Getting Started : Signing your NXOpen executable

Once you have fully tested your NXOpen API application, you should "sign" it before distributing it to your end users. Even if your end users also have access to the Author license which is required to load and run an NXOpen application, signing it has its benefits.

- Your application will load faster because NX will immediately recognize that it has been signed.
- The syslog (Help-> Log File) will not be cluttered with the output from the multiple checks that NX will do to determine whether it can load and run the program.

1) C/C++, C# or VB only (Java: Proceed directly to step 3)

   Add the NXSigningResource to your Visual Studio project

   In the Solution Explorer, select the Resource Files folder and
   <RMB>Add-> Existing Item

   C/C++:
   - Browse and select %UGII_BASE_DIR%\UGOPEN\NXSigningResource.cpp

   C# or VB .NET:
   - Browse and select %UGII_BASE_DIR%\UGOPEN\NXSigningResource.res
   - Select the newly added NXSigningResource.res and in set
     Properties-> Build Action = Embedded Resource

2) Make sure the build configuration is set to Release then
   Build-> Rebuild Solution

3) Sign the executable from an NX Command Prompt window:
   You probably already have one open because you *started* the Visual Studio or Eclipse from it. Otherwise:
   Start-> All Programs-> NX#-> NX Tools-> Command Prompt

   Java:
   ```
   signJar <path to your>.jar
   ```

   C/C++:
   - NX8.5 and later:
     ```
     signCpp <path to your>.dll or .exe
     ```
   - NX8.0 and earlier:
     ```
     nxSign <path to your>.dll or .exe
     ```

   C# or VB .NET:
   - NX8.5 and later:
     ```
     signDotNet <path to your>.dll or .exe
     ```
   - NX8.0 and earlier:
     ```
     signLibrary <path to your>.dll or .exe
     ```

Note: You do not need to use the path to the signing utilities because the PATH of the NX Command Prompt shell already includes both the UGOpen and UGII folders.