**Using the NMPRK Library in Windows**

This document describes in detail how to build, test, integrate and use Intel® Node Manager Programmer’s Reference Kit. There are two different ways one can use the kit. Building an application using the existing per-built release libraries or build the libraries from the sources.

H/W requirements –

Host machine to compile sources

Intel® Server Platform with Intel® Node Manager Technology

S/W requirements –

Microsoft® Visual Studio

Linux – Red Hat, Ubuntu, SLES

Test setup –

**Host computer to compile and run Intel® Node Manager Programmer’s reference kit**

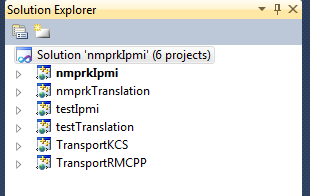
**Connected using Management LAN**

**Intel® Server with Intel® Node Manager Technology**

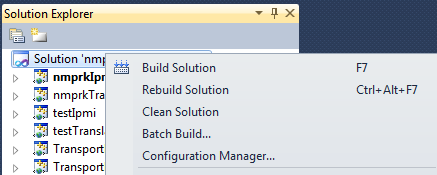
Please follow the instructions below for detailed steps.

Building the library:

1. Start by extracting the compressed nmprk release package to a folder of your choice.
2. Browse to <project source>\nmprk-visualStudio\nmprkIpmi and launch the file nmprkIpmi.sln in visual studio. You should see the following projects listed in the solution explorer:



1. Now start building the whole project. To do this right click on “Solution ‘nmprkIpmi’ and select build solution.



1. As long as the solution builds without errors you can ignore the warnings that are generated during compilation.



1. You can now find all the files you will need to use the library in the following locations:

**nmprkIpmi.lib**: the library to link against for the ipmi engine is located in <project source>\nmprk-visualStudio\nmprkIpmi\Debug

**TransportKCS.dll:** this dll is required by nmprkIpmi.lib for in band (KCS) communication to work(place it in the same directory as nmprkIpmi.lib). It is located in <project source> \nmprk-visualStudio\nmprkIpmi\Debug **(this .dll must be in the same directory as nmprkIpmi.lib for the .lib to work correctly, if you get errors that interface won’t load it’s because these .dlls are not in the same directory as the .lib)**

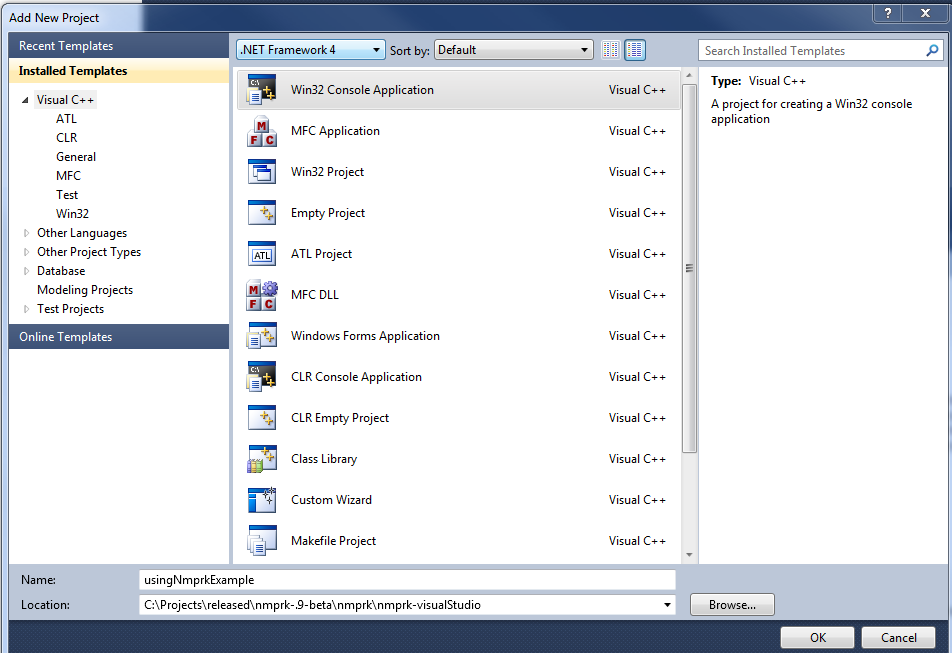
**TransportRMCPP.dll:**  this dll is required by nmprkIpmi.lib for out of band (lan/network) communication to work(place it in the same directory as nmprkIpmi.lib). It is located in <project source> \nmprk-visualStudio\nmprkIpmi\bin\Debug **(this .dll must be in the same directory as nmprkIpmi.lib for the .lib to work correctly, if you get errors that interface won’t load it’s because these .dlls are not in the same directory as the .lib)**

