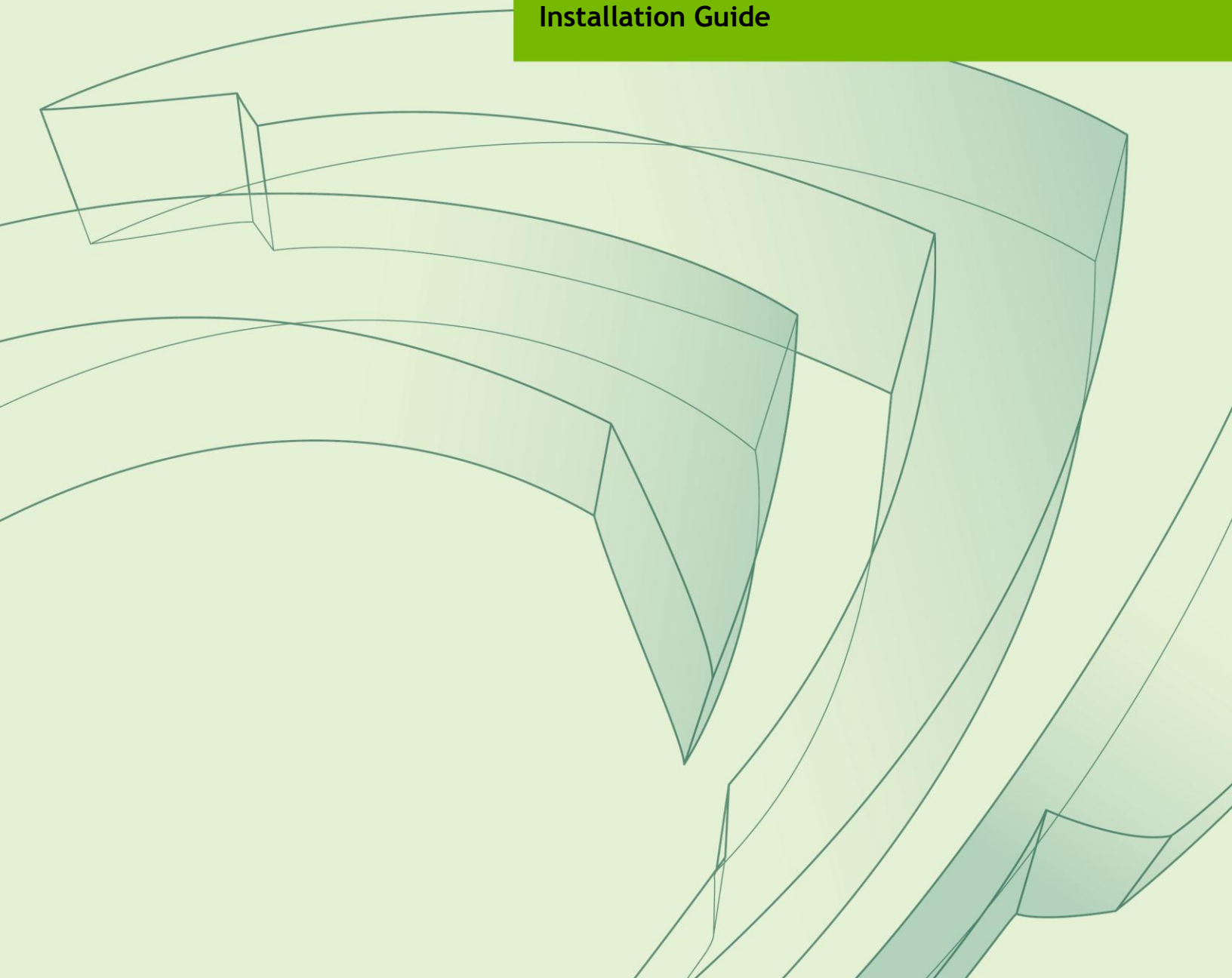




NVIDIA DEBUG MANAGER FOR ANDROID NDK - VERSION 12.0.0

DA-05628-001_v04 | March 2011

Installation Guide



DOCUMENT CHANGE HISTORY

DA-05628-001_v04

Version	Date	Authors	Description of Change
01			Initial release
02	01/19/2011	Sebastien	Updated ADT for 8.0.1
03	02/15/2011	Ryan	Updated ADT for 9.0.0
04	02/17/2011	Ryan	Updated Getting Started...
05	03/14/2011	Ryan	Updated ADT for 10.0.0
06	04/22/2011	Ryan	Updated ADT for 10.0.1
07	06/09/2011	Ryan	Updated ADT for 11.0.0
08	07/18/2011	Jeff	Updated ADT for 12.0.0

TABLE OF CONTENTS

Software Requirements	1
Software Components.....	1
Pre-Installation Requirements	1
Installation	3
Installing Eclipse SDK	3
Installing C/C++ Development Tools.....	3
Installing ADT	5
Installing NVIDIA Debug Manager for Android NDK Eclipse Plugin.....	7
Installing Cygwin.....	9
Getting Started... ..	14
Debugging Tegra Pack Samples.....	14
Debugging Android NDK Samples.....	14
Debugging your own project built outside of Eclipse	14
Configuring gdbserver Settings.....	16
Using an APK Bundled gdbserver	17
Troubleshooting	19
Limitations	19
Reporting Issues	19

SOFTWARE REQUIREMENTS

SOFTWARE COMPONENTS

The NVIDIA Debug Manager for Android NDK Eclipse plugin expects very specific components to be installed – in order to operate as expected. Please make sure that your Android application development environment is conformant with the following components versions:

- ▶ Android SDK r12 (r11 r10 r09 r08)
- ▶ Android NDK r6
- ▶ Eclipse Classic 3.6.x (Helios)
- ▶ Eclipse C/C++ Development Tools 7.0.x
- ▶ Android Development Tools (ADT) 12.0.x

PRE-INSTALLATION REQUIREMENTS

Before you install the NVIDIA Debug Manager for Android NDK Eclipse plugin, please make sure that the following components and environment variables are properly installed and configured:

- ▶ Install the Android SDK r12 (r11 r10 r09 r08) and the Android NDK r6. See <http://developer.android.com/sdk/installing.html> and <http://developer.android.com/sdk/ndk/index.html> for installation instructions.
- ▶ Download SDK and NDK from http://dl.google.com/android/android-sdk_r12-windows.zip and <http://dl.google.com/android/ndk/android-ndk-r6-windows.zip>
- ▶ The NDKROOT environment variable should be set to the location of the Android NDK.

- You should have a Tegra 2 board connected and available to adb. This can be verified by running two commands:

```
adb devices - this command should output at least one available device.  
adb shell ps - this command should output the list of processes running on the  
Tegra 2 board.
```

To run adb your PATH environment variable should contain the tools subdirectory from the Android SDK.

