Migrating to Borland C++BuilderX from Microsoft Visual C++ 6.0 for Symbian OS development - While keeping both environments working

John Pagonis, April 2004

Borland have recently released the powerful C++BuilderX Mobile Edition (both as stand alone and as part of the Mobile Studio distribution) that works with a plethora of compilers and tool-chains including these of Metrowerks, Borland and Microsoft.

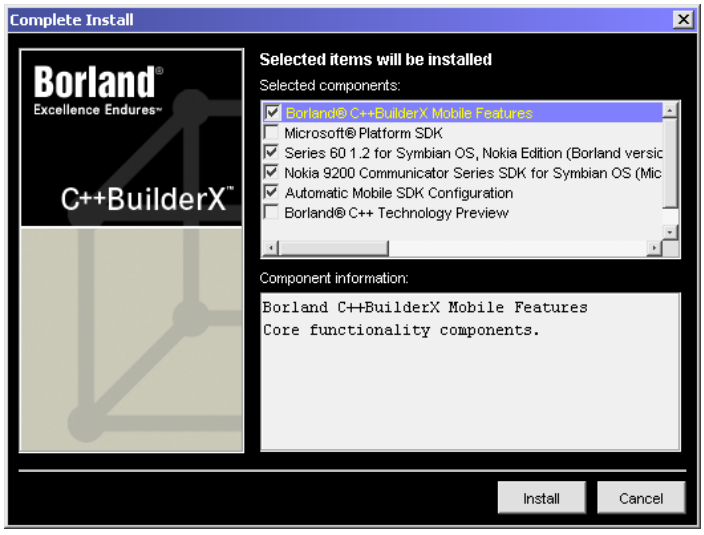
Developers that are still using Microsoft Visual C++ 6.0 (referred in this text as MSVC6) for Win32 hosted development and debugging of Symbian OS software, who nevertheless want to develop for Symbian OS using Borland's IDE can do so at no risk, while keeping MSVC6 intact on their development machines.

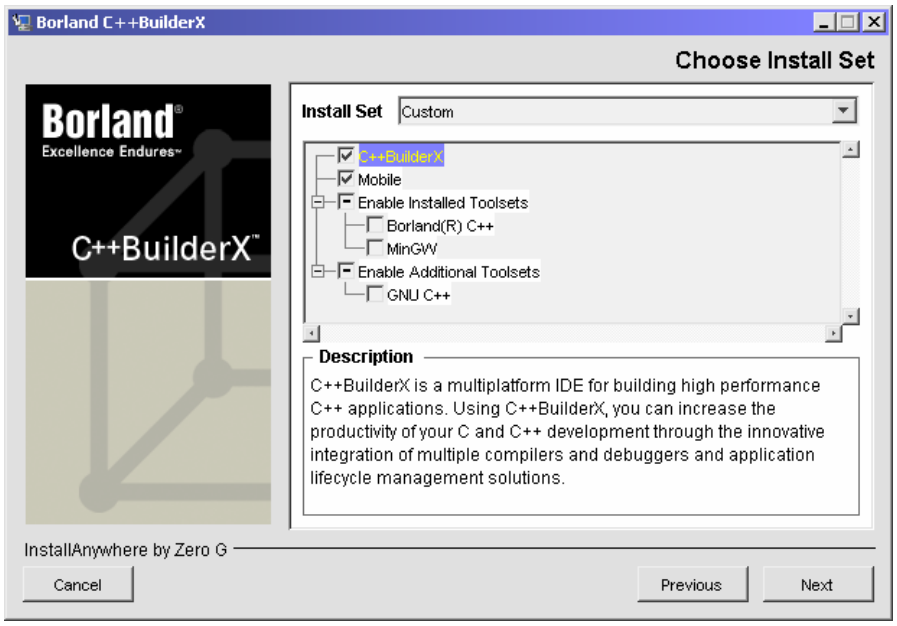
Effectively the Borland C++BuilderX IDE, can seamlessly work with all existing tool-chains usable with Symbian OS SDKs. In particular it can work with both Microsoft visual C++ 6.0 compiled binaries (a.k.a. WINS binaries) as well as Metrowerks CodeWarrior binaries (a.k.a. WINSCW binaries).

To use the already installed tool-chain provided by MSVC6 which has been used for WINS builds(i.e. MSVC6 compiled binaries for emulated Symbian OS on Win32) and is already installed on their workstations, developers should follow the next steps after purchasing C++BuilderX from Borland or downloading their trial demo.

