ARM® DS-5

Version 5.20

Getting Started Guide

ARM

ARM DUI0478T

ARM® DS-5

Getting Started Guide

Release Information

Document History

Issue	Date	Confidentiality	Change
A	30 June 2010	Non-Confidential	First release
B	30 September 2010	Non-Confidential	Update for DS-5 version 5.2
C	30 November 2010	Non-Confidential	Update for DS-5 version 5.3
D	30 January 2011	Non-Confidential	Update for DS-5 version 5.4
F	30 July 2011	Non-Confidential	Update for DS-5 version 5.6
G	30 September 2011	Non-Confidential	Update for DS-5 version 5.7
H	30 November 2012	Non-Confidential	Update for DS-5 version 5.8
I	28 February 2012	Non-Confidential	Update for DS-5 version 5.9
J	30 May 2012	Non-Confidential	Update for DS-5 version 5.10
K	30 July 2012	Non-Confidential	Update for DS-5 version 5.11
L	30 October 2012	Non-Confidential	Update for DS-5 version 5.12
M	15 December 2012	Non-Confidential	Update for DS-5 version 5.13
N	15 March 2013	Non-Confidential	Update for DS-5 version 5.14
O	14 June 2013	Non-Confidential	Update for DS-5 version 5.15
P	13 September 2013	Non-Confidential	Update for DS-5 version 5.16
Q	13 December 2013	Non-Confidential	Update for DS-5 version 5.17
R	14 March 2014	Non-Confidential	Update for DS-5 version 5.18
S	27 June 2014	Non-Confidential	Update for DS-5 version 5.19
T	17 October 2014	Non-Confidential	Update for DS-5 version 5.20

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents
ARM® DS-5 Getting Started Guide

List of Figures
ARM® DS-5 Getting Started Guide

List of Tables
ARM® DS-5 Getting Started Guide

Preface

This preface introduces the *ARM® DS-5 Getting Started Guide*.
It contains the following:

About this book

This book gives an overview of the product and includes tutorials that show you how to run and debug Linux applications, bare-metal, *Real-Time Operating System* (RTOS), Linux, and Android platforms.

Using this book

This book is organized into the following chapters:

Chapter 1 ARM® DS-5 Product Overview

Gives an overview of the main features of ARM® DS-5.

Chapter 2 ARM® DS-5 Tutorials

Describes how to run and debug applications using ARM DS-5 tools.

Chapter 3 ARM® DS-5 Installation and Examples

Describes the installation and licensing requirements and provides information on the examples provided with ARM DS-5.

Glossary

The ARM Glossary is a list of terms used in ARM documentation, together with definitions for those terms. The ARM Glossary does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See the *ARM Glossary* for more information.

Typographic conventions

italic
: Introduces special terminology, denotes cross-references, and citations.

bold
: Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
: Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`monospace`
: Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
: Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
: Denotes language keywords when used outside example code.

<and>
: Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title.
- The number ARM DUI0478T.
- The page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

——— **Note** ———

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

———————————

Other information

- *ARM Information Center*.
- *ARM Technical Support Knowledge Articles*.
- *Support and Maintenance*.
- *ARM Glossary*.

Chapter 1
ARM® DS-5 Product Overview

Gives an overview of the main features of ARM® DS-5.

It contains the following sections:

1.1 About DS-5

DS-5 is a professional software development solution for Linux-based systems and bare-metal embedded systems, covering all stages in development from boot code and kernel porting to application and bare-metal debugging including performance analysis.

It includes:

- DS-5 Debugger is a graphical debugger supporting end-to-end software development on ARM processor-based targets and *Fixed Virtual Platform* (FVP) targets.
- Eclipse for DS-5 is an *Integrated Development Environment* (IDE) that combines the Eclipse IDE from the Eclipse Foundation with the compilation and debug technology of the ARM tools.
- *Fixed Virtual Platform* (FVP) enable development of software without the requirement for actual hardware.
- ARM Streamline is a graphical performance analysis tool that enables you to transform sampling data and system trace into reports that present the data in both visual and statistical forms.
- ARM Compiler tools enable you to build applications and libraries suitable for bare-metal embedded systems.
- Debug hardware configuration utilities enable you to configure the debug hardware unit that provides the interface between your development platform and your PC.
- Dedicated examples, applications, and supporting documentation to help you get started with using the DS-5 tools.