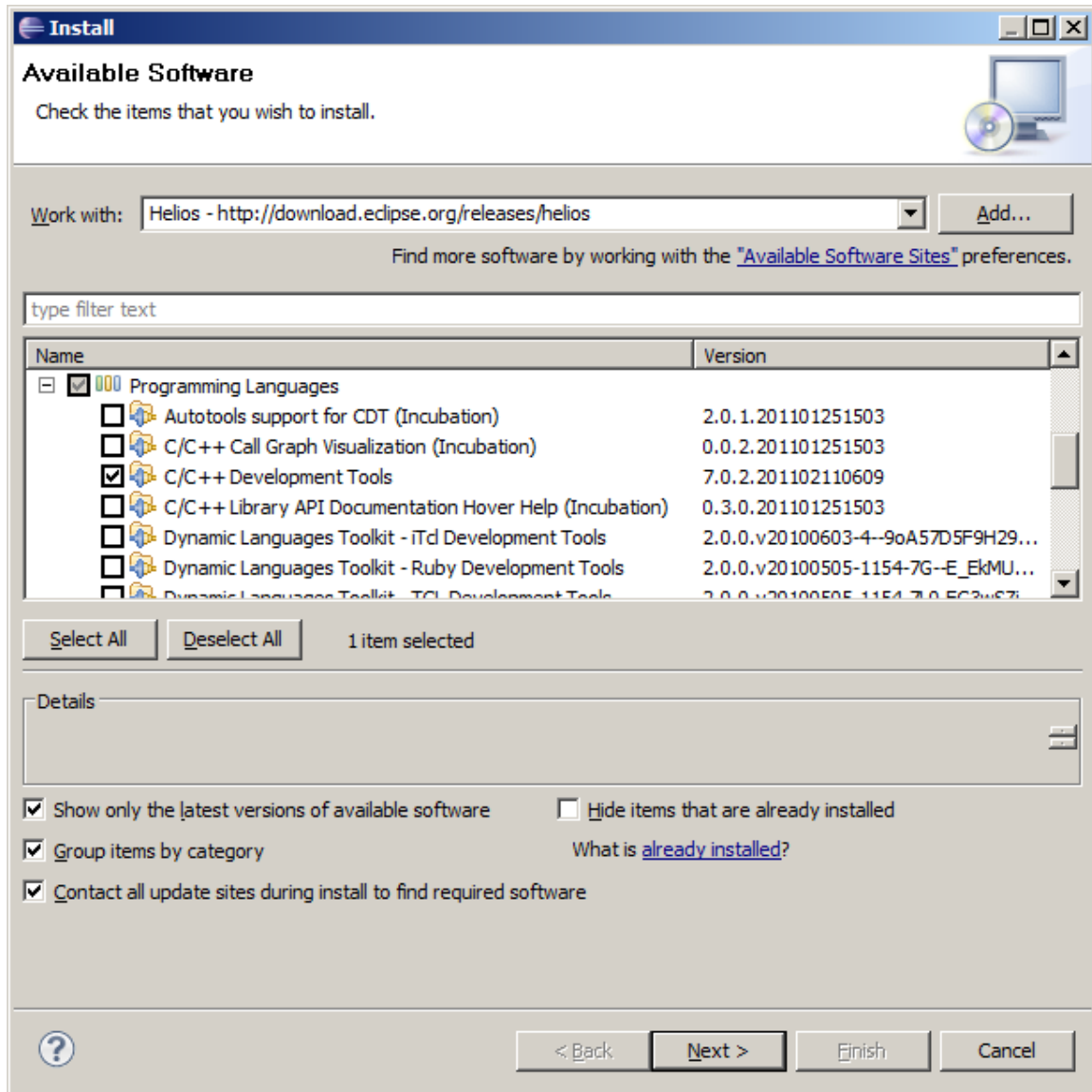
INSTALLATION

INSTALLING ECLIPSE SDK

1. Download Eclipse Classic 3.6.x (Helios) from <http://eclipse.org/downloads/>
2. Unpack the downloaded archive by 7zip or any other compress tool.
3. Start Eclipse, for instance by double-clicking eclipse.exe in the “eclipse” directory which is unpacked above.
4. Select a workspace location.
5. Verify the Eclipse version number by opening Help / About Eclipse SDK. Close the dialog.

INSTALLING C/C++ DEVELOPMENT TOOLS

6. Open Help / Install New Software. This shows the “Install” dialog.
7. In the “Work with” dropdown list choose “Helios – <http://download.eclipse.org/releases/helios>” and wait while the list of available software is loaded.
8. From the list of available software select “Programming Languages” / “C/C++ Development Tools 7.0.x”.



9. Press the "Next >" button. Then press the "Next >" button again and then accept the license agreement and press the "Finish" button.
10. Wait while the software is installed and then agree to restart Eclipse by pressing "Restart Now".
11. Verify that C/C++ Development Tools 7.0.x is installed by opening Help / About Eclipse SDK / Installation Details. You should see C/C++ Development Tools 7.0.x in the list of installed software. Close the "Eclipse SDK Installation Details" dialog and then the "About Eclipse SDK" dialog.

INSTALLING ADT

ADT needs be installed according to <http://developer.android.com/sdk/eclipse-adt.html>

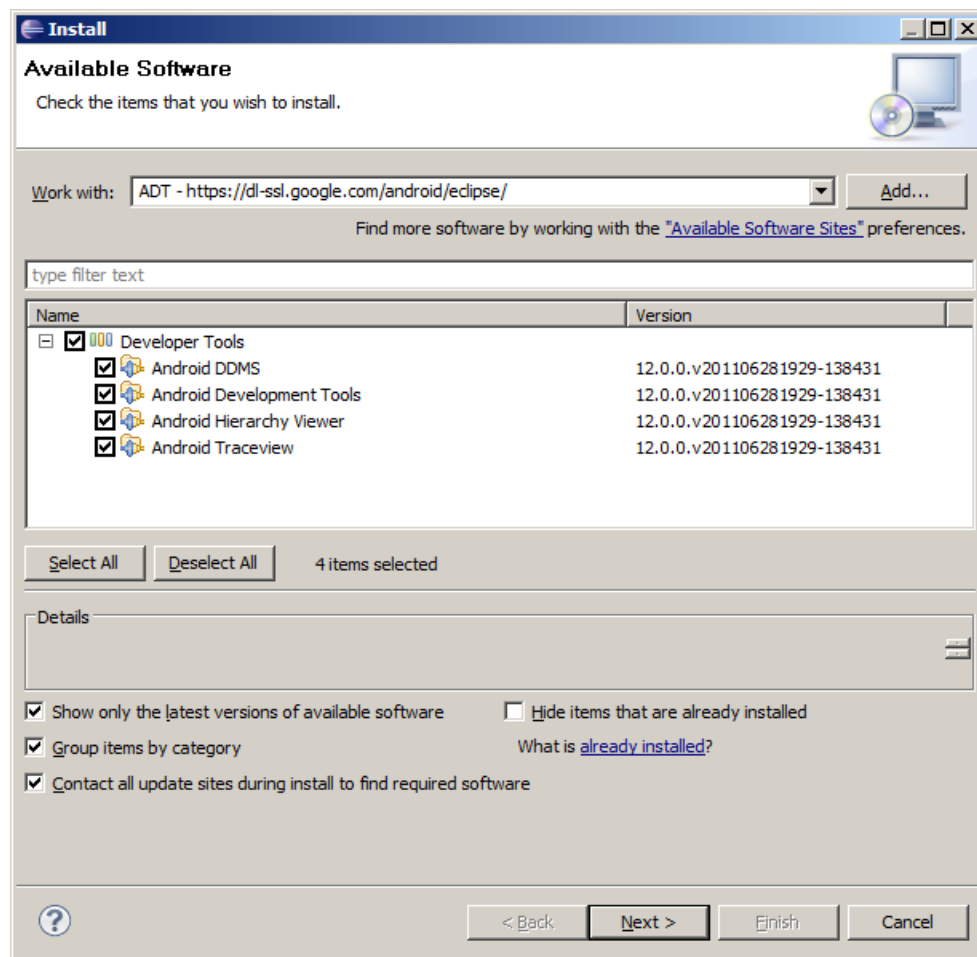
12. Open Help / Install New Software and press the “Add” button.
13. Type in “ADT” into the “Name” field and “https://dl-ssl.google.com/android/eclipse/” into “Location” and click “Ok”.

