invoked when the user has customized their parser packages. If the Expert relies on a specific parser to function, it should handle a case where it is missing gracefully when it is encountered.

### Display Help

Check that the correct help is displayed when “How do I use this Expert” is selected.

### Specific Tests

Develop a set of test cases to validate the intended function of the expert.