



Adding custom MIBS to NET+OS
V7.5 and above SNMP component

1 Document History

Version	Initials	Change Description	
V1.0	JZW	Initial entry	
V2.0	JZW	Add more details & screen shots	

2 Table of Contents

1	Document History	2
2	Table of Contents	3
3	Introduction	4
3.1	Problem Solved	4
3.2	Audience	4
3.3	Assumptions	4
3.4	Scope	5
3.5	Theory of Operation	5
4	Basics	6
4.1	SNMP	6
4.2	MIBS	6
4.2.1	Converting MIBS into C code using Digi ESP IDE	8
4.2.2	Converting MIBs into C using The GNU Command Line Interface	14
4.3	Integrating the Generated Code into Your application	18
4.4	Next Steps – build and deploy	24
4.5	Connections to Device Data	24
5	Compiling MIBs into the MIB browser	25
6	Conclusion	25
7	Glossary of Terms	25

3 Introduction

This white paper contains a more detailed explanation of the methods and procedures required for adding a custom MIB to NET+OS V7.X SNMP component. This includes both converting your MIB into C code and adding callback functions to the application thus giving the SNMP component of NET+OS access to your device data. For a quick overview of the process required for adding MIBs to your application, Digi recommends that you read the readme file associated with the example application entitled snmpv3.

Additionally, starting with NET+OS development environment V7.5 we have made two changes to the SNMP component, making it easier to add custom MIBs to your product. First, you will no longer have to write code to implement your MIB in the NET+OS V7.5 SNMP agent. That is done for you by the MIB to C compiler. Second, for customers developing using Digi ESP, an SNMP wizard has been added to Digi ESP, allowing you to compile your project's C code, MIB and your web pages (if you had them) all from within Digi ESP.

3.1 Problem Solved

Integrating SNMP into an embedded device is not easy. The protocols and standards integrated into SNMP are made to make interfacing into SNMP generic and flexible but in doing so, understanding the how to do this is complex. The NET+OS SNMP component has gone a long way to separating the work of accessing your device data from the mechanics of integrating your data into the SNMP data structures.

This white paper shows that using NET+OS V7.5's (and above) SNMP component, following the directions already contained within the NET+OS documentation set and using some additional details laid out here, to integrate your data into SNMP is just not that hard. We would like to think that when you have completed reading this paper and looking at the associated sample application and sample MIB that you'll say, "Oh is that all there is!"

3.2 Audience

This paper is of a technical nature. It is intended for software engineers and software practitioners with knowledge of SNMP, MIBs, NET+OS V7.5 (and above) and development experience in GNU (CLI) or GNU/ESP environments. In order to debug an SNMP-based application you will also need to have some knowledge of the TCP/IP protocol stack and the use of a network protocol analyzer.

3.3 Assumptions

This paper assumes that you have access to a GNU NET+OS V7.5 development environment. We do not believe you will take in as much by just passively reading this document as you will by reading it and trying out the sample application simultaneously. Additionally, we presume that you have access to a MIB browser. You will require a MIB browser in order to access the SNMP component of NET+OS. The MIB browser

must also have the ability to integrate new MIBs into its MIB database (sometimes called MIB compiling). We do not make recommendations of any particular MIB browsers as they are developed by third parties. The three we have used are MGSoft by <who makes it>. iReasonings <product> and the command line interface product net-snmp by <whom>.

3.4 Scope

This document adds additional details to already existing SNMP and MIB-related NET+OS V7.5 (and above) documentation. It is intended to help customers get “over the hump” of adding a custom MIB to their application. The feeling is that once a customer has added one, adding additional custom MIBs should be that much easier.

The following is a list of items and subjects that this document does NOT cover:

- Instructions in C programming
- Instructions on the details of building (creating) a custom MIB
- Instructions on adding customer MIBs to a version of the NET+OS development environment available prior to V7.5. Versions 7.4, 7.3 and earlier are NOT covered in this document.
- Details of the SNMP protocol
- Details of ASN1
- Instructions of developing NET+OS applications under Green Hills or GNU
- Instructions on running a MIB browser or compiler
- Instructions on adding a MIB to your MIB browser.
- Instructions on TCP/IP protocols
- Instructions on using a network protocol analyzer
- Instructions on downloading NET+OS applications into a Digi device
- Anything referencing .NET or LINUX

Our feeling is that there are a plethora of texts dealing with these subjects and taking time to cover these subjects is outside the scope of this document.

If you would like additional information in some of these areas, the following are some texts that we would recommend:

Understanding SNMP MIBS by Perkins & McGinnis

Managing Internetworks with SNMP by Miller

Internetworking with TCP/IP by Comer

The Digi web site <http://www.digi.com>

3.5 Theory of Operation

Device data is your data. You created it; you own it and you understand it. It is not the purpose of the SNMP component of NET+OS to have any understanding of your data. What the SNMP component does do is act as a formatter and a conduit for your data between your Digi device and an SNMP MIB browser.

SNMP uses a complicated protocol for formatting and transferring data between your device (called an SNMP agent) and your SNMP management station (generally an SNMP Browser). The NET+OS SNMP component abstracts the ASN1 formatting complexities from your application. Thus you do not have to worry about it. All you do have to worry about is physically accessing your data, through callback functions, and returning that data to an unknown caller. If you have used the “stub function” callbacks that the AWS component utilizes, these MIB-related callback functions are not that far removed from the AWS-related stub functions. Please note that in the NET+OS development environment starting with version 7.5, these functions are generated and filled in for you.

So what's required? You need to write your MIB. Your MIB is where this entire process starts. You'll need to run your MIB through Digi's MIB to C utility (mib2c). The mib2c utility reads in your MIB file and generates a number of C and H files, specifically geared to your application and your MIB. You'll need to make some edits (to your code) that are explained in a file created as a result of mib2c converting your MIB (<mib file name>.ins). Once you have completed that, you'll need to compile and build your application, with the SNMP component included. Before downloading your application into your device, for the first time, you'll need to “compile” your MIB into your MIB browser. You will save yourself a lot of time by performing this step before attempting to debug your application.

What should happen? You'll point your MIB browser at your device. You'll connect to your device. You'll tell your MIB browser to walk through your device's MIB.

4 Basics

4.1 SNMP

Simple Network Management Protocol (SNMP) has become the protocol used for managing network-connected devices. SNMP is layered on top of UDP/IP. SNMP can be broken into two primary components, namely managers and agents. The manager might be the user using a MIB browser to access some device the user wishes to manage (control in some way). The agent is a software layer, generally residing within a device that accesses device data.

The device being managed contains objects. The objects are housed in databases, accessible by the manager through the agent. These databases are arranged in logical structures entitled management information bases (MIBs).

4.2 MIBS

As mentioned above, your device data is contained in data structures entitled MIBs. A MIB is a very precisely defined structured document that you create. It is editable with a text editor. The structure, layout and keyword set of a MIB are beyond the scope of this white paper. To further understand MIBs we recommend the book Understanding SNMP MIBs by Perkins and McGinnis and published by Prentice-Hall PTR. Your MIB(s) is used by the mib2c utility to convert your MIB into C code that can then be integrated into

Adding custom MIBs to NET+OS V7.x's SNMP component

your application. The MIB is used to generate data structures and functions used to access the device data that the MIB represents to the agent.

Starting with NET+OS development environment V7.5, we have greatly reduced the steps required for you to integrate your MIB into your application. You are no longer required to write callback functions for implementing get & set. Also, if you'd like to initialize a conceptual table with values, we provide an easy way of doing so. Converting your MIB to C in the Digi ESP IDE environment, in the command line (gnu) environment and setting up default values are covered in the next few sections of this paper.