Note: if no versions are available on this site, you may download an archived .zip at:

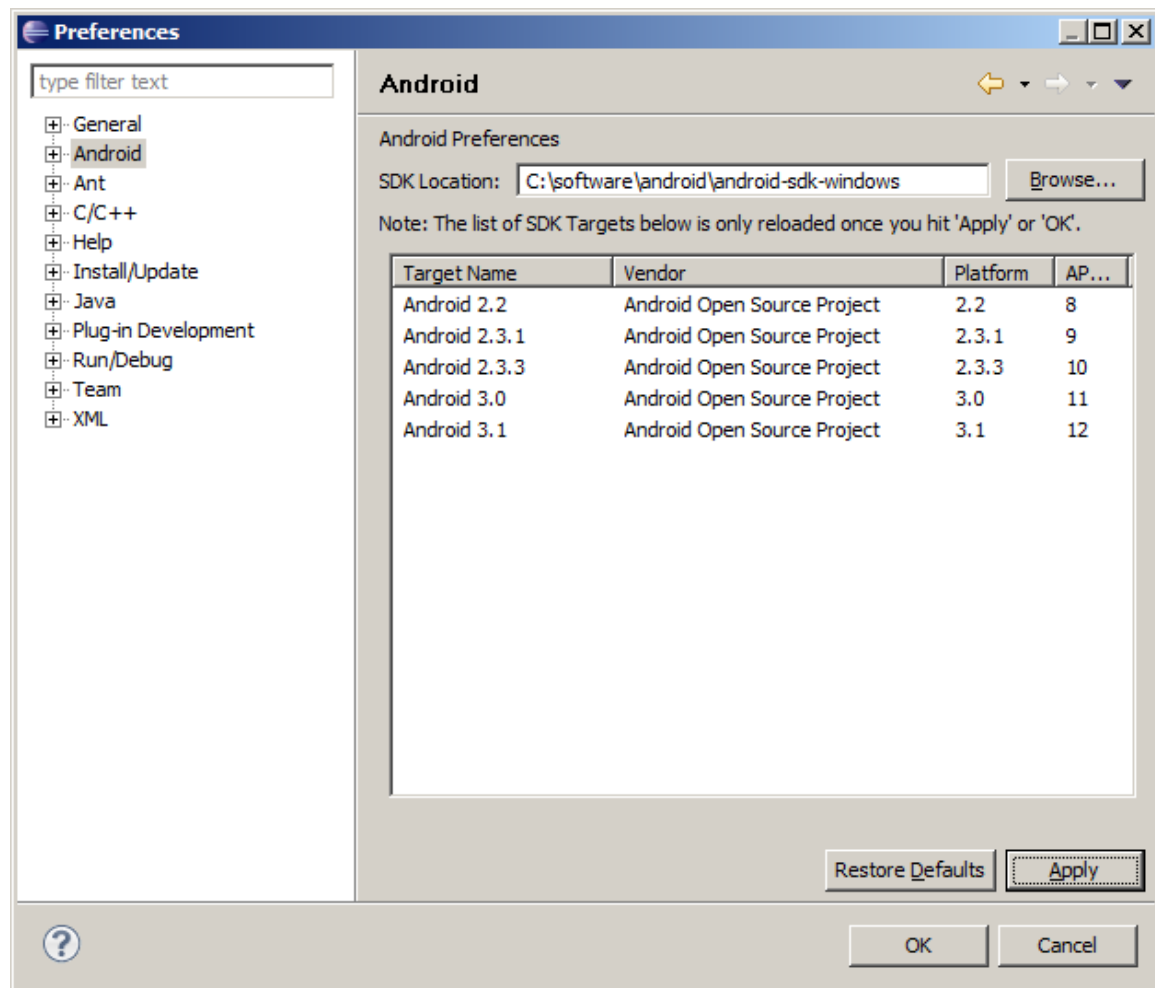
<http://dl.google.com/android/ADT-12.0.0.zip>

Click the “Archive...” button and enter the proper path to ADT-x.0.0.zip in “Add Repository”.

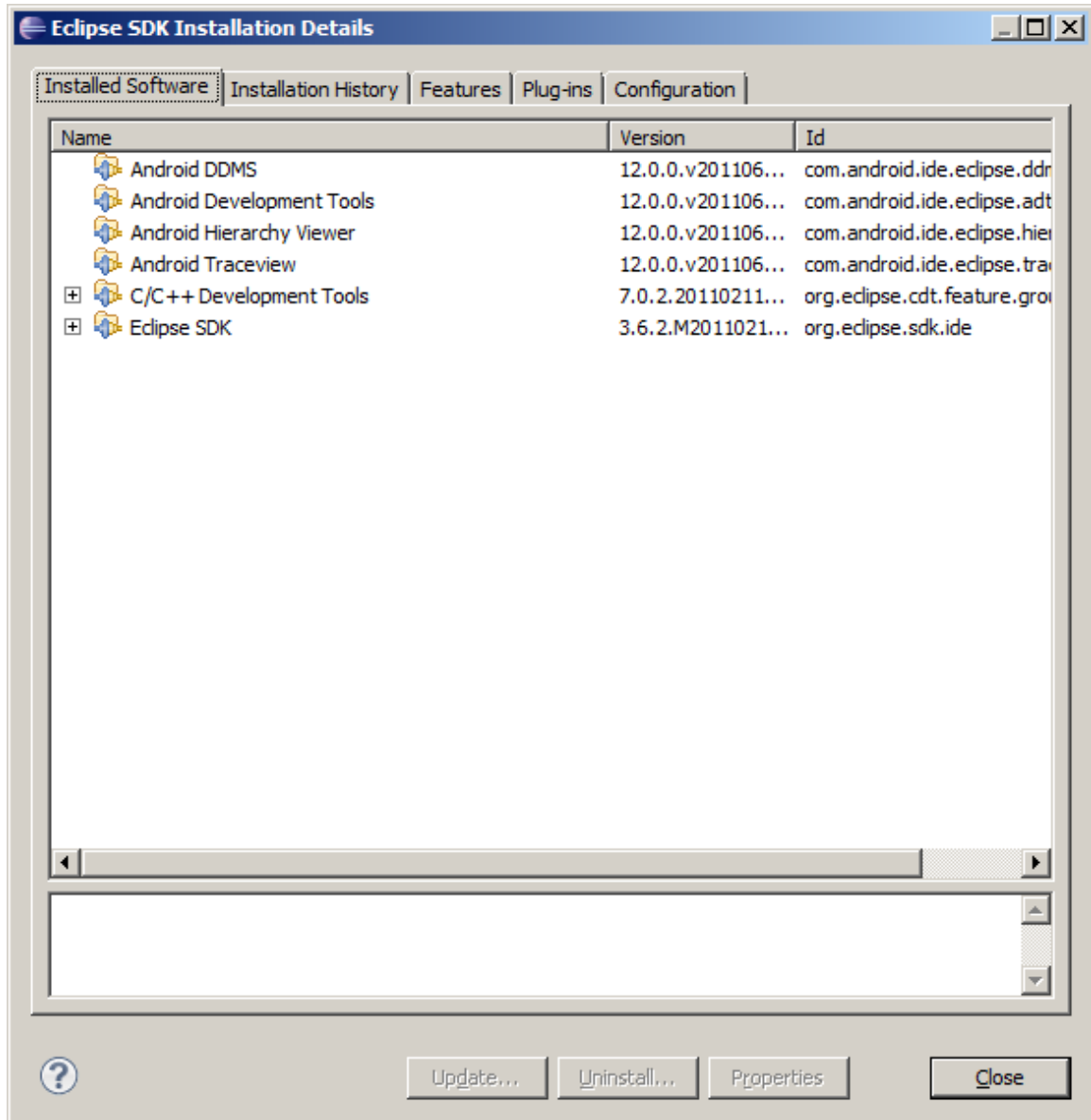
14. Select “Developer Tools”, and you can see all the available software s are selected in the software list if you open the folder.