Related concepts

1.2 About Eclipse for DS-5 on page 1-14.
1.3 About DS-5 Debugger on page 1-15.
1.4 About Fixed Virtual Platform (FVP) on page 1-16.
1.5 About ARM® Compiler on page 1-17.
1.6 About ARM® Streamline Performance Analyzer on page 1-19.
1.8 About debug hardware configuration utilities on page 1-21.

Related references

3.4 Licensing and product updates on page 3-82.
3.5 Documentation provided with DS-5 on page 3-83.
3.6 Examples provided with DS-5 on page 3-84.

Related information

DS-5 Knowledge Articles.

1.2 About Eclipse for DS-5

Eclipse for DS-5 is an *Integrated Development Environment* (IDE) that combines the Eclipse IDE from the Eclipse Foundation with the compilation and debug technology of the ARM tools.

Some third-party compilers are compatible with DS-5. For example, the GNU Compiler tools enable you to compile bare-metal, Linux kernel and Linux applications for ARM targets.

It includes:

Project manager
: The project manager enables you to perform various project tasks such as adding or removing files and dependencies to projects, importing, exporting, or creating projects, and managing build options.

Editors
: Editors enables you read, write, or modify C/C++ or ARM assembly language source files.

Perspectives and views
: Perspectives provide customized views, menus, and toolbars to suit a particular type of environment. DS-5 uses the C/C++ and **DS-5 Debug** perspectives.

Related concepts

1.1 About DS-5 on page 1-13.

Related information

Getting started with Eclipse.

1.3 About DS-5 Debugger

DS-5 Debugger is a graphical debugger supporting end-to-end software development on ARM processor-based targets and *Fixed Virtual Platform* (FVP) targets.

It makes it easy to debug Linux and bare-metal applications with comprehensive and intuitive views, including synchronized source and disassembly, call stack, memory, registers, expressions, variables, threads, breakpoints, and trace.

Using the Debug Control view, you can single-step through applications at source-level or instruction-level and see the other views update as the code is executed. Setting breakpoints or watchpoints can assist you by stopping the application and enabling you to explore the behavior of the application. You can also use the Trace view on some targets to trace function executions in your application with a timeline showing the sequence of events.

You can also debug using the **DS-5 Command Prompt** command-line console.

Related concepts

1.1 About DS-5 on page 1-13.
1.4 About Fixed Virtual Platform (FVP) on page 1-16.

Related information

Getting started with the debugger.

1.4 About *Fixed Virtual Platform* (FVP)

Fixed Virtual Platform (FVP) enable development of software without the requirement for actual hardware. The functional behavior of the FVP is equivalent to real hardware from a programmers view.

Absolute timing accuracy is sacrificed to achieve fast simulated execution speed. This means that you can use a model for confirming software functionality, but you must not rely on the accuracy of cycle counts, low-level component interactions, or other hardware-specific behavior.

DS-5 provides Cortex®-A8, quad-core Cortex-A9, and AEMv8 (DS-5 Ultimate Edition only) executables. You can also connect to a variety of other ARM and third-party simulation models.

The executables are located in `tools_directory`. You can use them to test your applications from either the command-line or within Eclipse.

Related concepts

1.1 About DS-5 on page 1-13.

Related references

3.3 Installation directories on page 3-81.

Related information

Fixed Virtual Platform (FVP) Reference.

1.5 About ARM® Compiler

ARM Compiler tools enable you to build applications and libraries suitable for bare-metal embedded systems.

DS-5 is supplied with two versions of ARM Compiler for compiling bare-metal applications:

- ARM Compiler 5 - Includes support for all ARM architectures from ARMv4 to ARMv7 that are currently supported by ARM.

  Note

  All architectures before ARMv4 are obsolete and are no longer supported by ARM Compiler 5.

- ARM Compiler 6 - Supports ARMv8 architecture.

  Note

  ARM Compiler 6 is only supported on 64-bit host platforms and is only supplied in the 64-bit installable product.

The ARM Compiler tools are located in *tools_directory*. You can use them to build your applications from either the command-line or within Eclipse.