**nmprkTranslation.lib:** the library to link against for the ipmi engine is located in <project source>\nmprk-visualStudio\nmprkIpmi\Debug

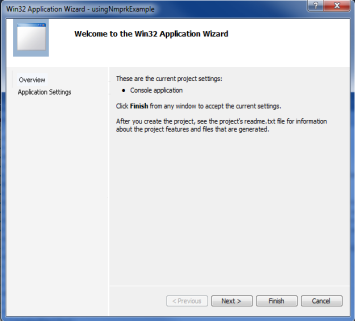
**Creating an application project using the nmprk visual studio solution:**

Here is sample power monitoring and limiting application using Intel® Node Manager Programmer’s Reference Kit APIs. The application will get a device ID, then collect power readings for the device and finally set a 10% power cap based on the current reading. (Note: There are two examples of interfacing to the library come with the NMPRK package. They are the test programs testIpmi.cpp and testTranslation.cpp)

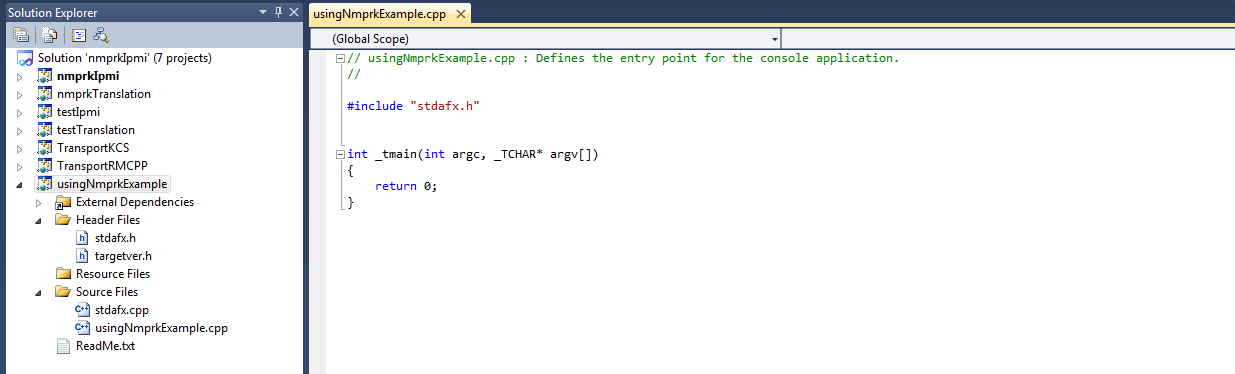
1. The fastest way to start using the nmprk libraries is by adding a new project directly to the nmprkIpmi solution. To start (assuming solution is already open) go to File -> Add -> New Project which will open up setup wizard for the new project.



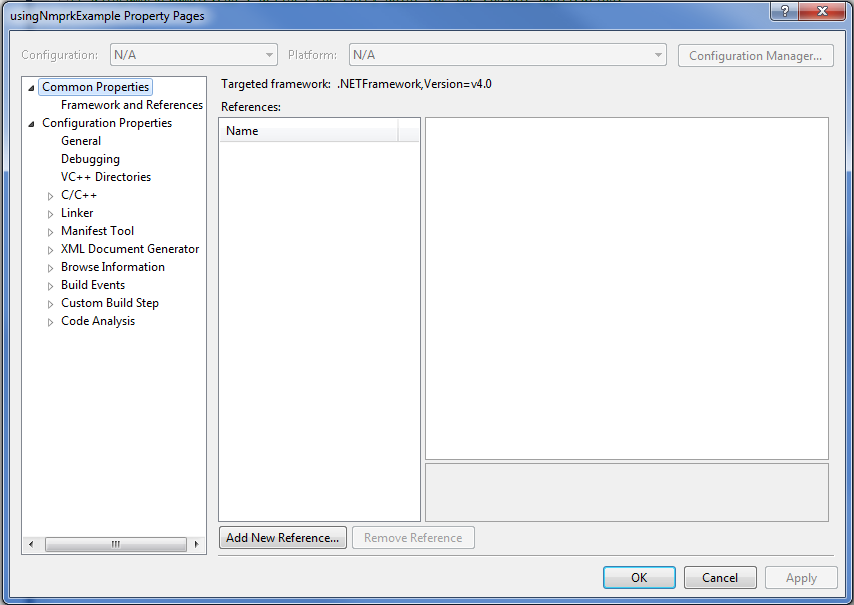
1. For this project we will be making a simple Win32 console application. Go ahead and enter in a name for the new project (we are using “usingNmprkExample”) and you can specify a different default location if desired. With these fields filled in, hit ok to bringing up the next part of the wizard. On the first page hit “Next>”.