15. Press Next, then Next again, then read and accept End User License Agreement and press the “Finish” button.
16. Press the “Ok” button on the security warning claiming that you are installing unsigned content. Wait for installation to complete, then agree to restart Eclipse by pressing “Restart Now”.
17. After Eclipse restarts open “Window” / “Preferences”, select “Android” on the left. Enter the correct Android SDK location – typically “c:\android\android-sdk-windows\” (based on your Android SDK installation path) - and press “Ok”.



18. Verify that ADT x.0.0 is installed by opening Help / About Eclipse SDK / Installation Details. You should see “Android DDMS x.0.0” , “Android Development Tools x.0.0” and “Android Hierarchy Viewer x.0.0” in the list of installed software. Close the “Eclipse SDK Installation Details” dialog and then the “About Eclipse SDK” dialog.



INSTALLING NVIDIA DEBUG MANAGER FOR ANDROID NDK ECLIPSE PLUGIN

19. Open Help / Install New Software and press the “Add” button.
20. Press “Archive” and enter the location of the supplied update site archive (zip file).

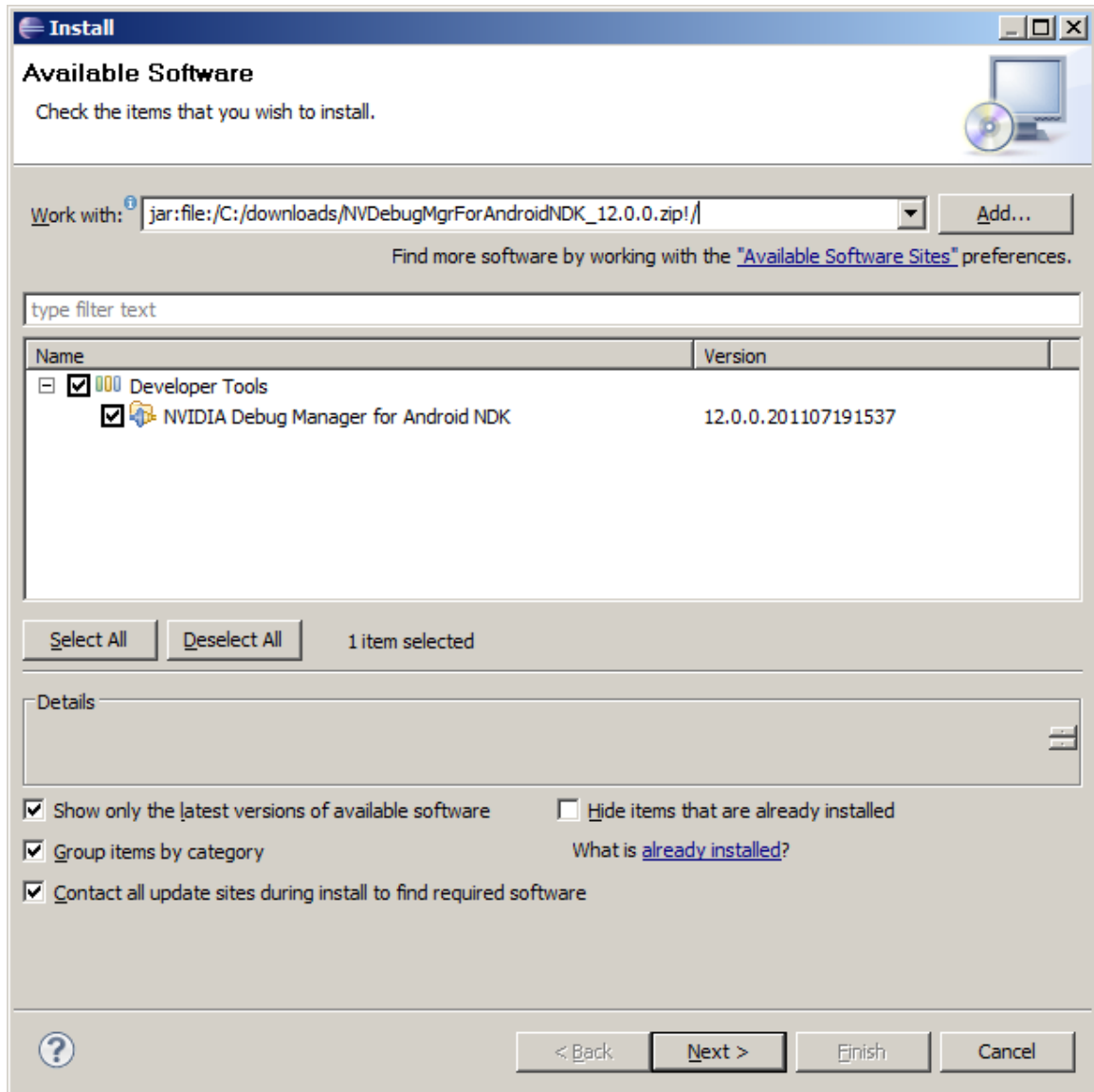
NVDebugMgrForAndroidNDK_12.0.0.zip

Available from:

<http://developer.nvidia.com/tegra/downloads#Tools>

Then press “Ok”.