Table 1-1 ARM Compiler tools

Tool	Description
armar	Librarian. This enables sets of ELF format object files to be collected together and maintained in archives or libraries. You can pass such a library or archive to the linker in place of several ELF files. You can also use the archive for distribution to a third party for application development.
armasm	Assembler. This assembles ARM and Thumb assembly language sources.
armcc	Compiler. This compiles your C and C++code. It supports inline and embedded assemblers, and also includes the NEON vectorizing compiler.
armclang	Compiler and Assembler. This compiles C and C++ code, and assembles A32, A64, and T32 GNU syntax assembly code.
armlink	Linker. This combines the contents of one or more object files with selected parts of one or more object libraries to produce an executable program.
fromelf	Image conversion utility. This can also generate textual information about the input image, such as disassembly and its code and data size.

Note

ARM Compiler is license managed. Specific features depend on your installed license.

For example, a license might limit the use of ARM Compiler to specific processor types, or place a maximum limit on the size of images that can be produced, or require that you work with proprietary format (ORC) objects instead of ELF format objects.

You can enable additional features by purchasing a license for the full DS-5 suite. Contact your tools supplier for details.

Related concepts

1.1 About DS-5 on page 1-13.

Related references

3.3 Installation directories on page 3-81.

Related information

Creating a new C or C++ project.

1.6 About ARM® Streamline Performance Analyzer

Describes ARM Streamline and also lists the main features included in DS-5.

ARM Streamline is a graphical performance analysis tool that enables you to transform sampling data and system trace into reports that present the data in both visual and statistical forms.

ARM Streamline uses hardware performance counters with kernel metrics to provide an accurate representation of system resources.

Related concepts

1.1 About DS-5 on page 1-13.

Related tasks

2.11 Performance analysis of threads application running on ARM® Linux on page 2-49.

Related information

Using ARM Streamline.

1.7 Debug options supported by DS-5

DS-5 supports various debug options.

Debug adapters vary in complexity and capability but, combined with software debug agents, they provide high-level debug functionality for the target that is being debugged, for example:

- Reading/Writing registers.
- Setting breakpoints.
- Reading from memory.
- Writing to memory.

——— **Note** ———————

A debug adapter or connection is not the application being debugged, nor the debugger itself.

———————————————

Supported ARM debug hardware adapters include:

- ARM DSTREAM

  ——— **Note** ———————

  You must use DSTREAM for ARMv8 development.

  ———————————————

- ARM RVI™
- KEIL® ULINK™2
- KEIL® ULINK™pro
- KEIL® ULINK™pro D

Supported debug connections include:

- ARM VSTREAM
- CMSIS-DAP
- CADI
- Ethernet to gdbserver

Supported third-party debug hardware adapters include:

- Altera USB-Blaster II
- Yokogawa Digital Computer Corporation adviceLUNA (JTAG ICE)

  ——— **Note** ———————

  DS-5 Debugger can connect to Altera Arria V SoC and Cyclone V SoC boards using Altera USB-Blaster and USB-Blaster II debug units.

  To enable the connections, ensure that the environment variable *`QUARTUS_ROOTDIR`* is set and contains the path to the Altera Quartus tools installation.

  On Windows, this environment variable is usually set by the Quartus tools installer. On Linux, you might have to manually set the environment variable to the Altera Quartus tools installation path. For example, `~/altera/13.0/qprogrammer`.

  For information on installing device drivers for USB-Blaster and USB-Blaster II, consult your Altera Quartus tools documentation.

  ———————————————

Related information

Setting up the ARM DSTREAM Hardware.

Setting up the ARM RVI Hardware.

1.8 About debug hardware configuration utilities

Debug hardware configuration utilities enable you to configure the debug hardware unit that provides the interface between your development platform and your PC.

The following utilities are provided:

Debug Hardware Config IP
Used to configure the IP address on a debug hardware unit.

Debug Hardware Update
Used to update the firmware and devices on a debug hardware unit.

Debug Hardware Config
Used to configure networking for a debug hardware unit.

Related concepts

1.1 About DS-5 on page 1-13.

Related information

Using the Debug Hardware Configuration Utilities.

1.9 About registering a new compiler toolchain

You can use a different compiler toolchain other than the one installed with DS-5.

If you want to build projects using a toolchain that is not installed with DS-5, you must first register the toolchain you want to use. You can register toolchains:

- using the **Preferences** dialog in Eclipse for DS-5.
- using the `add_toolchain` utility from the DS-5 command prompt.

You might want to register a compiler toolchain if:

- You upgrade your version of DS-5 but you want to use an earlier version of the toolchain that was previously installed.
- You install a newer version or older version of the toolchain without re-installing DS-5.