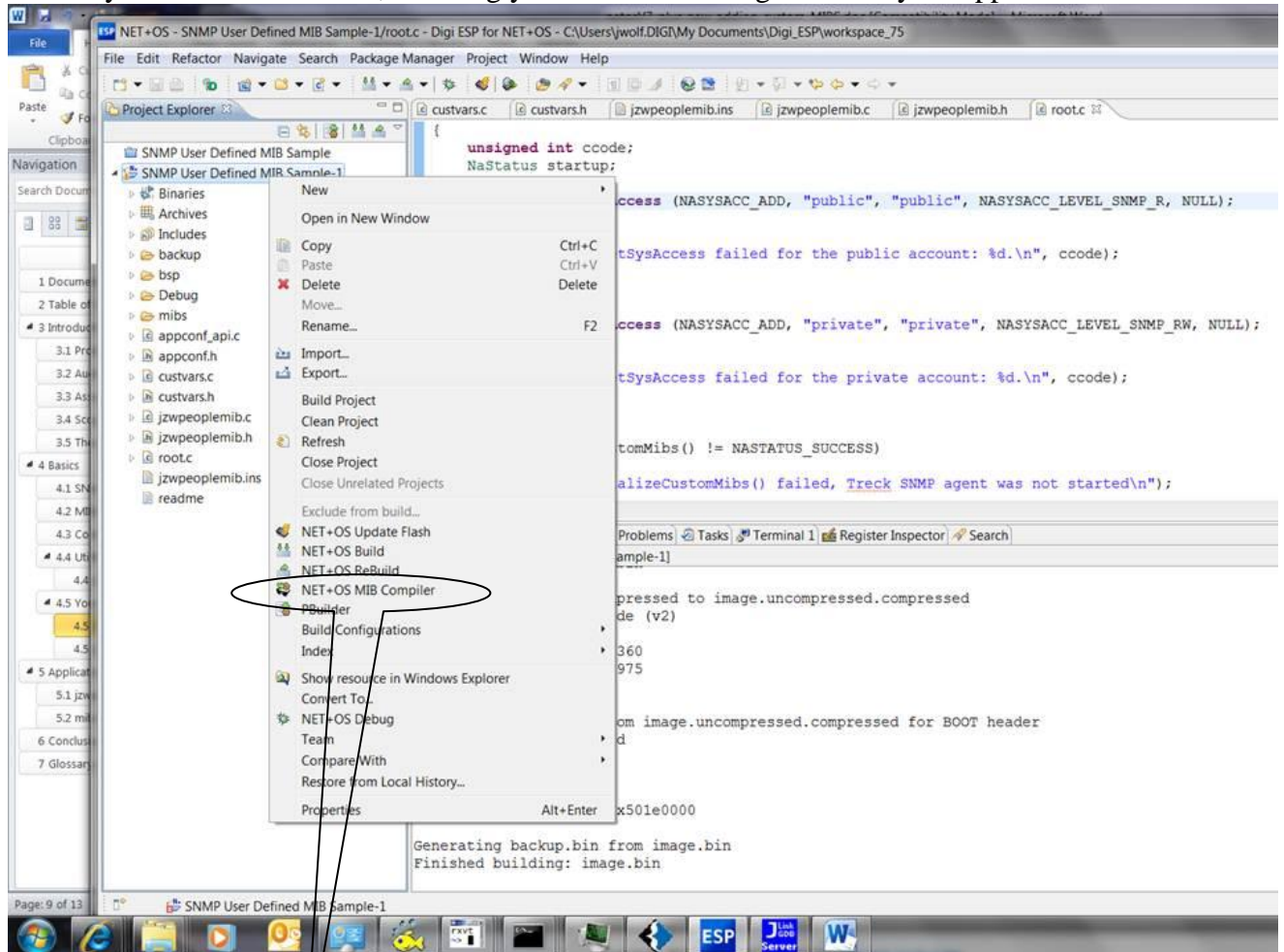
Sample code can be found at the following links –

[Digi ESP Project Code](#)

[Command Line Application Code](#)

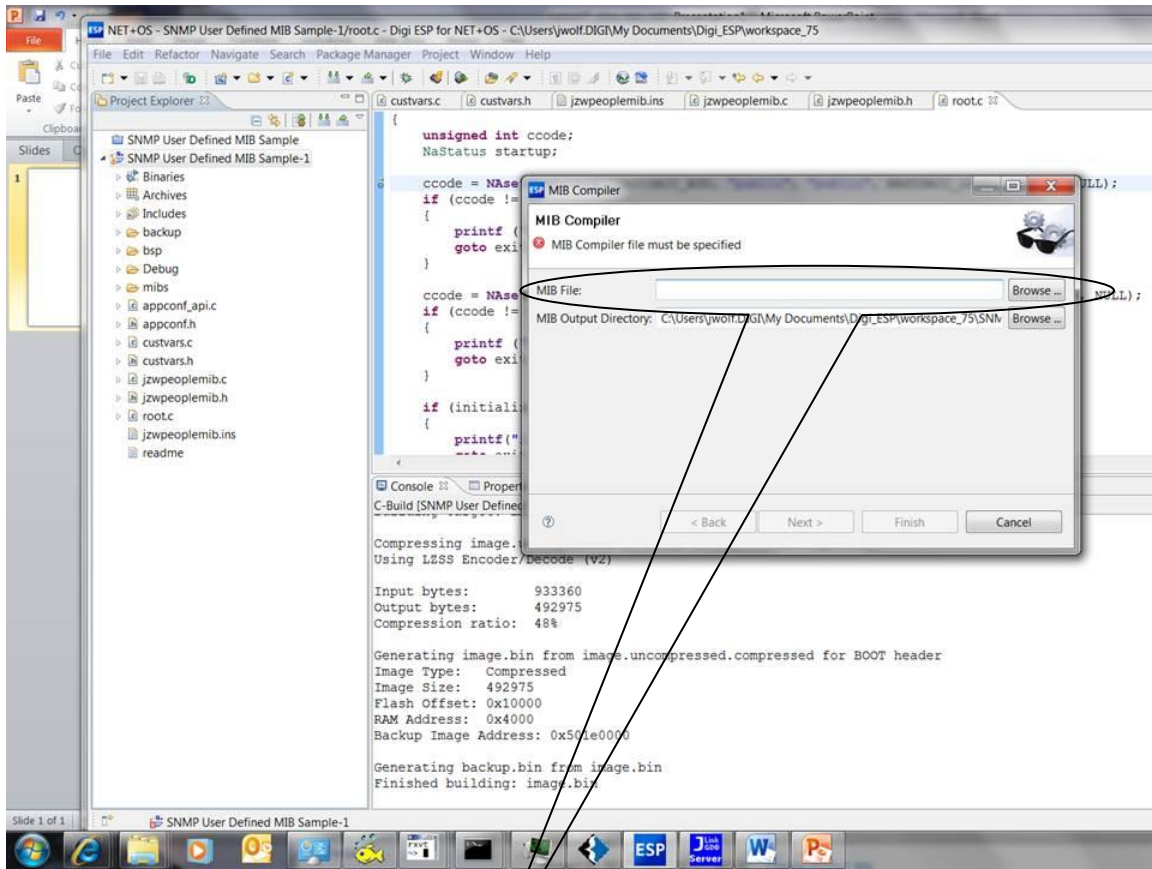
4.2.1 Converting MIBS into C code using Digi ESP IDE

Using a number of screen shots this section walks you through the steps required to convert your MIB into C code, allowing your MIB to be integrated into your application.