5 Easy Steps to reaching Borland

1. On installation, the user is presented with the option to install the Borland compiler, the Borland tool-chain, the GCC and the .NET framework and compilers (not an option in the demo download).





Developers should know that, in general, if one installs the Microsoft .NET tool-chain, then MSVC6 will stop working. Subsequently this option needs to be de-selected in the installation pane.

2. After completing the fundamental IDE installation, the SDK selection allows installing any SDKs shipped with the tool, or that have been downloaded from the Web.

3. When installation has been completed and the license has been installed, one can not go and immediately use the SDKs as the Borland IDE doesn't yet know about the MSVC6 tool-chain.

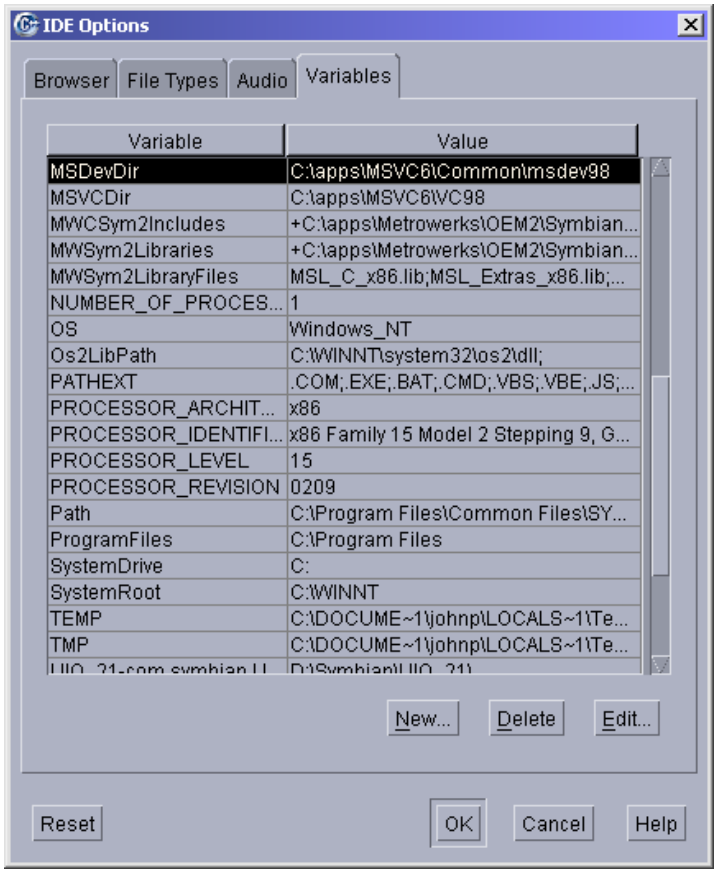
To get the MSVC6 tool-chain working from within C++BuilderX IDE, the paths to the MSVC6 components have to be made known to the IDE. This peculiarity is due to different setups of environment and path variables under different configurations of Windows NT/W2K/XP. Since the paths may have been set correctly, C++BuilderX may pick them up. In case they haven't, we can add them easily.

This is achieved by locating the VCVARS32.BAT in the VC98\Bin directory of the root installation directory of MSVC6 (e.g., c:\apps\MSVC6\VC98\Bin) and adding the following variable declarations (found in the VCVARS32.BAT file) and their values:

VSCommonDir
MSDevDir
MSVCDir
LIB

as well as appending to the Path variable the PATH (as found in VCVARS32.BAT) into the Borland IDE under the

Tools->IDE Options->Variables pane:



These values are entered in the pane by either pressing ‘New’ or by editing any existing variables such as ‘PATH’ by pressing ‘Edit’. Obviously for different installations these paths and values may vary according to the location of the installed MSVC6 components.

4. At this point, one can compile and build from within the C++BuilderX IDE using the MSVC6 tool-chain. Unfortunately, to debug one needs a bit more effort. This is because unlike MSVC6, C++BuilderX does not embed a debugger in the IDE executable; which is a good thing since it has been designed to work and drive many debuggers.

Consequently, C++BuilderX IDE needs to communicate to a debugger that understands the MS compiler x86 binaries, before it can go into a WINS debugging session.