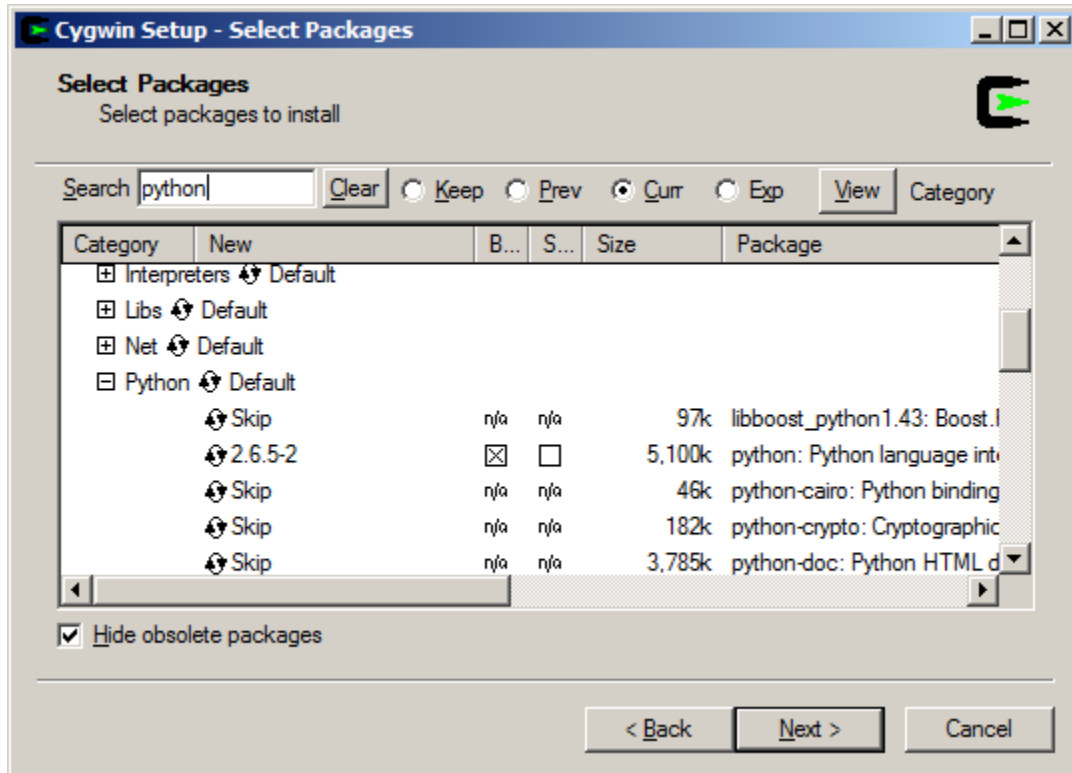
21. Open the “Developer Tools” folder in the available software list and select “NVIDIA Debug Manager for Android NDK”.

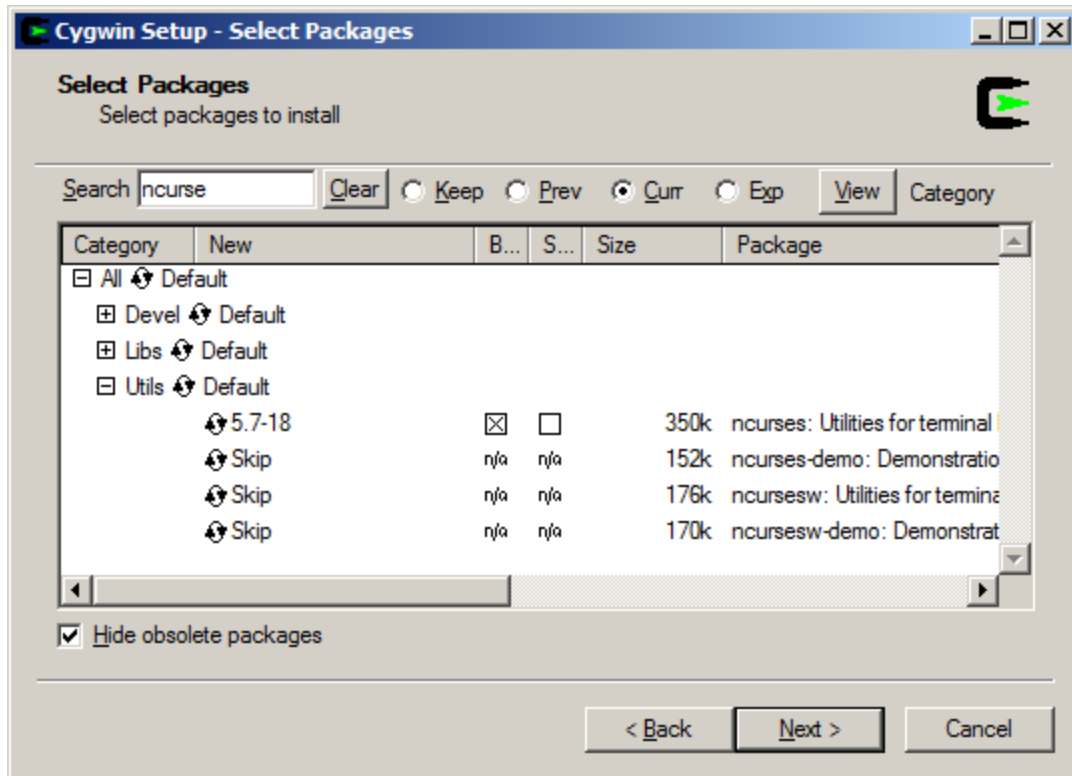


22. Press Next, then Next again, then read and accept the End User License Agreement and press “Finish”.
23. Press “Ok” button on the security warning claiming that you are installing unsigned content. Wait for installation to complete, then agree to restart Eclipse by pressing “Restart Now”.

INSTALLING CYGWIN

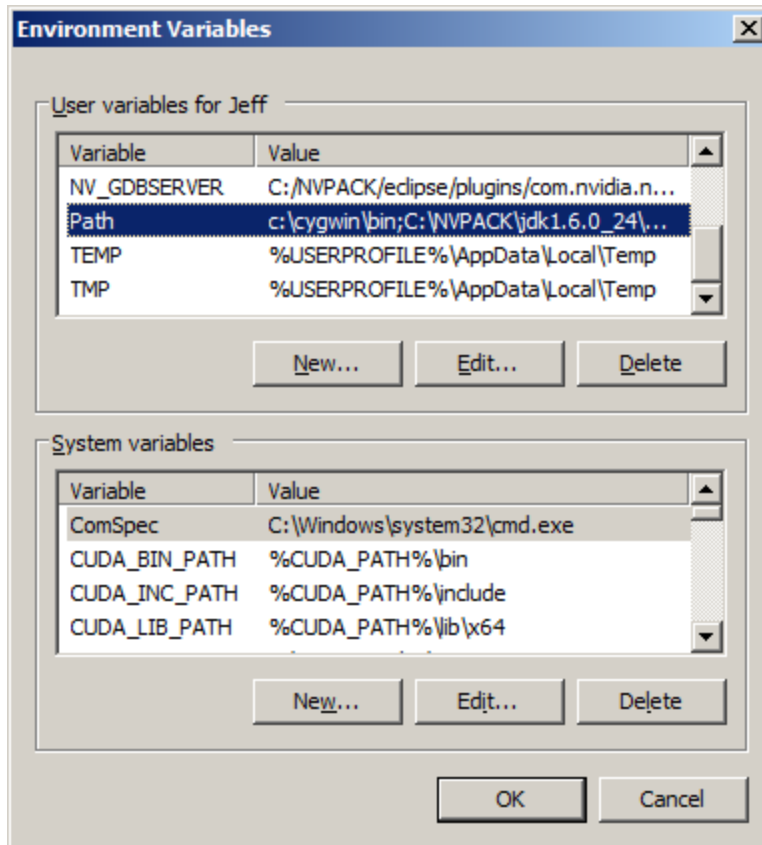
24. Install Cygwin and make sure Python 2.6 and Ncurses are selected to be installed.