When you register a toolchain, the toolchain is be available for new and existing projects in DS-5.

——— **Note** ———

You can only register ARM or GCC toolchains.

———————————

Related tasks

2.18 Registering a compiler toolchain from Eclipse on page 2-74.
2.17 Registering a compiler toolchain from the DS-5 command prompt on page 2-70.

Chapter 2
ARM® DS-5 Tutorials

Describes how to run and debug applications using ARM DS-5 tools.

It contains the following sections:

2.1 Configuring a compiler toolchain for the DS-5 command prompt

When you want to compile or build from the DS-5 command prompt, you can select the compiler toolchain you want to use. You can set this as the default toolchain so that you do not need to select a toolchain every time you start the DS-5 command prompt.

Note

By default, the DS-5 command prompt is not configured with a compiler toolchain.

On Linux, run `suite_exec` with the `--toolchain` *`name`* option to configure a compiler toolchain for the DS-5 environment. Run `suite_exec` with no arguments for the list of available toolchains. For example, to use the ARM Compiler 5 toolchain included in DS-5, run:

```
<DS-5 install_directory>/bin/suite_exec --toolchain "ARM Compiler 5 (DS-5 built-in)"
bash --norc
```

On Windows, to set a default compiler toolchain for the DS-5 command prompt, use the `select_default_toolchain` command.

The following procedure describes the steps for using `select_default_toolchain` on Windows.

Procedure

1. Open the DS-5 command prompt, by selecting **Start** > **All Programs** > **DS-5 Command Prompt**.
2. Enter `select_default_toolchain` on the DS-5 command prompt. This lists the available compiler toolchains.

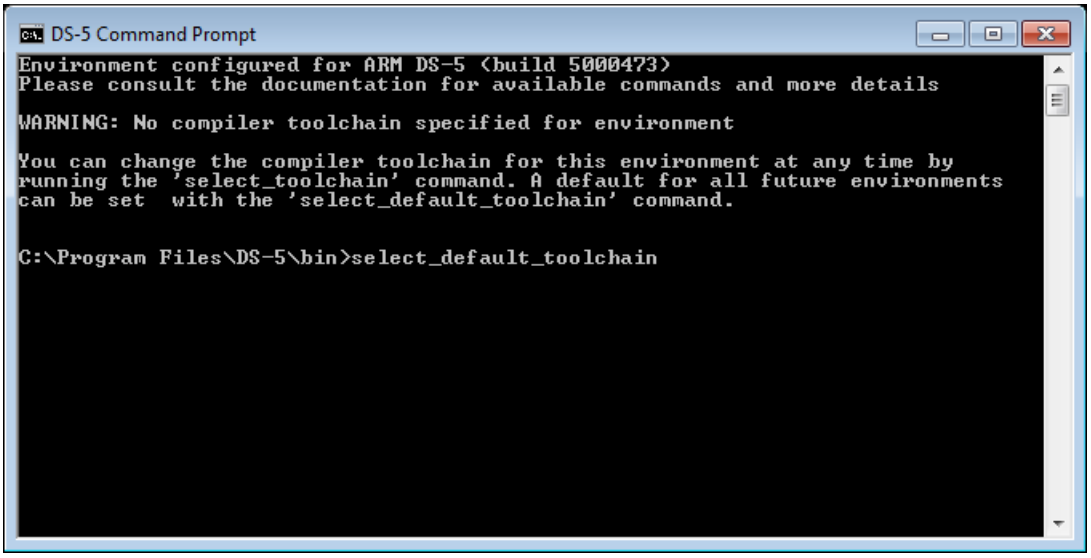


Figure 2-1 Configuring a default toolchain

3. Select your default compiler toolchain. For example, enter `1` to select the ARM Compiler 5 that is built-in with DS-5. This configures the DS-5 command prompt for the selected toolchain. When you open a new DS-5 command prompt, the environment is still configured for your selected toolchain.

Note

To configure a compiler toolchain for the current DS-5 command prompt, without changing the default toolchain, use the `select_toolchain` command.

2.2 Installing DS-5 into a custom Eclipse environment

Follow these instructions to install DS-5 into your custom Eclipse environment.

Prerequisites

Ensure that you have a full installation of DS-5 and the following packages installed on your target workstation before you attempt to install DS-5 into a custom Eclipse environment.