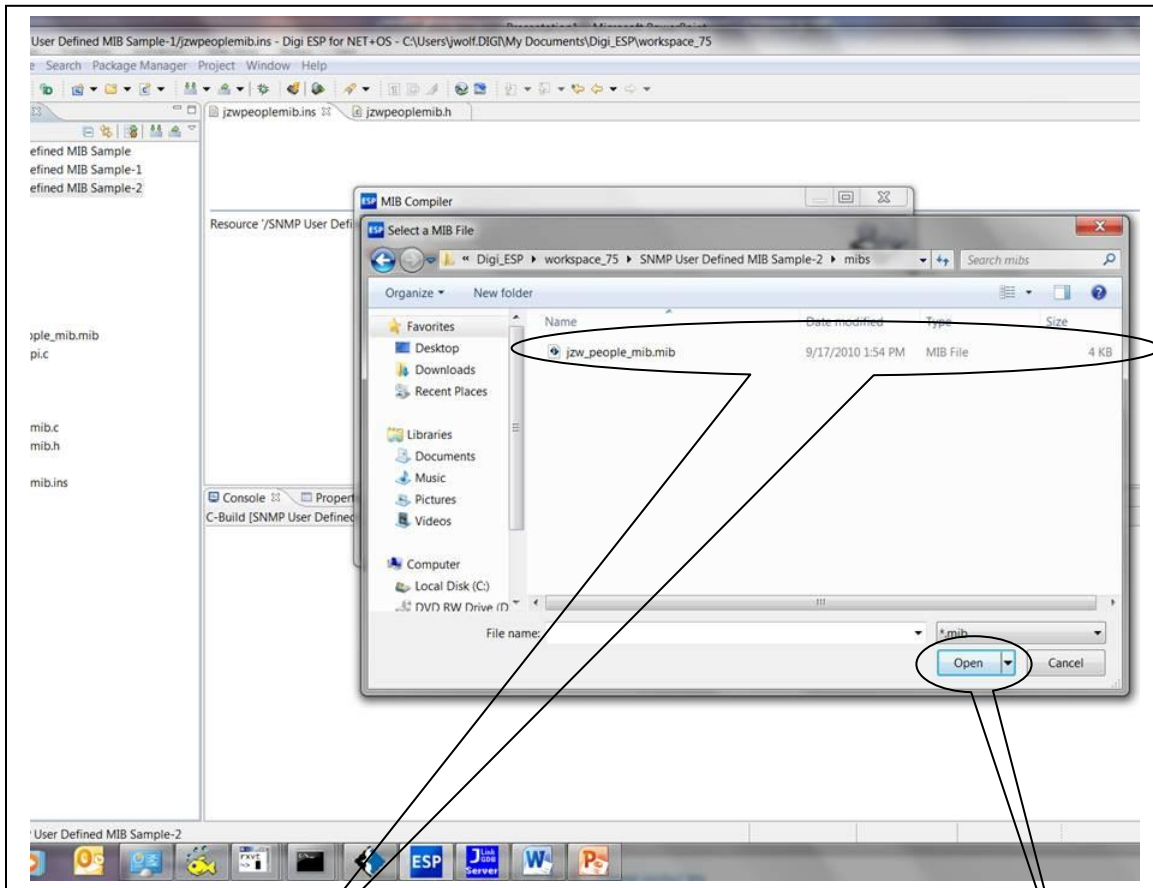
Right click on your project name and the pull-down menu shown here will appear. You'll be selecting the NET+OS MIB Compiler option.

Adding custom MIBs to NET+OS V7.x's SNMP component



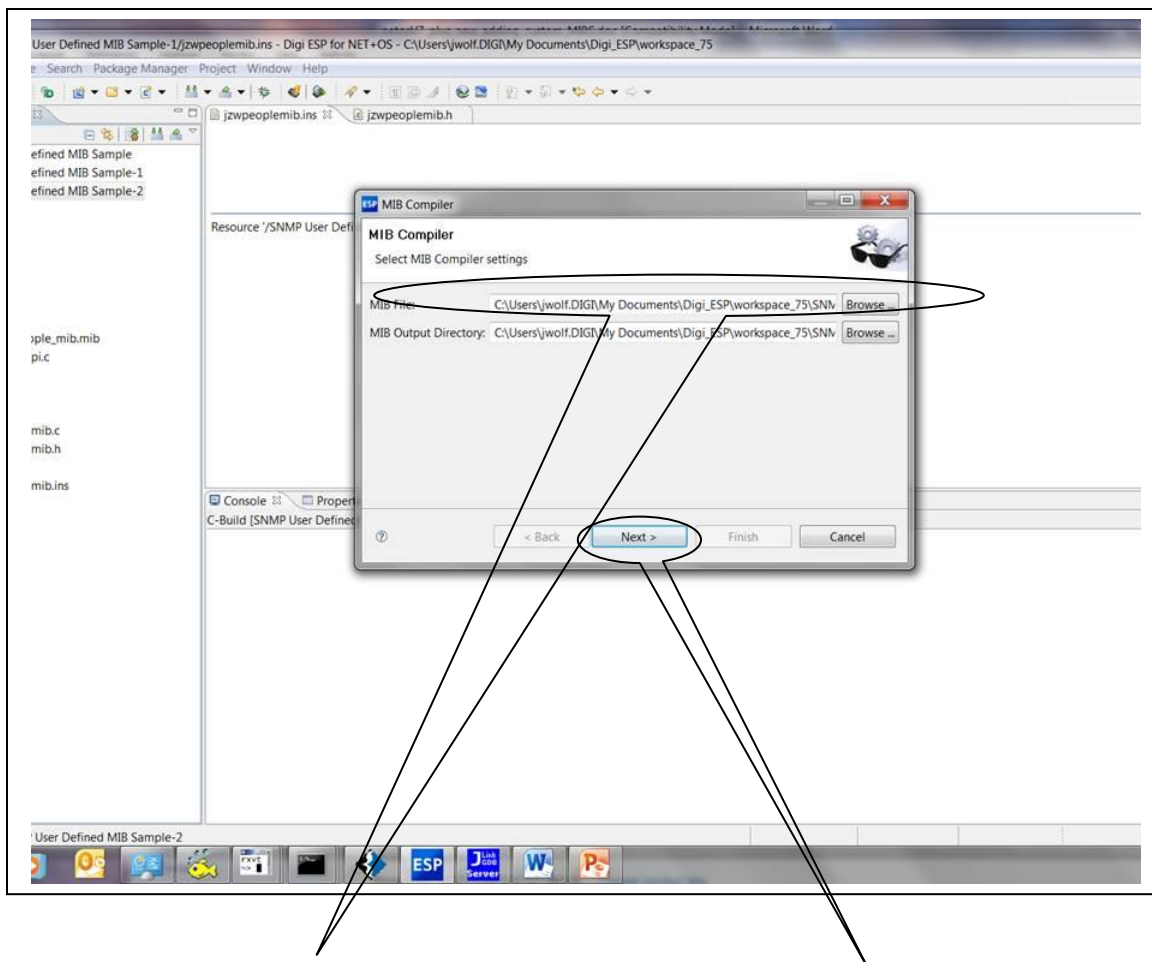
You'll need to supply the name of the MIB to be converted. You can use the file browse button to select your MIB.

Adding custom MIBs to NET+OS V7.x's SNMP component



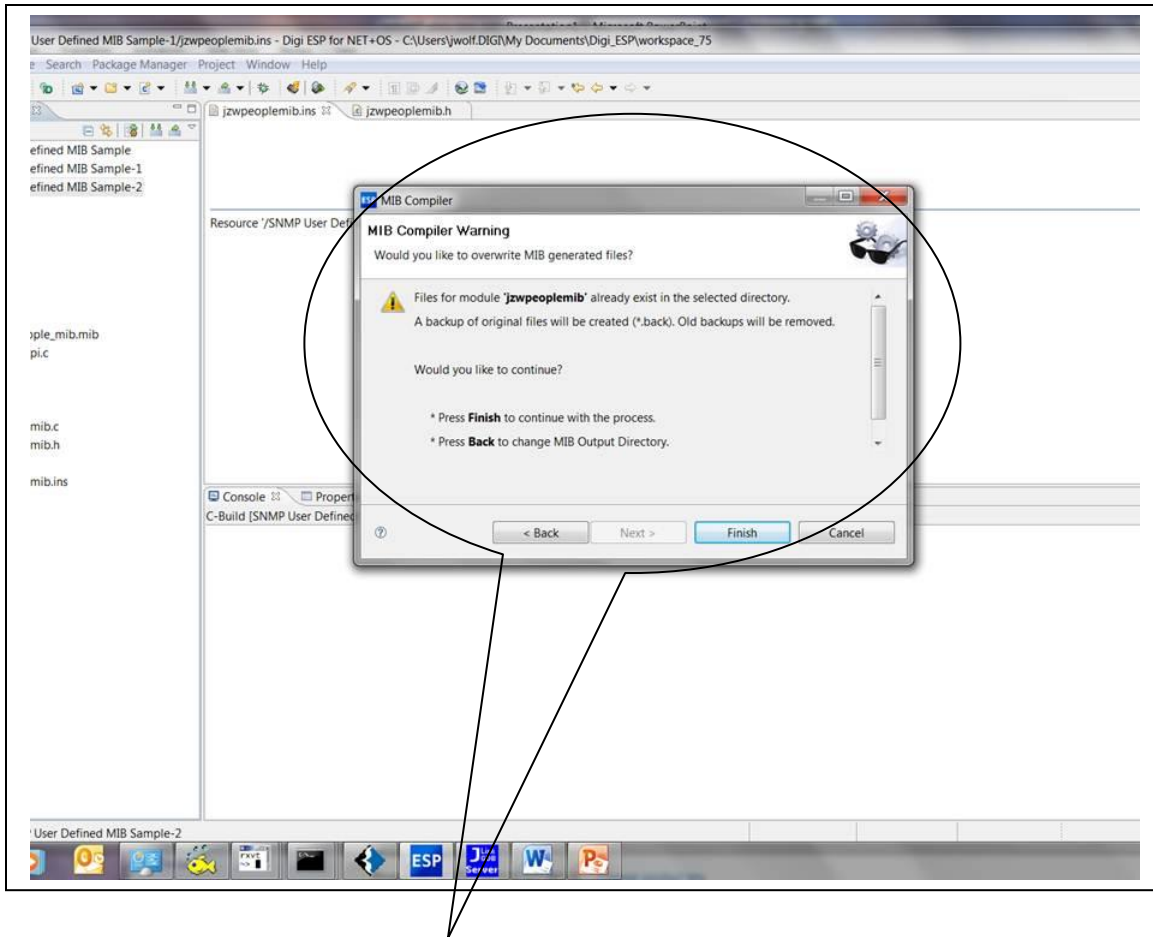
You should see the MIB file that you requested. Select the MIB file and click Open.