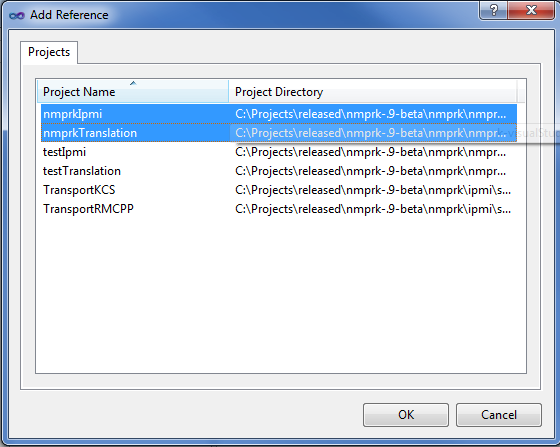
1. On the second page of the wizard we want to uncheck “precompiled header” and then click “finish”. Visual studio should now open up to the main source file for the new project.



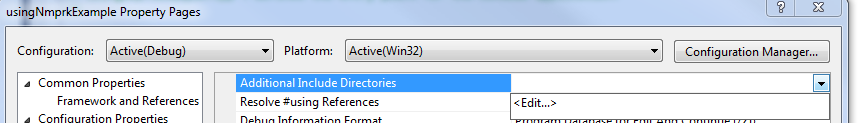
1. First add references to nmprk libraries. For this application we will linking against both libraries, to do this right click on the project in the solution explorer and select “properties”. The following window should open:



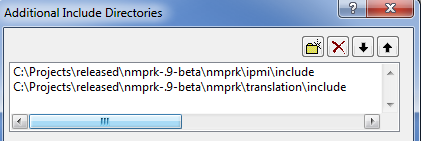
1. We want to go to the Common Properties page. Click the “Add New Reference” button and select nmprkIpmi and nmprkTranslation from the list of projects and hit ok.



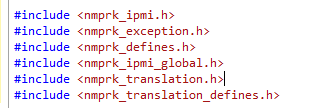
1. Now going back to our new projects source file (in this case usingNmprkExample.cpp) we next need to tell visual studio where to find the header files so we can include them in our project. To do this re-open the “properties” window, this time browsing to Configuration Properties -> C/C++ -> General finding the field called “Additional Include Directories” and clicking in it, and selecting edit after clicking the drop down arrow.



1. In the new windows that pops up click in the first free space in the text box and click the “…” button which will open up a file explorer dialog. We want to add the following folders to the include directories. <project source>\ipmi\include and <project source>\translation\include . When you are done your window should look something like this (make adjustments for where you saved the project source to):



1. Now that we are done with this we can get back to write the actual example application. Going back to our .cpp file first we are going to include in all the header files for the functions we will be using. For this project include the following headers:

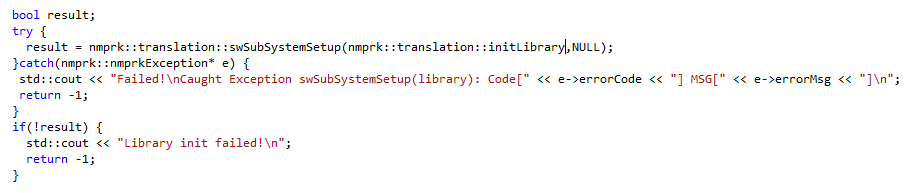


1. First we need to start off by declaring our variable to hold our devices information. Then we need to set what type of device we are using (Node Manager in this case) and what type of connection this is (local/KCS or OOB/lan). We can do this with the following three lines of code.