- 32 or 64-bit version of Java SE 7 or higher.
- 32 or 64-bit version of Eclipse version 4.3.1 (Kepler).

——— **Note** ———

Your Eclipse installation must be compatible with your Java installation. You must use 32-bit Eclipse with 32-bit Java, and 64-bit Eclipse with 64-bit Java.

———————————

The following optional Eclipse packages are required for Python development:

- *Java Development Toolkit* (JDT).
- *Python Development* (PyDev).

——— **Note** ———

Some tools such as compilers or models must be defined in the `PATH` environment variable of your OS to be available within your custom Eclipse. Add the `<DS-5 install_directory>/bin` directory to your `PATH` environment variable and restart Eclipse.

If running DS-5 from a command shell:

- For Windows, enter `set PATH=<DS-5 install_directory>\bin;%PATH%`
- For Linux, enter `export PATH=<DS-5 install_directory>/bin:$PATH`

To make the change permanent, so that the modified path is available in future sessions:

- For Windows, right-click **My Computer** > **Properties** > **Advanced system settings** > **Environment Variables** and under **User Variables**, create a `PATH` variable with the value `<DS-5 install_directory>\bin`, or else append `;<DS-5 install_directory>\bin` to any existing `PATH` variable.
- For Linux, set up the `PATH` in the appropriate shell configuration file. For example, in `.bashrc`, add the line `export PATH=<DS-5 install_directory>/bin:$PATH`

———————————

Procedure

1. Launch your custom Eclipse.
2. Select **Help** > **Install New Software...** to display the **Install** dialog box.
3. Click **Add...** to display the Add Repository dialog box.
   a) Enter a **Name** for the update location, for example: `DS-5 Update Site`.
   b) Click **Archive** and select the archive file available in the following directory: `<DS-5 install_directory>\sw\eclipse\update-site.zip`.
   c) Click **Open**.
4. Click **OK** in the Add Repository dialog box to add the update location.
5. Select **ARM DS-5** in the list of items.
6. Click **Next**.
7. Review the list of items to be installed.

Note

DS-5 has dependencies on external packages. Unless they are already installed, access to the Eclipse website is required to locate and install these packages.

8. Click **Next**.
9. Read the license agreements and accept them. If you do not accept the license agreements, you cannot install DS-5.
10. Click **Finish**.
11. When prompted, click **Restart Now** to restart Eclipse and complete the installation.
12. If the license is not configured, then a dialog box opens to enable you to add a license. Alternatively, you can configure this later by selecting **Help** > **ARM License Manager...** from the main menu.
13. Some Eclipse plug-ins need access to the DS-5 installation folder and might not work if this is not defined in the Eclipse preferences. To set the path to the DS-5 installation folder.
    a) Open Eclipse preferences from **Window** > **Preferences**.
    b) Select the **DS-5** item.
    c) **Browse** and select the **DS-5 install directory**.
14. Set up the **Configuration Database** for DS-5:
    a) Select **Window** > **Preferences** from the main menu.
    b) Expand the **DS-5** group and select **Configuration Database**.
    c) Click **Add** to display the **Add configuration database location** dialog box.
    d) Enter a **Name** for the configuration database, for example: `DS-5 Configuration database`.
    e) Enter the **Location** or click **Browse** and point it to the folder containing the database which is located at: *`<DS-5 install_directory>`*`\sw\debugger\configdb`.
    f) Ensure that the new target database is at the top of the list.
    g) Click **Rebuild database...** to clear any system caches and validate the platform definitions.
    h) Click **OK** to close the **Preferences** dialog box.

Related concepts

1.1 About DS-5 on page 1-13.

Related references

2.15 Managing DS-5 licenses on page 2-60.

Related information

Preferences dialog box.
Java SE Downloads.
Eclipse.

2.3 Importing the example projects into Eclipse

To use the example projects provided with DS-5, you must first import them.

Procedure

1. Launch Eclipse:
   - On Windows, select **Start** > **All Programs** > **ARM DS-5** > **Eclipse for DS-5**.
   - On Linux, enter `eclipse` in the Unix bash shell.
2. ARM recommends that you create a new workspace for the example projects so that they remain separate from your own projects. To do this you can either:
   - Create a new workspace directory during the start up of Eclipse.
   - If Eclipse is already open, select **File** > **Switch Workspace** > **Other** from the main menu.
3. Select **Cheat Sheet...** from the **Help** menu.
4. Expand the **ARM Eclipse for DS-5** group.
5. Select **Automatically Import the DS-5 Example Projects into the Current Workspace** from the list of ARM cheat sheets.
6. Click **OK**.
7. Follow the steps in the cheat sheet to import all the DS-5 example projects into your workspace.