Adding custom MIBs to NET+OS V7.x's SNMP component



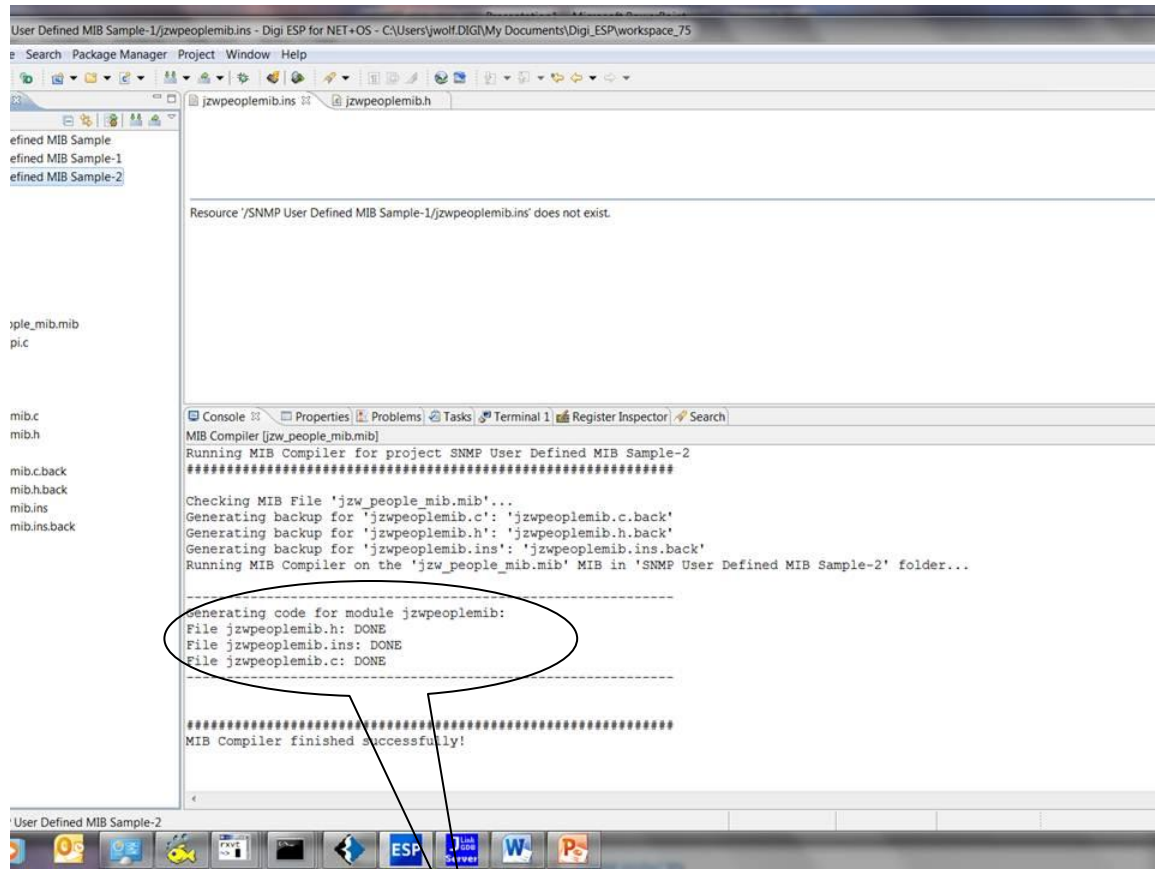
With your MIB file name filled into the appropriate box, click next.

Adding custom MIBs to NET+OS V7.x's SNMP component



If you had previously converted your MIB file in this directory, you will get this pop-up window. We will back-up your generated files before continuing. Or you can choose to stop now and move files around.

Adding custom MIBs to NET+OS V7.x's SNMP component

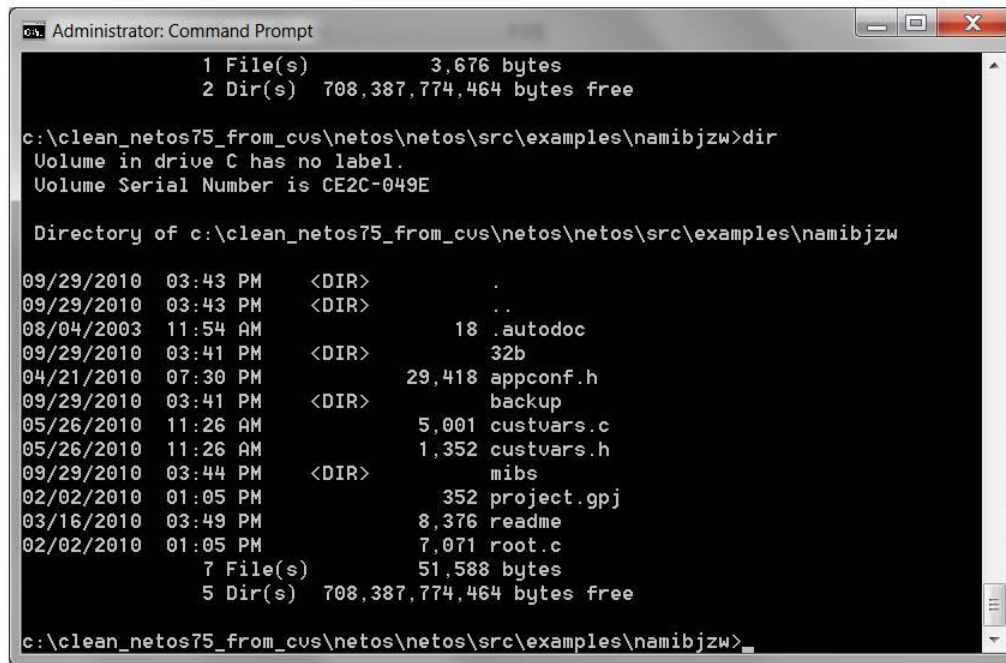


If you clicked the finish key, as shown on the last slide, and if your MIB file is error free, the output should look similar to the screenshot above. Please notice that the files generated are listed at the bottom of the screen.

4.2.2 Converting MIBs into C using The GNU Command Line Interface

The last section explained converting your MIB file into C code using the Digi ESP IDE. Just in case you develop using the command line interface, this section describes performing the same steps as the last section showed, but using the command line interface as opposed to using the Digi ESP IDE.