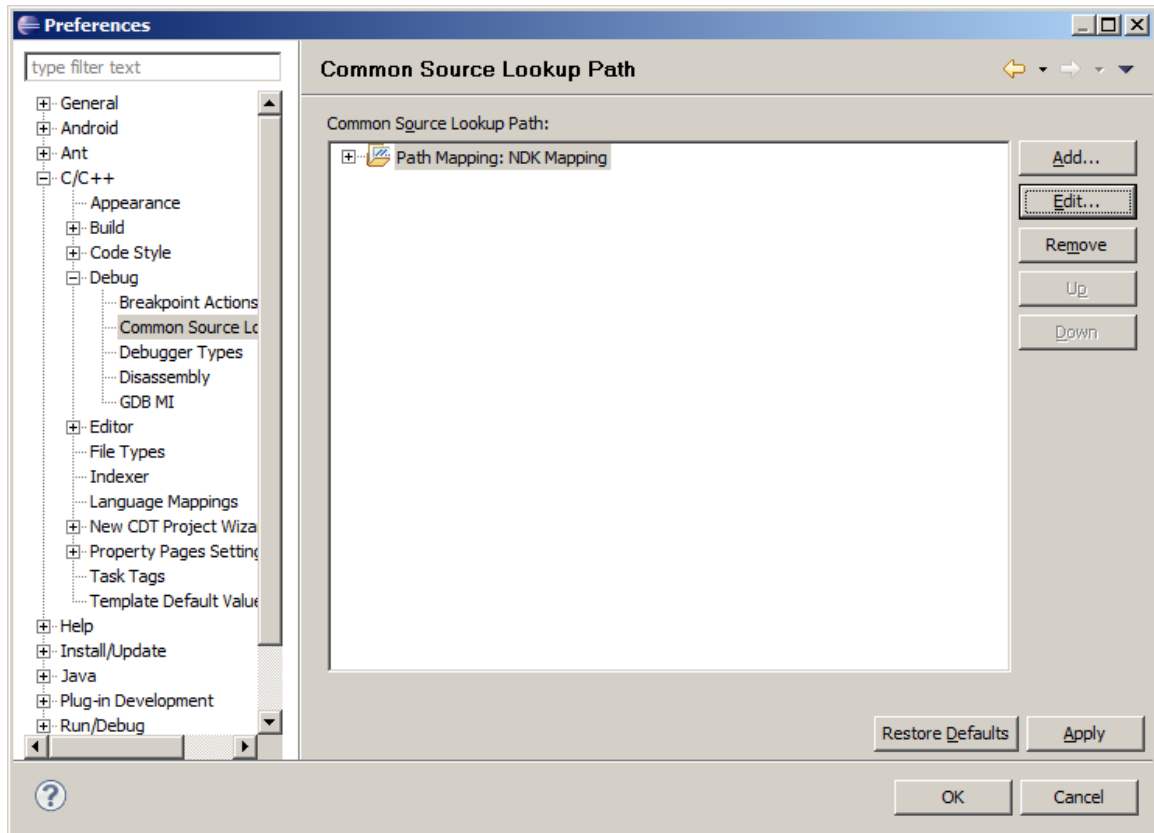


25. Add `$CYGWIN_HOME/bin` to your PATH environment variable.

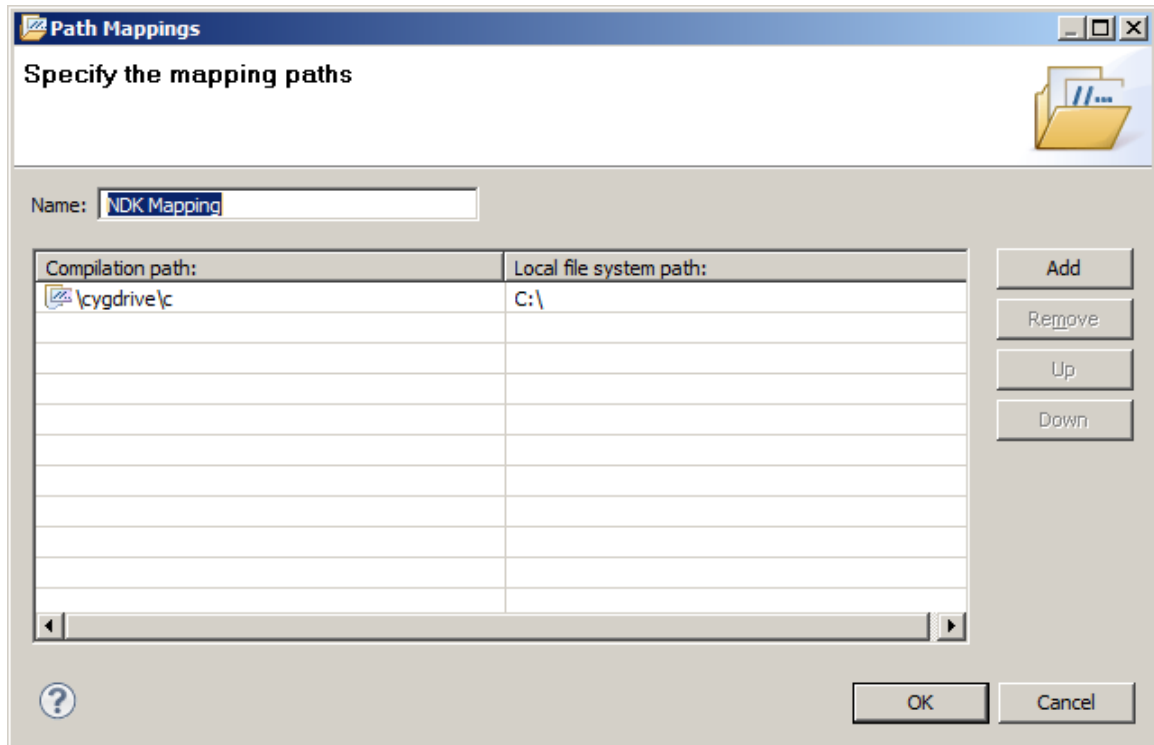
Note: If you are running on Windows, use `%CYGWIN_HOME%` instead.