When the examples are imported, you can optionally follow the remaining cheat sheet instructions to switch on working sets if required.

Related tasks

2.4 Building the Gnometris project from Eclipse on page 2-28.
2.5 Building the Gnometris project from the command-line on page 2-29.
2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.
2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.
2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.
2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.
2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

About working sets.
Creating a working set.
Changing the top level element when displaying working sets.
Deselecting a working set.

2.4 Building the Gnometris project from Eclipse

Gnometris is an ARM Linux application that you can run and debug on your target. The supplied project contains prebuilt image binaries for the Gnometris application. Use these instructions to rebuild the project.

Procedure

1. Download the optional package containing the example Linux distribution project and the compatible headers and libraries from the ARM website.
2. Import both the `gnometris` and `distribution` example projects from the relevant ZIP archive files into Eclipse.
3. Select the `gnometris` project in the Project Explorer view.
4. Select **Build Project** from the **Project** menu.

   The Gnometris example contains a `Makefile` to build the project. The `Makefile` provides the usual make rules: `clean`, `all`, and `rebuild`.

When you build the Gnometris project, it produces the following applications:

- A stripped version of the application containing no debug information. This is for downloading to the target.
- A larger sized version of the application containing full debug information for use by the debugger when debugging at the source level.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.

2.5 Building the Gnometris project from the command-line on page 2-29.

2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.

2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.

2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.

2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.

2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Working with projects.

2.5 Building the Gnometris project from the command-line

Gnometris is an ARM Linux application that you can run and debug on your target. The supplied project contains prebuilt image binaries for the Gnometris application. Use these instructions to rebuild the project.

Procedure

1. Download the optional package containing the example Linux distribution project and the compatible headers and libraries from the ARM website.
2. Extract both the `gnometris` and `distribution` example projects from the relevant ZIP archive files into a working directory.
3. Open the **DS-5 Command Prompt** command-line console or a Unix bash shell.
4. Navigate to `...\ARMLinux\gnometris`.
5. At the prompt, enter `make`.

   The Gnometris example contains a `Makefile` to build the project. The `Makefile` provides the usual make rules: `clean`, `all`, and `rebuild`.

When you build the Gnometris project, it produces the following applications:

- A stripped version of the application containing no debug information. This is for downloading to the target.
- A larger sized version of the application containing full debug information for use by the debugger when debugging at the source level.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.

2.4 Building the Gnometris project from Eclipse on page 2-28.

2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.

2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.

2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.

2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.

2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

2.6 Loading the Gnometris application on a *Fixed Virtual Platform* (FVP)

You can load the Gnometris application on to an FVP that is running ARM Linux.

An FVP enables you to run and debug applications on your host workstation without using any hardware targets.

A preconfigured FVP connection is available that automatically boots Linux, launches `gdbserver`, and then launches the application.

Procedure

1. Launch Eclipse.
2. Click on the Project Explorer view.
3. Expand the **gnometris** project folder.
4. Right-click on the launch file, **gnometris-FVP-example.launch**.
5. In the context menu, select **Debug As**.
6. Select the **gnometris-FVP-example** entry in the submenu.
7. Debugging requires the **DS-5 Debug** perspective. If the Confirm Perspective Switch dialog box opens, click **Yes** to switch perspective.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.
2.4 Building the Gnometris project from Eclipse on page 2-28.
2.5 Building the Gnometris project from the command-line on page 2-29.
2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.
2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.
2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.
2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Configuring a connection to an FVP.
Debug Configurations - Connection tab.
Debug Configurations - Files tab.
Debug Configurations - Debugger tab.
Debug Configurations - Environment tab.

2.7 Loading the Gnometris application on to an ARM® Linux target

Describes how to load the Gnometris application on to an ARM Linux target.

You can load the Gnometris application on to a target that is running ARM Linux.

DS-5 provides preconfigured target connection settings that connect the debugger to `gdbserver` running on supported ARM architecture-based platforms.