Adding custom MIBs to NET+OS V7.x's SNMP component



```
Administrator: Command Prompt

1 File(s)      3,676 bytes
2 Dir(s)  708,387,774,464 bytes free

c:\clean_netos75_from_cvs\netos\netos\src\examples\namibjzw>dir
Volume in drive C has no label.
Volume Serial Number is CE2C-049E

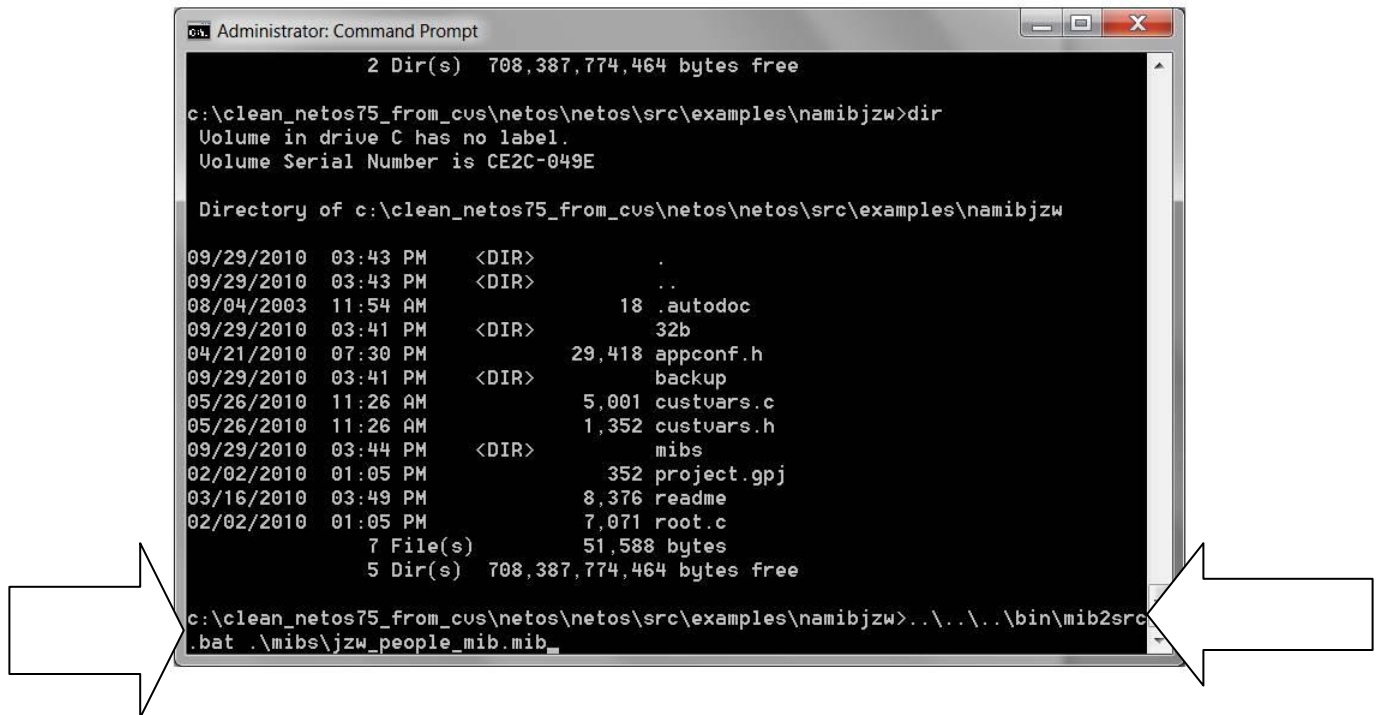
Directory of c:\clean_netos75_from_cvs\netos\netos\src\examples\namibjzw

09/29/2010  03:43 PM    <DIR>          .
09/29/2010  03:43 PM    <DIR>          ..
08/04/2003  11:54 AM             18 .autodoc
09/29/2010  03:41 PM    <DIR>          32b
04/21/2010  07:30 PM      29,418 appconf.h
09/29/2010  03:41 PM    <DIR>          backup
05/26/2010  11:26 AM       5,001 custvars.c
05/26/2010  11:26 AM       1,352 custvars.h
09/29/2010  03:44 PM    <DIR>          mibs
02/02/2010  01:05 PM       352 project.gpj
03/16/2010  03:49 PM       8,376 readme
02/02/2010  01:05 PM       7,071 root.c
              7 File(s)      51,588 bytes
              5 Dir(s)  708,387,774,464 bytes free

c:\clean_netos75_from_cvs\netos\netos\src\examples\namibjzw>
```

Before starting the conversion process, your directory probably looks something like the screen shot above.

Adding custom MIBs to NET+OS V7.x's SNMP component



```
Administrator: Command Prompt

2 Dir(s) 708,387,774,464 bytes free

c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw>dir
Volume in drive C has no label.
Volume Serial Number is CE2C-049E

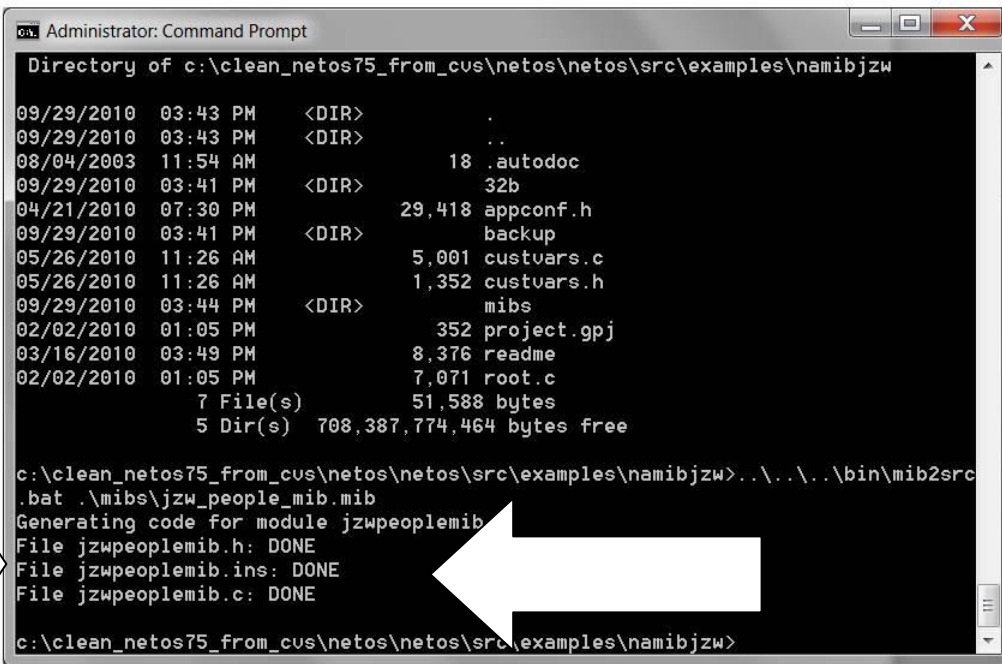
Directory of c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw

09/29/2010 03:43 PM <DIR>          .
09/29/2010 03:43 PM <DIR>          ..
08/04/2003 11:54 AM          18      .autodoc
09/29/2010 03:41 PM <DIR>          32b
04/21/2010 07:30 PM      29,418  appconf.h
09/29/2010 03:41 PM <DIR>          backup
05/26/2010 11:26 AM      5,001  custvars.c
05/26/2010 11:26 AM      1,352  custvars.h
09/29/2010 03:44 PM <DIR>          mibs
02/02/2010 01:05 PM      352  project.gpj
03/16/2010 03:49 PM      8,376  readme
02/02/2010 01:05 PM      7,071  root.c
              7 File(s)      51,588 bytes
              5 Dir(s) 708,387,774,464 bytes free

c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw>..\..\..\bin\mib2src
.bat .\mibs\jzw_people_mib.mib_
```

In this screen shot we are ready to execute the MIB to C converter entitled mib2c.bat. It is located in netos\bin\mib2c.bat. You can place the entire path in your command line (as I did here) or you can place netos\bin in your path environment variable. Either way, by default, the MIB files are located in a directory entitled mibs. Directory mibs is a subdirectory to your project's main directory.