26. In Eclipse, select the menu item "Window->Preferences", select "C/C++ -> Debug -> Common Source Lookup Path", click "Add" and select "Path Mapping".



Under “Compilation path”, type “\cygdrive\c” and under “Local file system path”, select “C:”. This ensures that gdb and Eclipse are on the same page with regards to source code. If your source code resides on other drives than C:, please add those drives in a similar manner.



The installation now is complete.

GETTING STARTED...

DEBUGGING TEGRA PACK SAMPLES

To debug Tegra sample projects, simply import the project into eclipse, build the project, and then right click the project -> Debug as -> Android NDK application, the debug configuration will be properly created and you can start debugging.

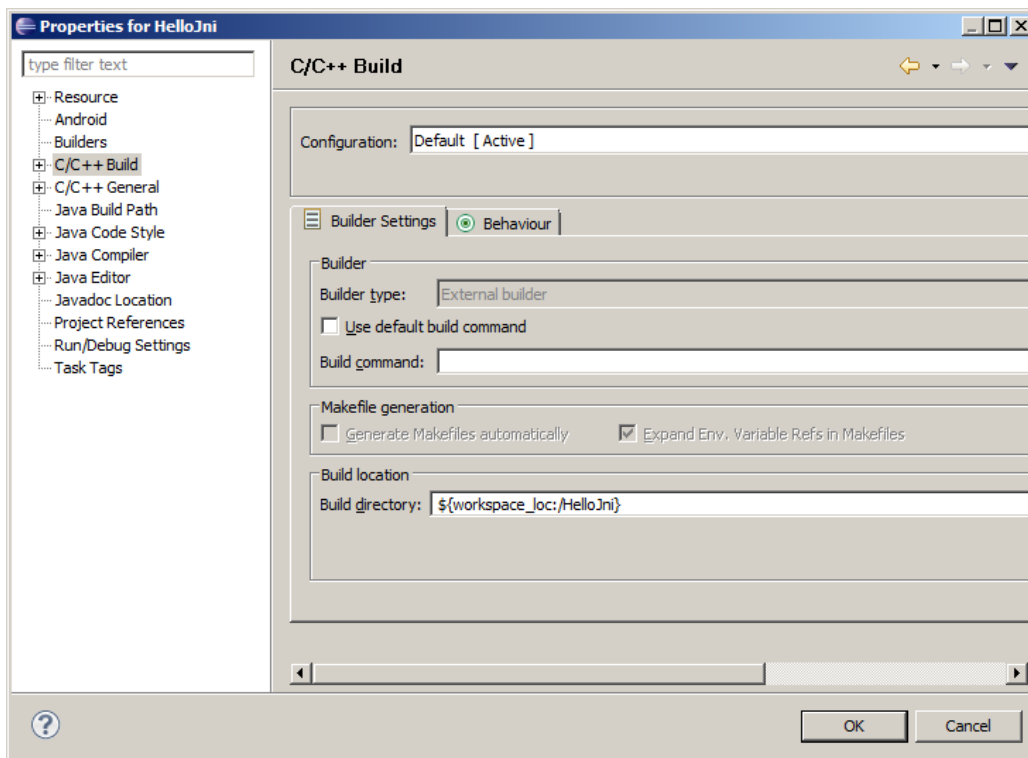
DEBUGGING ANDROID NDK SAMPLES

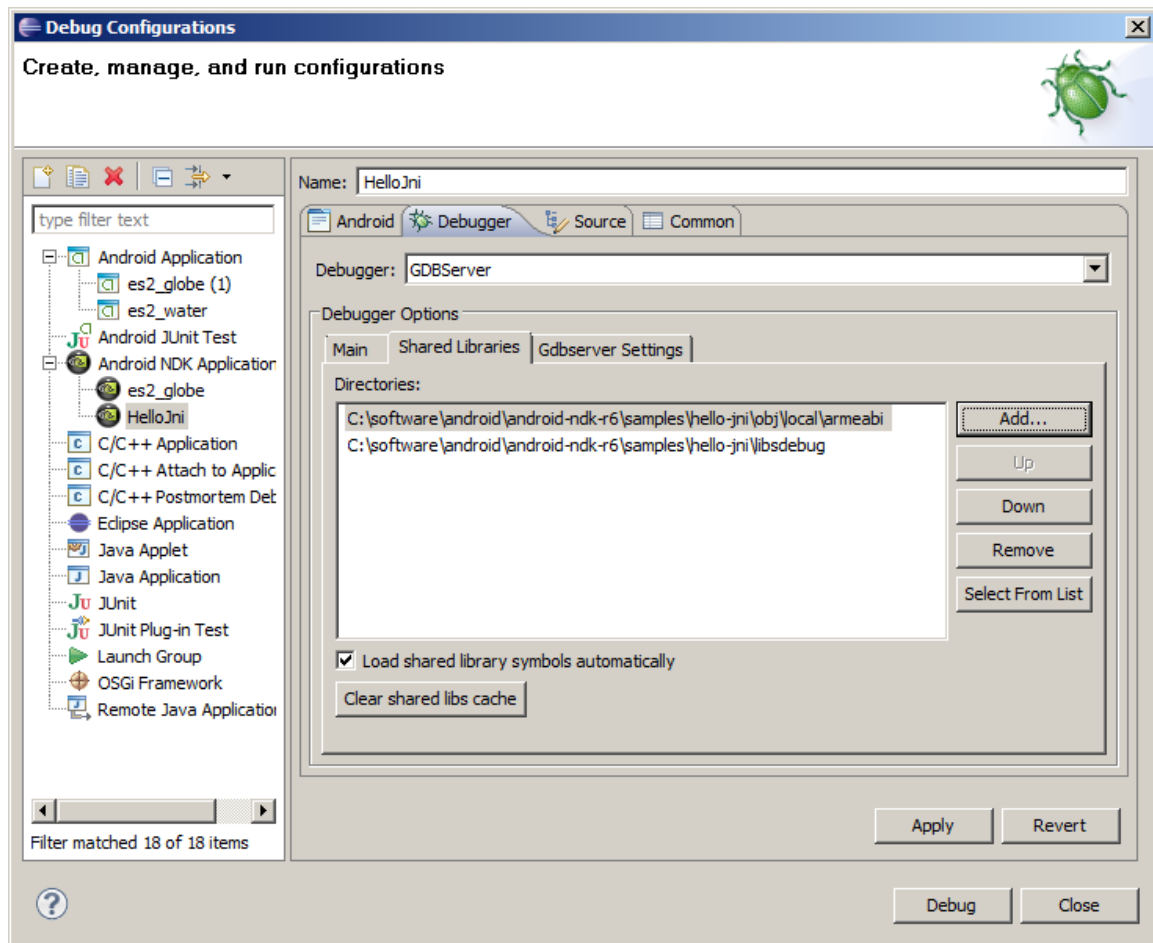
To debug Android NDK sample projects, you need to create Android project from existing source, convert it to C/C++ project (refer “Tegra Android Setup Guide” which can be downloaded from <http://developer.nvidia.com/tegra/downloads> for details) and then start debugging. Remember adding ‘NDK_DEBUG=1’ to your build command-line if you use ndk-build to build your application.