Procedure

1. Obtain the IP address of the target. You can use the `ifconfig` application in a Linux console. The IP address is denoted by the **inet addr**.
2. Boot the appropriate Linux distribution on the target.
3. Launch Eclipse.
4. Transfer the application and related files to the ARM Linux target, run the application, and then connect the debugger. There are several ways to do this:
   - Use a *Secure SHell* (SSH) connection with the *Remote System Explorer* (RSE) provided with DS-5 to set up the target and run the application. When the application is running you can then connect the debugger to the running target.
   - Use an external file transfer utility such as `PuTTY`.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.

2.4 Building the Gnometris project from Eclipse on page 2-28.

2.5 Building the Gnometris project from the command-line on page 2-29.

2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.

2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.

2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.

2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Debug Configurations - Connection tab.

Debug Configurations - Files tab.

Debug Configurations - Debugger tab.

Debug Configurations - Environment tab.

Target management terminal for serial and SSH connections.

Remote Systems view.

2.8 Configuring an RSE connection to work with an ARM® Linux target

Describes how to use an RSE connection to work with an ARM Linux target.

Procedure

1. In the Remote Systems view, click on **Define a connection to remote system** in the Remote Systems view toolbar.
2. In the Select Remote System Type dialog box, expand the **General** group and select **Linux**.

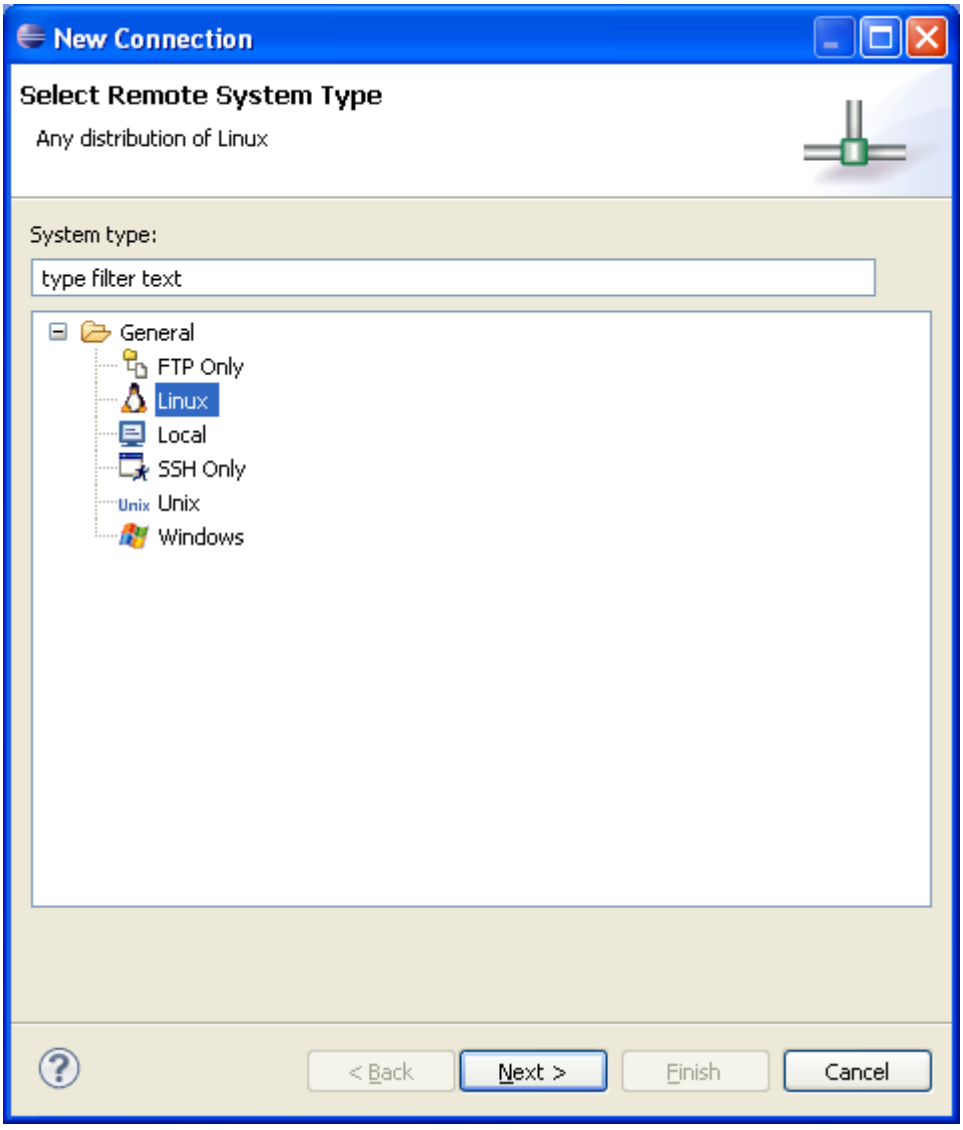


Figure 2-2 Selecting a connection type

3. Click **Next**.
4. In the Remote Linux System Connection, enter the remote target IP address or name in the Host name field.