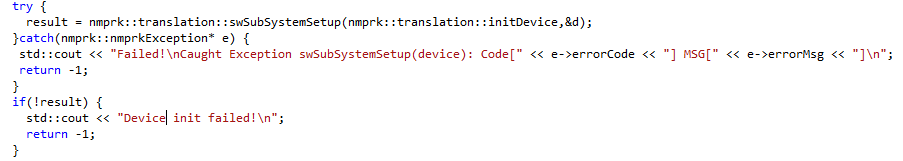


1. We are now ready to start calling our functions. Before we start doing the actual work we first need to init the library and the device. We do this with the following lines:

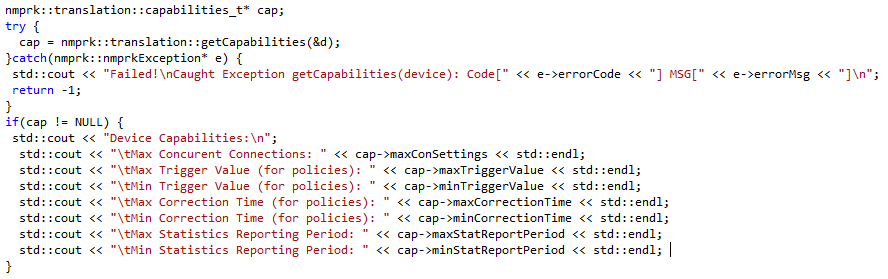
Init the library:



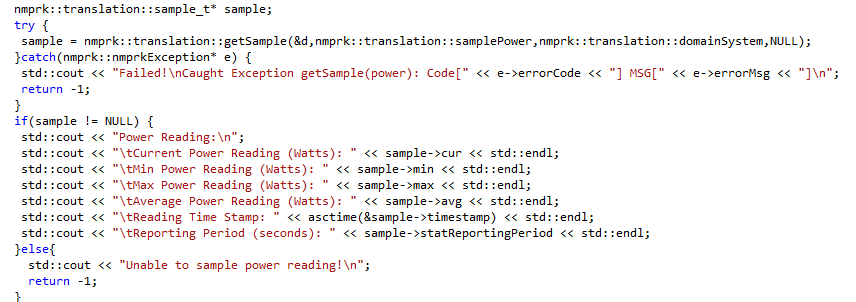
And then the device:



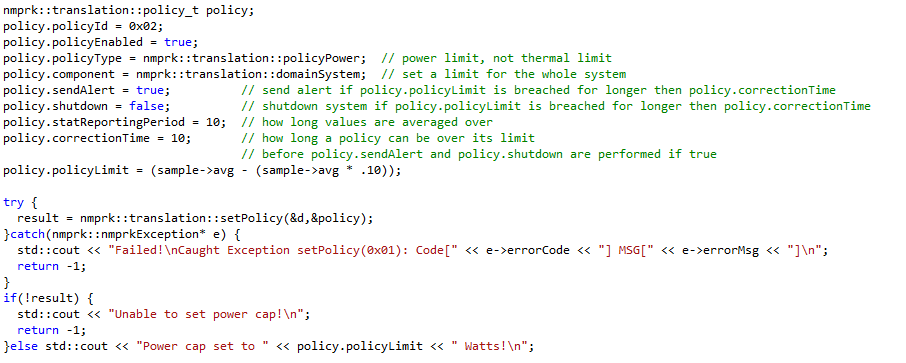
1. With everything initialized we are now ready to start calling our functions. First we will call nmprk::ipmi:: getDeviceId and display the resulting info that is returned. We do this like so:



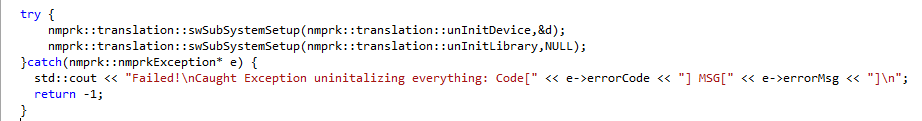
1. Now let’s get our power reading



1. Which just leaves us with setting a 10% power cap:



1. As a final step we just need to uninit the library and device:



**FULL DEMO SOURCE CODE:**

// usingNmprkExample.cpp : Defines the entry point for the console application. For the latest example code, please check

// the testIpmi.cpp and testTranslation.cpp application that are provided as part of the NMPRK

//

#include "stdafx.h"

#include <iostream>

#include <nmprk\_ipmi.h>

#include <nmprk\_exception.h>

#include <nmprk\_defines.h>

#include <nmprk\_ipmi\_global.h>

#include <nmprk\_translation.h>

#include <nmprk\_translation\_defines.h>

int \_tmain(int argc, \_TCHAR\* argv[])

{

nmprk::ipmi::device d;

d.type = nmprk::ipmi::device\_nm;

d.address = "local";

bool result;

try {

result = nmprk::translation::swSubSystemSetup(nmprk::translation::initLibrary,NULL);