DEBUGGING YOUR OWN PROJECT BUILT OUTSIDE OF ECLIPSE

APK projects built using build systems other than Eclipse (such as those built using command-line ndk-build and Java “ant”) must be imported into Eclipse in order to debug them via NVDM. Generally, the process is analogous to that for setting up a debugging project for the Android NDK samples: importing the Android Java/APK project into Eclipse from existing source and then converting to a C/C++ project. Note that Eclipse C/C++ projects include a C/C++ build command. Since your custom-built application does not use Eclipse’s build system, you may need to set

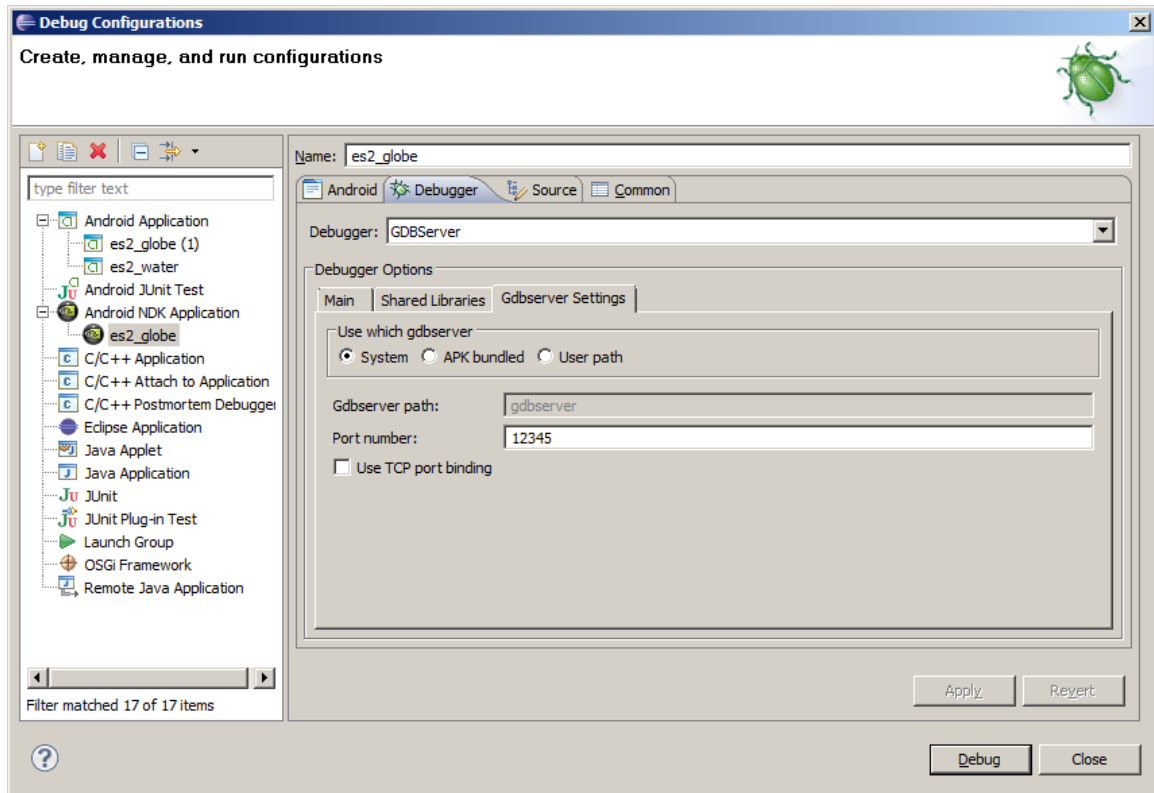
the C/C++ build command to be custom (and set it to be blank). Failing to do so will cause Eclipse to run “make” on your project directory, which is likely to fail. Finally, NVDM’s default search paths for native code symbols may not match your application’s custom build tree. This can cause missed breakpoints. In this case, you may need to set the shared library search path manually after you create the debug configuration, before launching your first debugging session.





CONFIGURING GDBSERVER SETTINGS

NVDM provides users the ability to use a custom gdbserver binary. To configure this setting, select the menu item “Run -> Debug Configurations”. Navigate to the project under “Android NDK Application”, select it, and then click on the “Debugger” tab. Click on the “Gdbserver Settings” tab under “Debugger Options”.



By default, the System setting is selected, which uses the gdbserver binary in the system path. You may select to use an APK bundled gdbserver, or choose a path to a gdbserver binary located on the target device. Make sure to select a TCP port number that is not already in use by another application (default is 12345). By default NVDM uses TCP Sockets, you may disable this feature and use Unix Domain Sockets on your device by disabling the “Use TCP port binding” option, it requires a gdbserver binary with Unix Domain Socket support compiled in.


USING AN APK BUNDLED GDBSERVER

If you select to use the APK bundled gdbserver, you must set the project to be “Debuggable”. To accomplish this, find the “AndroidManifest.xml” file in the project and open it. Select the “Application” tab. Click the “Debuggable” drop-down, and then click “true”. Save the file.

es2_globe Manifest

Android Manifest Application

▼ **Application Toggle**

 The [application](#) tag describes application-level components contained in the package, as well as general application attributes.

☒ Define an <application> tag in the AndroidManifest.xml

▼ **Application Attributes**

Defines the attributes specific to the application.

Name	<input type="text"/>	<input type="button" value="Browse..."/>	Enabled	<input type="text"/>
Theme	<input type="text" value="@android:style/Theme.No"/>	<input type="button" value="Browse..."/>	Debuggable	<input type="text" value="true"/>
Label	<input type="text" value="@string/app_name"/>	<input type="button" value="Browse..."/>	Vm safe mode	<input type="text"/>

TROUBLESHOOTING

LIMITATIONS

Missing Breakpoints

The current version of the plugin may be missing breakpoint in native code executed before GDB is attached. We recommend that the developer wanting to debug native code that could be executed during this transient period add a `while(i)` with `int i` would be initialized to a non-zero value, and changing the `i` to zero once GDB is attached, in order to resume the debugging session and hit the breakpoint.

REPORTING ISSUES

In case of problems working with the NVIDIA Debug Manager for Android NDK plugin the following information might be helpful to provide with a bug report:

- ▶ Exact versions of Android SDK and Android NDK.
- ▶ Exact versions of installed Eclipse software (From Eclipse it is easy to obtain this information from “Help” / “About Eclipse SDK” / “Installation Details” / “Configuration”).
- ▶ Eclipse error log which resides in `$workspace_location/.metadata/.log`, where `$workspace_location` is the location of your Eclipse workspace.
- ▶ Android console output.
- ▶ General description what the problem is and how to reproduce the problem.

and post the issue to the following Tegra Android Development support forum with a subject line referring to the Debug Manager plugin:

<http://developer.nvidia.com/tegra/forums/tegra-forums/android-development>

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, and <add all the other product names listed in this document> are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2010 NVIDIA Corporation. All rights reserved.