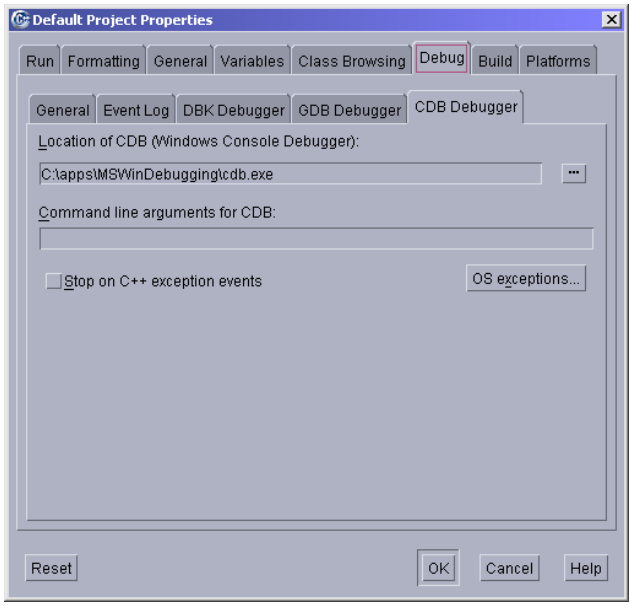
Fortunately a command line debugger, named CDB, is available for download from Microsoft, as part of the Microsoft Debugging Tools package, at

http://www.microsoft.com/whdc/ddk/debugging/installx86.mspx

5. After downloading and installing the Microsoft debugging tools, the only thing that is left to do, is to tell the C++BuilderX IDE where CDB is located. Since the Borland C++BuilderX IDE knows how to talk to CDB, there is nothing more to do.
Therefore in the

Project->Default Project Properties->Debug->CDB pane

one must enter the location of cdb.exe (e.g., C:\apps\MSWinDebugging\cdb.exe) and nothing else (since standard C++ exceptions are not supported by Symbian OS and no command line arguments are needed).

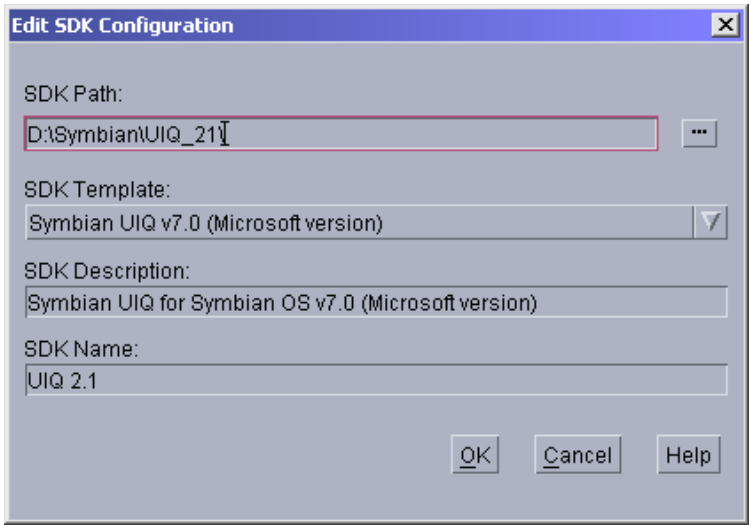


Suffice to say, if one wants to install C++BuilderX for Symbian OS development and doesn't have any Microsoft compilers already installed, Borland's installer can simply install the .NET compilers and allow WINS development without going through step 3 while steps 4 and 5 are always necessary.

Getting to know the SDKs

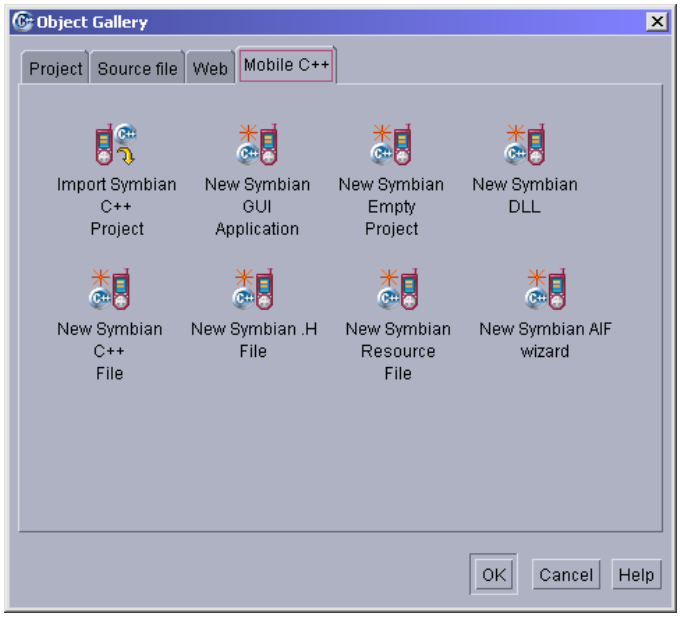
In order for C++BuilderX to make use of any installed Symbian SDKs, one has to simply point the IDE to the root path of that SDK installation, while specifying it's template type and compiler requirements, in the :