Adding custom MIBs to NET+OS V7.x's SNMP component



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window displays the output of a directory listing for the path `c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw`. The listing shows various files and directories, including `.autodoc`, `32b`, `appconf.h`, `backup`, `custvars.c`, `custvars.h`, `mibs`, `project.gpj`, `readme`, and `root.c`. Below the listing, the command `c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw>..\..\..\bin\mib2src.bat .\mibs\jzw_people_mib.mib` is executed. The output shows the generation of code for the module `jzwpeoplemib`, with the files `jzwpeoplemib.h`, `jzwpeoplemib.ins`, and `jzwpeoplemib.c` all marked as "DONE". A large white arrow points from the text below to the command prompt output, and a smaller white arrow points from the left margin to the command prompt output.

```
Administrator: Command Prompt

Directory of c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw

09/29/2010  03:43 PM    <DIR>          .
09/29/2010  03:43 PM    <DIR>          ..
08/04/2003  11:54 AM             18      .autodoc
09/29/2010  03:41 PM    <DIR>          32b
04/21/2010  07:30 PM      29,418  appconf.h
09/29/2010  03:41 PM    <DIR>          backup
05/26/2010  11:26 AM       5,001  custvars.c
05/26/2010  11:26 AM       1,352  custvars.h
09/29/2010  03:44 PM    <DIR>          mibs
02/02/2010  01:05 PM        352  project.gpj
03/16/2010  03:49 PM       8,376  readme
02/02/2010  01:05 PM       7,071  root.c
              7 File(s)      51,588 bytes
              5 Dir(s)  708,387,774,464 bytes free

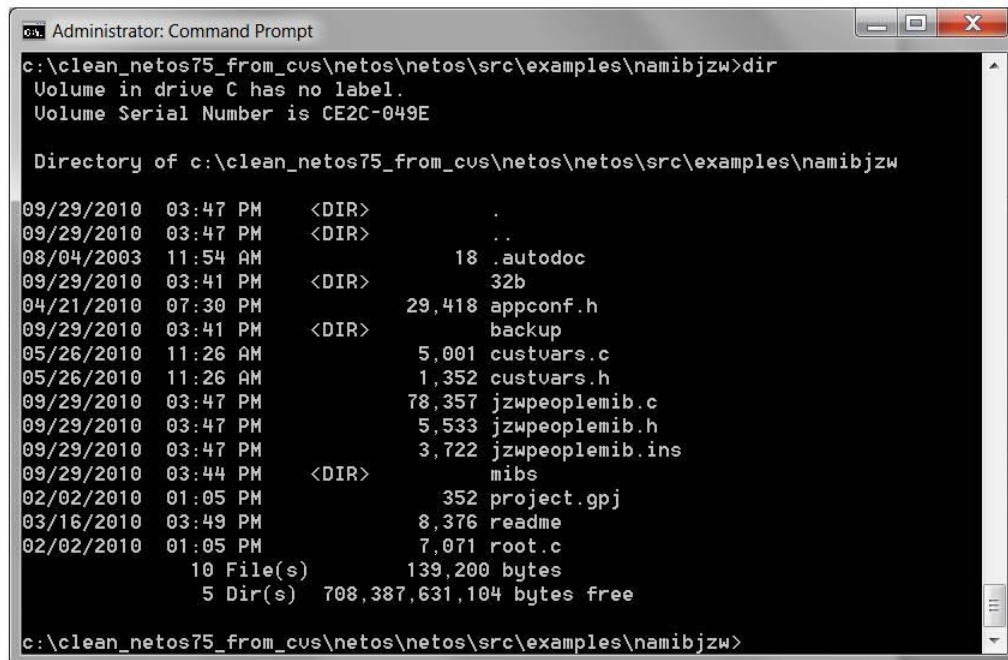
c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw>..\..\..\bin\mib2src
.bat .\mibs\jzw_people_mib.mib
Generating code for module jzwpeoplemib
File jzwpeoplemib.h: DONE
File jzwpeoplemib.ins: DONE
File jzwpeoplemib.c: DONE

c:\clean_netos75_from_cus\netos\netos\src\examples\namibjzw>
```

Assuming your MIB file contained no errors and it was available, the output from executing `mib2c.bat` should look something like the screenshot above. Please notice that the files that `mib2c.bat` generated are listed at the bottom of the file. Also please notice that the same files are generated via the Digi ESP IDE (as seen previously) as are generated via the command line interface. So either way, you end up with the same files.

4.3 Integrating the Generated Code into Your application

Your MIB file is now converted into .c and .h files for integration into your application. Since at this point the step differences between using the Digi ESP IDE and the GNU command line interface are similar enough, the remainder of the descriptions will be contained in one section.



```

Administrator: Command Prompt
c:\clean_netos75_from_cvs\netos\netos\src\examples\namibjzw>dir
Volume in drive C has no label.
Volume Serial Number is CE2C-049E

Directory of c:\clean_netos75_from_cvs\netos\netos\src\examples\namibjzw

09/29/2010  03:47 PM    <DIR>          .
09/29/2010  03:47 PM    <DIR>          ..
08/04/2003  11:54 AM             18 .autodoc
09/29/2010  03:41 PM    <DIR>          32b
04/21/2010  07:30 PM      29,418 appconf.h
09/29/2010  03:41 PM    <DIR>          backup
05/26/2010  11:26 AM       5,001 custvars.c
05/26/2010  11:26 AM       1,352 custvars.h
09/29/2010  03:47 PM      78,357 jzwpeoplemib.c
09/29/2010  03:47 PM       5,533 jzwpeoplemib.h
09/29/2010  03:47 PM       3,722 jzwpeoplemib.ins
09/29/2010  03:44 PM    <DIR>          mibs
02/02/2010  01:05 PM        352 project.gpj
03/16/2010  03:49 PM      8,376 readme
02/02/2010  01:05 PM      7,071 root.c
               10 File(s)      139,200 bytes
               5 Dir(s)  708,387,631,104 bytes free

c:\clean_netos75_from_cvs\netos\netos\src\examples\namibjzw>
  
```

Whether you performed the conversion using the Digi ESP IDE or the GNU command line interface, your project directory should look something like the screenshot above. Please notice that three files have been added to the directory, namely jzwpeoplemib.c, jzwpeoplemib.h and jzwpeoplemib.ins.

Adding custom MIBs to NET+OS V7.x's SNMP component

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt - more jzwpeplembins". The window contains two SQL INSERT statements. A large white arrow points from the label "Instructions" to the first INSERT statement. Another large white arrow points from the label "Code block" to the second INSERT statement.

```

INSERT INTO custvars.c BEFORE LINE: const ttSnmpSubtree TM_CONST_QLF custSnmpSubtrees[] =

const struct tsSnmpVariable tuSnmpPeopletableUars[] = {
    (tfvar_peopletable_entry, TM_SNMP_READONLY, 0, 2, PEINDEX, TM_SNMP_INTEGER, TM_8BIT_NO, (1.1)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PENAME, TM_SNMP_STRING, TM_8BIT_NO, (1.2)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PESTREETADDRESS, TM_SNMP_STRING, TM_8BIT_NO, (1.3)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PECITY, TM_SNMP_STRING, TM_8BIT_NO, (1.4)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PESTATE, TM_SNMP_STRING, TM_8BIT_NO, (1.5)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PEZIPCODE, TM_SNMP_STRING, TM_8BIT_NO, (1.6)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PEHOUSETYPE, TM_SNMP_INTEGER, TM_8BIT_NO, (1.7)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PEHOUSEMORTH, TM_SNMP_INTEGER, TM_8BIT_NO, (1.8)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PEDATEPURCHASED, TM_SNMP_STRING, TM_8BIT_NO, (1.9)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PEMAILADDRESS, TM_SNMP_STRING, TM_8BIT_NO, (1.10)),
    (tfvar_peopletable_entry, TM_SNMP_RCREATE, 0, 2, PEROWSTATUS, TM_SNMP_INTEGER, TM_8BIT_YES, (1.11))
};

INSERT INTO custvars.c into const ttSnmpSubtree TM_CONST_QLF custSnmpSubtrees[]
in any order before an ending empty structure:

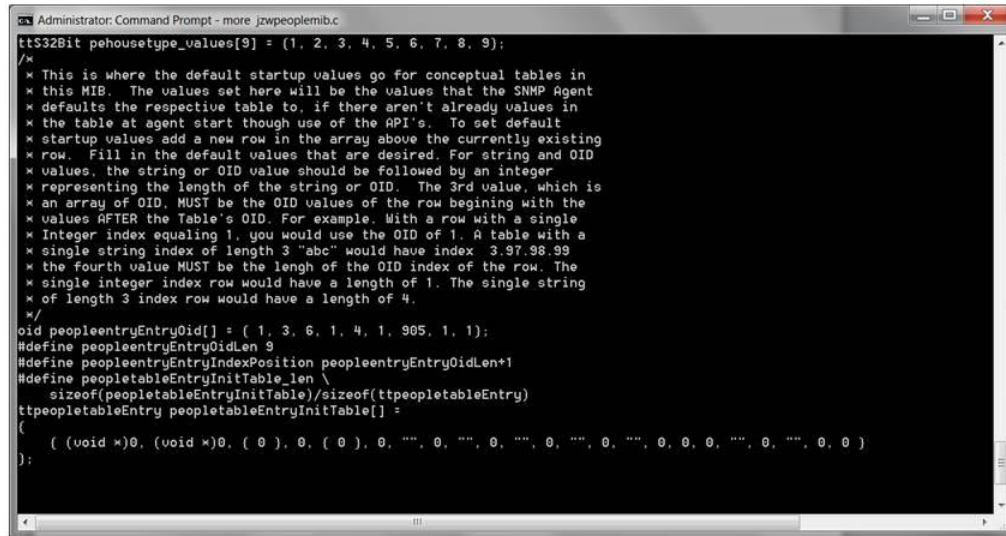
((ttSnmpVariablePtr)tuSnmpPeopletableUars,
    {1.3, 6, 1, 4, 1, 905, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    sizeof(tuSnmpPeopletableUars)/sizeof(*tuSnmpPeopletableUars),
    sizeof(*tuSnmpPeopletableUars), 8, 0, ('\0')).

