Unidata doesn't provide netcdf fortran prebuild binary for windows, User need to build it under local environment.

Here are the my steps to build netcdf fortran and fortran90 libary in windows with intel compiler and Visual Studio 2010.

1. download the netcdf-C binary library from unidata, `http://www.unidata.ucar.edu//downloads//netcdf//index.jsp`. Uzip the file to a local folder. Add the path of netcdf-C library to the system path. Assume the netcdf-C binary is installed on `netcdf`, here are the netcdf pathes needed to be on system path:

   1. `netcdf\lib`
   2. `netcdf\include`
   3. `netcdf\deps\x64\lib`
   4. `netcdf\deps\x64\include`

2. download the latest stable netcdf fortran source code from unidata. unzip and save to a local hard disk.

3. Say the uzipped netcdf fortran source code folder is `netcdf-fortran-4.4.2`. In the same location create another folder, say `build`, to save the visual studio project. From the `build` start a Windows command prompt. In the command window, type command (assume you have cmake 2.8 installed)

```
% cmake -G "Visual Studio 10 Win64" --build . ..\netcdf-fortran-4.4.2
```

4. Open the `build` folder, Open the resulting Visual Studio solution. In the solution explorer you should see a number of projects. One of them is *ncfortran* that is the project of netcdf Fortran77 API and another one is *netcdff* that is supposed the proejct for netcdf Fortran90 API. In my point of view project *netcdff* is a unsuccessful project with incorrect configuration and can't be build to generate netcdf Fortran90 API. Also project *ncfortran* needs a little bit configuration tuning up to create a Fortan77 API libary. We go through those two projects one by one.

5. This step is dedicated to project *ncfortran* to build netcdf Fortran77 library. Click the project, Go to **Properties->C/C++ ->Preprocessor->Preprocessor Definitions**. Add preprocessor *DLL_NETCDF* at the beginning of the definitions list, also add *INTEL_COMPILER* and *NC_DLL_EXPORT*

at the end of the list. Then save the change and compile the project to generate netcdf Fortran77 library. You will find two result files `ncfortran.lib` and `ncfortran.dll` under `\build\libsrc\Release`.

6. This step is for netcdf Fortran90 library building.

6.1 In the Solution Explorer create a new Intel(R) Visual Fortran Static Library, say **netcdf_f90**, and save it to the build folder.

6.2 From the existing project **netcdff**, copy all the F90 files into the source files of the newly created project **netcdf_f90**.

6.3 Right click project **netcdf_f90**. Go to **Properties->Fortran->Preprocessor**. On the right of the Window,

Set **Preprocess Source File** to **Yes(/fpp)**. Select **Additional Include Directories**, add the path to folder `fortran` and `libsrc`, both of them are folder under the netcdf-fortran source code. In the **Preprocessor Definitions**, add preprocessor *DLL_NETCDF, USE_NETCDF4, NC_DLL_EXPORT, _WINDLL* in order. Then save the change. Go back to the left panel of the **Properties** window, select **Librarian->General**. Add `netcdf.lib` to **Additional Dependencies** and also add the netcd-C lib path to **Additionl Libary Directories**, where `netcdf.lib` is.

6.4 Several netcdf function for file and var parallel access need to be remove to succeed. There is a explanation from

`https://www.unidata.ucar.edu/support/help/MailArchives/netcdf/msg12965.html`. Based on the comments from netcdf developer, building netcdf fortran lib with parallel I/O support in Windows is often not required and complicates later linking with the libraries.

To avoid building parallel I/O support, the definition and usage three functions are removed,

1. nf90_create_par in file netcdf4_visibility.f90 and netcdf4_func.f90,
2. nf90_open_par in file netcdf4_visibility.f90 and netcdf4_func.f90,
3. nf90_var_par_access in file netcdf4_visibility.f90 and netcdf4_func.f90,
4. nf_create_par in file netcdf4_file.f90 and nf_nc4.f90,
5. nf_open_par in file netcdf4_file.f90 and nf_nc4.f90,
6. nf_var_par_access in file netcdf4_file.f90 and nf_nc4.f90.

This changes involves four files,

1. netcdf4_file.f90

2. netcdf_func.f90

3. netcdf4_visibility.f90

4. nf_nc4.f90.

In file `netcdf4\_visibility.f90`, remove those parallel function from the public interface. In file `netcdf4\_func.f90`, comment out all the blocks defining those three functions. In file `netcdf4\_file.f90`, there two usages of *nf_open_par* and *nf_create_par*, and replace them with their serial version as *nf_open* and *nf_create* respectively. In file `nf\_nc4.f90`, comment out the definitions of *nf_create_par*,*nf_open_par* and *nf_var_par_access*.

After finished those changes, netcdf f90 library can be build. Visual studio may throw out a number of waring message in the building.

6.5 Go back to the project **netcdff**, remove all the F90 files (not delete). Then rebuilt it, this project will creat a netcdff.dll file, which is also needed to use netcdf F90 api.

7. Select a testing project for F90 api to testing the functionality of the newly build library. To simplify the include and linking paths configurations for the testing project, it is better to copy all the resulting mod and lib files generated by project **netcdf_f90** and

dll file from project **netcdff** into a folder. In the project configuration, add this folder as the additional include and library searching path. Aslo library files **netcdf_f90.lib** and **netcdff.dll** needs to be in the linker additional dependency list. I found

to avoid some conflicting definition with MS visual studio library, **LIBCMT.lib** and **libifcoremt.lib** needed to be ignored in the linking. This can be configured as input to **Ignore Specific Library** in the Linker configuration.