Tools->Symbian SDK Configuration->Add/Edit pane

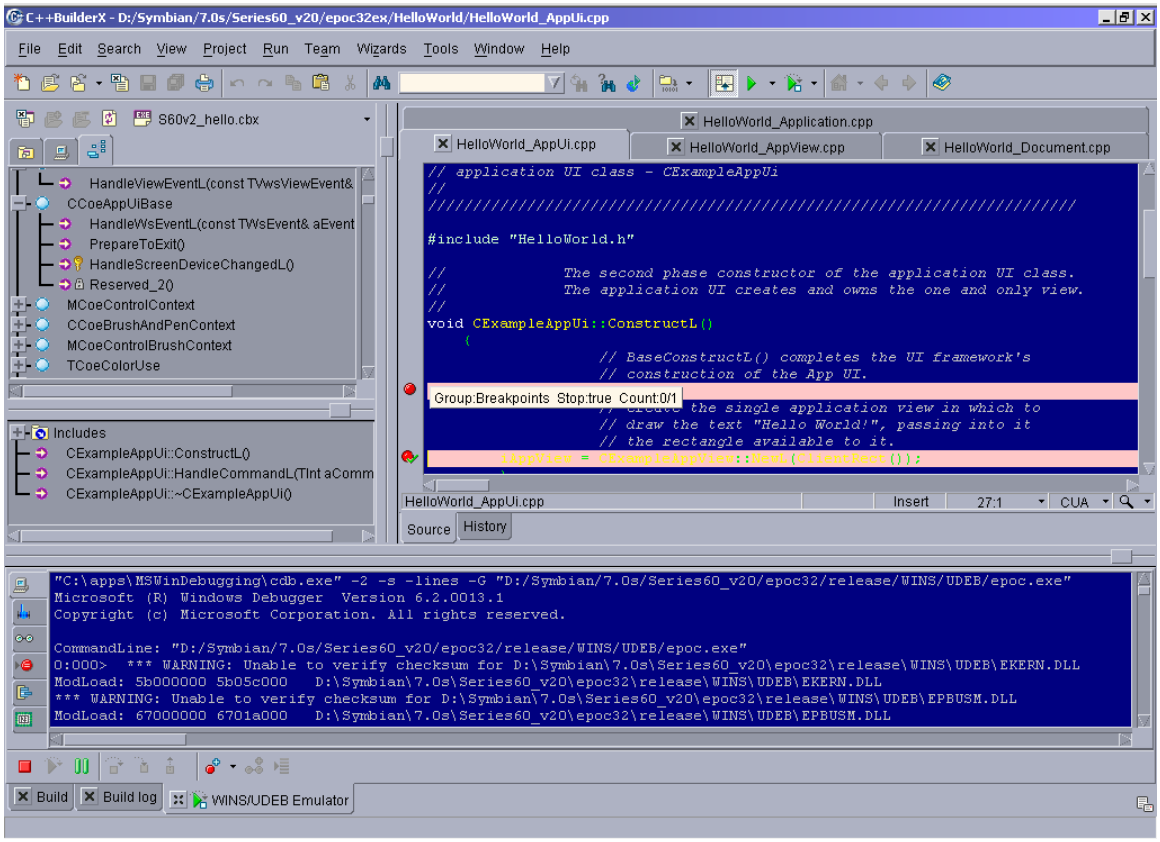


To make use of any SDK one only has to create anew project, or import one from a bld.inf file, which will always be kept in sync from Borland's project manager !
This is achieved from the:

File->New->Mobile C++ pane



Now, all should be setup and ready to go while keeping both tools on the system without impacting any of them !



Relevant links:

http://www.borland.com/mobile/
http://www.borland.com/mobile/cbuilderx/index.html
http://www.microsoft.com/whdc/ddk/debugging/installx86.mspx
http://www.borland.com/products/downloads/download_mobile.html

Want to be kept informed of new articles being made available on the Symbian Developer Network?

Subscribe to the Symbian Community Newsletter.
The Symbian Community Newsletter brings you, every month, the latest news and resources for Symbian OS.