```

The screen shot above shows some of the contents of the .ins file generated by running the mib2.c utility (either from the command line or from the IDE). You will need to perform each step using your favorite text editor. You will be grabbing blocks of code from the .ins file and placing those blocks, as directed, into the file to which you are directed. Please notice that ABOVE each block of code to be moved is listed the file into which the block is to be moved and the place within that file where that block should be placed.

This step should be performed with care. If it is performed incorrectly, your project will fail to compile. Please follow and perform each and every step listed in the .ins file.

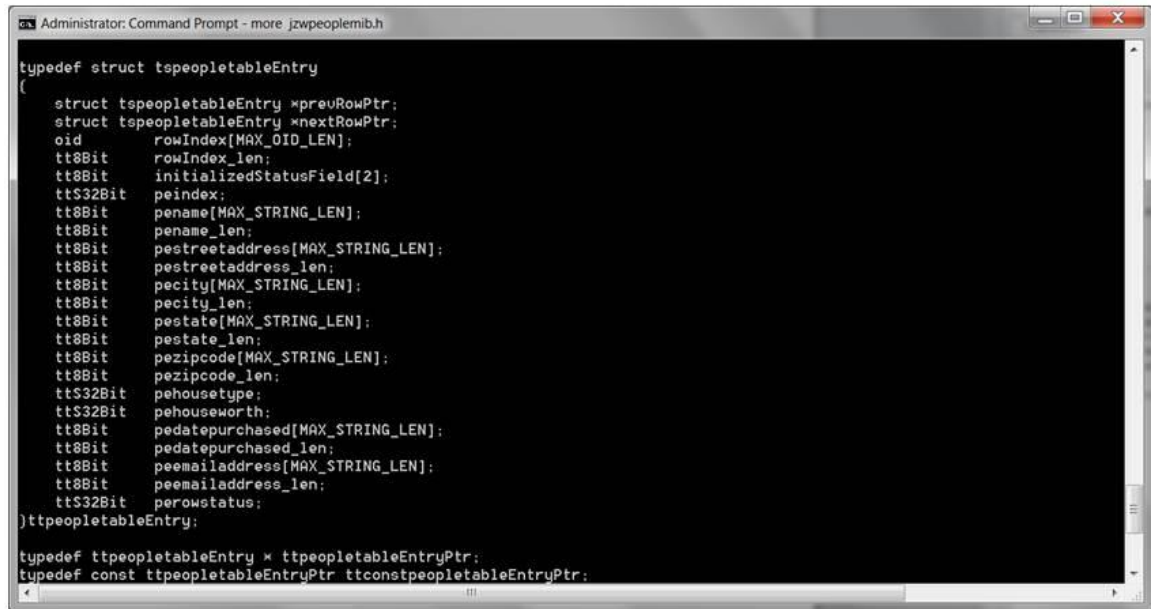
Adding custom MIBs to NET+OS V7.x's SNMP component



```
Administrator: Command Prompt - more jzwpeplemb.c
tt$32Bit pehousetype_values[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
/*
 * This is where the default startup values go for conceptual tables in
 * this MIB. The values set here will be the values that the SNMP Agent
 * defaults the respective table to, if there aren't already values in
 * the table at agent start though use of the API's. To set default
 * startup values add a new row in the array above the currently existing
 * row. Fill in the default values that are desired. For string and OID
 * values, the string or OID value should be followed by an integer
 * representing the length of the string or OID. The 3rd value, which is
 * an array of OID, MUST be the OID values of the row beginning with the
 * values AFTER the Table's OID. For example. With a row with a single
 * Integer index equaling 1, you would use the OID of 1. A table with a
 * single string index of length 3 "abc" would have index 3.97.98.99
 * the fourth value MUST be the length of the OID index of the row. The
 * single integer index row would have a length of 1. The single string
 * of length 3 index row would have a length of 4.
 */
oid peopleentryEntryOid[] = { 1, 3, 6, 1, 4, 1, 905, 1, 1};
#define peopleentryEntryOidLen 9
#define peopleentryEntryIndexPosition peopleentryEntryOidLen+1
#define peopletableEntryInitTable_len \
    sizeof(peopletableEntryInitTable)/sizeof(ttpeopletableEntry)
ttpeopletableEntryInitTable[] =
{
    ( (void *)0, (void *)0, ( 0 ), 0, ( 0 ), 0, "", 0, "", 0, "", 0, "", 0, 0, 0, "", 0, "", 0, 0 )
};
```

The next couple of slides refer to the .c file. If you want to have a conceptual table filled with default values, you will need to fill in the table as seen above. The lines added will be added ABOVE the existing line. Do not remove the default line. Edit this file using your favorite text editor.

Adding custom MIBs to NET+OS V7.x's SNMP component



```
Administrator: Command Prompt - more jzwpeoplemib.h

typedef struct tspeopletableEntry
(
    struct tspeopletableEntry *prevRowPtr;
    struct tspeopletableEntry *nextRowPtr;
    oid                        rowIndex[MAX_OID_LEN];
    tt8Bit                    rowIndex_len;
    tt8Bit                    initializedStatusField[2];
    ttS32Bit                  peindex;
    tt8Bit                    pename[MAX_STRING_LEN];
    tt8Bit                    pename_len;
    tt8Bit                    pestreetaddress[MAX_STRING_LEN];
    tt8Bit                    pestreetaddress_len;
    tt8Bit                    pecity[MAX_STRING_LEN];
    tt8Bit                    pecity_len;
    tt8Bit                    pestate[MAX_STRING_LEN];
    tt8Bit                    pestate_len;
    tt8Bit                    pezipcode[MAX_STRING_LEN];
    tt8Bit                    pezipcode_len;
    ttS32Bit                  pehouseworth;
    ttS32Bit                  pehouseworth;
    tt8Bit                    pedatepurchased[MAX_STRING_LEN];
    tt8Bit                    pedatepurchased_len;
    tt8Bit                    peemailaddress[MAX_STRING_LEN];
    tt8Bit                    peemailaddress_len;
    ttS32Bit                  perowstatus;
)tspeopletableEntry;

typedef tspeopletableEntry * tspeopletableEntryPtr;
typedef const tspeopletableEntryPtr ttconstpeopletableEntryPtr;
```

This shot shows the data structure that the agent uses for dealing with the data in the MIB. We will be referring to this structure as we look at filling in the fields of the default MIB records, as referred to on the last slide. You might want print this page out and keep it handy.