}catch(nmprk::nmprkException\* e) {

std::cout << "Failed!\nCaught Exception swSubSystemSetup(library): Code[" << e->errorCode << "] MSG[" << e->errorMsg << "]\n";

return -1;

}

if(!result) {

std::cout << "Library init failed!\n";

return -1;

}else std::cout << "Library initalized!\n";

try {

result = nmprk::translation::swSubSystemSetup(nmprk::translation::initDevice,&d);

}catch(nmprk::nmprkException\* e) {

std::cout << "Failed!\nCaught Exception swSubSystemSetup(device): Code[" << e->errorCode << "] MSG[" << e->errorMsg << "]\n";

return -1;

}

if(!result) {

std::cout << "Device init failed!\n";

return -1;

}else std::cout << "Device initalized!\n";

nmprk::translation::capabilities\_t\* cap;

try {

cap = nmprk::translation::getCapabilities(&d);

}catch(nmprk::nmprkException\* e) {

std::cout << "Failed!\nCaught Exception getCapabilities(device): Code[" << e->errorCode << "] MSG[" << e->errorMsg << "]\n";

return -1;

}

if(cap != NULL) {

std::cout << "Device Capabilities:\n";

std::cout << "\tMax Concurent Connections: " << cap->maxConSettings << std::endl;

std::cout << "\tMax Trigger Value (for policies): " << cap->maxTriggerValue << std::endl;

std::cout << "\tMin Trigger Value (for policies): " << cap->minTriggerValue << std::endl;

std::cout << "\tMax Correction Time (for policies): " << cap->maxCorrectionTime << std::endl;

std::cout << "\tMin Correction Time (for policies): " << cap->minCorrectionTime << std::endl;

std::cout << "\tMax Statistics Reporting Period: " << cap->maxStatReportPeriod << std::endl;

std::cout << "\tMin Statistics Reporting Period: " << cap->minStatReportPeriod << std::endl;

}

nmprk::translation::sample\_t\* sample;

try {

sample = nmprk::translation::getSample(&d,nmprk::translation::samplePower,nmprk::translation::domainSystem,NULL);

}catch(nmprk::nmprkException\* e) {

std::cout << "Failed!\nCaught Exception getSample(power): Code[" << e->errorCode << "] MSG[" << e->errorMsg << "]\n";

return -1;

}

if(sample != NULL) {

std::cout << "Power Reading:\n";

std::cout << "\tCurrent Power Reading (Watts): " << sample->cur << std::endl;

std::cout << "\tMin Power Reading (Watts): " << sample->min << std::endl;

std::cout << "\tMax Power Reading (Watts): " << sample->max << std::endl;

std::cout << "\tAverage Power Reading (Watts): " << sample->avg << std::endl;

std::cout << "\tReading Time Stamp: " << asctime(&sample->timestamp) << std::endl;

std::cout << "\tReporting Period (seconds): " << sample->statReportingPeriod << std::endl;

}else{

std::cout << "Unable to sample power reading!\n";

return -1;

}

nmprk::translation::policy\_t policy;

policy.policyId = 0x02;

policy.policyEnabled = true;

policy.policyType = nmprk::translation::policyPower; // power limit, not thermal limit

policy.component = nmprk::translation::domainSystem; // set a limit for the whole system

policy.sendAlert = true; // send alert if policy.policyLimit is breached for longer then policy.correctionTime

policy.shutdown = false; // shutdown system if policy.policyLimit is breached for longer then policy.correctionTime

policy.statReportingPeriod = 10; // how long values are averaged over

policy.correctionTime = 10; // how long a policy can be over its limit

// before policy.sendAlert and policy.shutdown are performed if true

policy.policyLimit = (sample->avg - (sample->avg \* .10));

try {

result = nmprk::translation::setPolicy(&d,&policy);

}catch(nmprk::nmprkException\* e) {

std::cout << "Failed!\nCaught Exception setPolicy(0x01): Code[" << e->errorCode << "] MSG[" << e->errorMsg << "]\n";

return -1;

}

if(!result) {

std::cout << "Unable to set power cap!\n";

return -1;

}else std::cout << "Power cap set to " << policy.policyLimit << " Watts!\n";

try {

nmprk::translation::swSubSystemSetup(nmprk::translation::unInitDevice,&d);

nmprk::translation::swSubSystemSetup(nmprk::translation::unInitLibrary,NULL);

}catch(nmprk::nmprkException\* e) {

std::cout << "Failed!\nCaught Exception uninitalizing everything: Code[" << e->errorCode << "] MSG[" << e->errorMsg << "]\n";

return -1;

}

return 0;

}