Adding custom MIBs to NET+OS V7.x's SNMP component

Let's assume that I wanted to add a default table entry for Mickey Mouse. I would add a line as follows:

```
{(void)0, (void)0, {1}, 1, { peopletableEntry_allbitsLast }, 1, "Mickey Mouse", 12, "220 East Cheese Lane", 20, "Orlando", 7, "FL", 2, "98776", 5, 1, 500000, "1/1/1955", 8, "mickey.mouse@dland.com", 22, TM_SNMP_ROWSTATUS_ACTIVE },
```

And the one for Donald Duck might look like the following:

```
{(void)0, (void)0, {2}, 1, { peopletableEntry_allbitsLast }, 2, "Donald Duck", 11, "230 West WebbedFoot Rd", 22, "Orlando", 7, "FL", 2, "98776", 5, 2, 550000, "1/1/1955", 8, "donald.duck@dland.com", 22, TM_SNMP_ROWSTATUS_ACTIVE },
```

Let's assume that I wanted to add a default table entry for Mickey Mouse. I would add a line as follows:

```
{(void)0, (void)0, {1}, 1, { peopletableEntry_allbitsLast }, 1, "Mickey Mouse", 12,
"220 East Cheese Lane", 20, "Orlando", 7, "FL", 2, "98776", 5, 1, 500000, "1/1/1955", 8,
"mickey.mouse@dland.com", 22, TM_SNMP_ROWSTATUS_ACTIVE },
```

And the one for Donald Duck might look like the following:

```
{(void)0, (void)0, {2}, 1, { peopletableEntry_allbitsLast }, 2, "Donald Duck", 11,
"230 West WebbedFoot Rd", 22, "Orlando", 7, "FL", 2, "98776", 5, 2, 550000, "1/1/1955", 8,
"donald.duck@dland.com", 22, TM_SNMP_ROWSTATUS_ACTIVE },
```

I have included the text from the last page, to allow us to give it further scrutiny. We will concentrate on the table entry added for Mickey Mouse. First, the easy fields. All text fields are surrounded with double quotation marks. After each text element is a number. The number represents the number of bytes in the text. With "Mickey Mouse", for example, there are 12 bytes between the quotations marks.

The 1 between the 5 and the 500000 is a house type. House types are defined in the MIB. 500000 is the house worth. It is an integer number.

The first two fields are used internally. Leave them as is.

The third field is {1} for Mickey and {2} for Donald. These are indices. For a standard MIB these are integers increasing by 1 starting with 1. The fourth field represents the length of the third field. If you are indexing with something other than integers then this field will vary. In our case this is 1 byte.

The fifth field will contain a definition from your .h file. Use the definition containing allbitsLast. Notice that this field is contained within {and}.

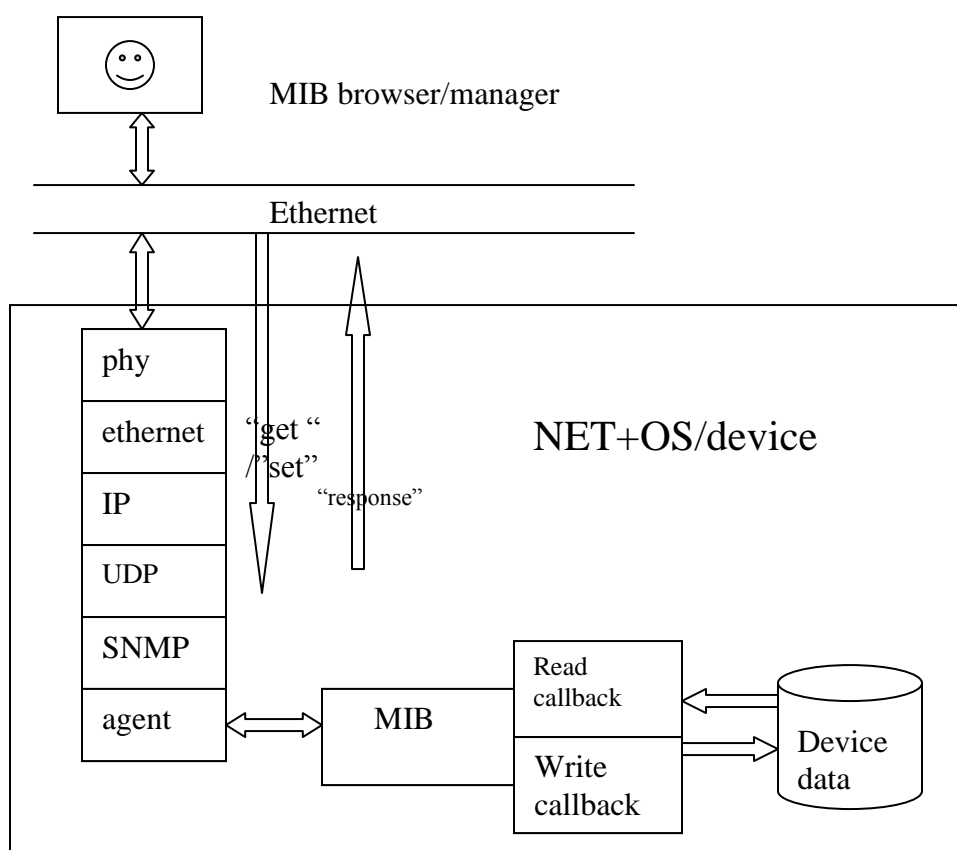
The last field is row status (I placed my row status field as the last field in my MIB. This was my decision.). For MIB entries added internally as opposed to being added by a MIB browser, we recommend filling this field with TM_SNMP_ROWSTATUS_ACTIVE

4.4 Next Steps – build and deploy

You have successfully generated the .c and .h files that represent your MIB to the SNMP agent. You have integrated your code into the application code. Next you must build your application. Assuming your build is successful, you should be prepared to test out your MIB in your device.

4.5 Connections to Device Data

The following diagram describes the parts of an SNMP application:



5 Compiling MIBs into the MIB browser

Before you can successfully walk your MIB, including your newly included MIB, you need to perform the task of “compiling” your MIB into your MIB browser. In the book “Understanding SNMP MIBS”, Perkins refers to this as a “backend MIB compiler” built into a MIB browser. The purpose of this step is to allow the MIB browser to access the definition objects in the MIB in a more expeditious manner.

The specifics of MIB browser-based backend MIB compilers is quite MIB browser specific and thus beyond the scope of this paper. You will, therefore, need to consult the help and/or documentation of your MIB browser to understand how it handles this activity. But I must emphasize that you should not begin attempting to walk your MIB, or looking for your new MIB until you have performed this task.

6 Conclusion

If you had suffered through SNMP development using the NET+OS development environment v 6.3, I am fairly sure you were happier but not completely satisfied with versions 7.0 – 7.4. These versions were improvements over V6.3 but still left the developer with a lot of code writing to do. I hope I have showed you that starting with V7.5, we have all but eliminated the necessity of writing code when implementing your custom MIB into your SNMP-enabled application and made it an easier process for you.

7 Glossary of Terms

CLI – Command line interface. A non-graphical method of accessing commands

ESP – Digi's graphical development environment

MIB – Management information base. A description of data layout used by SNMP for accessing user data

MIB Browser – A utility that is SNMP-aware and capable of exchanging SNMP messages with an SNMP-capable device

mib2c utility – A utility provided as part of the NET+OS V7.X product, that converts MIB files into C files for inclusion into a NET+OS SNMP-based application

NET+OS – A Digi produced embedded operating system and development environment for developing embedded applications for running on the family of Digi microprocessors and modules

Network protocol analyzer – either a piece of software or a software/hardware component that is capable of accessing (capturing) some or all packets on a LAN and displaying those packets in a formatted, human readable format. Examples are etherpeek, wireshark, ethereal

Adding custom MIBs to NET+OS V7.x's SNMP component

SNMP – Simple Network Management Protocol, a network protocol, which is generally encapsulated within UDP/IP and is used for managing objects on the network.

TCP/IP – Transmission Control Protocol/ Internet Protocol – A connection-based protocol that is part of the IP suite of network protocols

UDP/IP – User Datagram Protocol/Internet Protocol – A datagram-based protocol that is part of the IP suite of network protocols