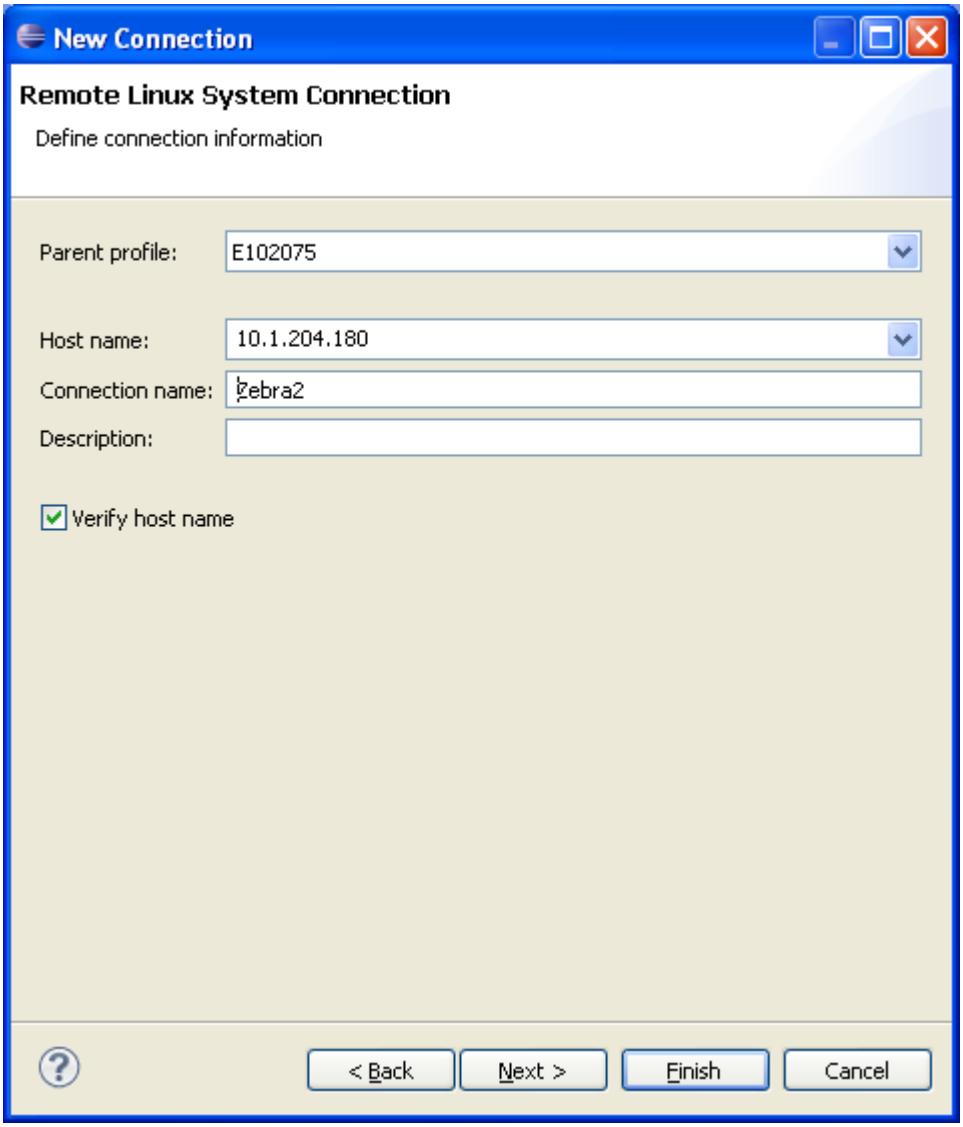


Figure 2-3 Defining the connection information

5. Click **Next**.
6. Select SSH protocol file access.

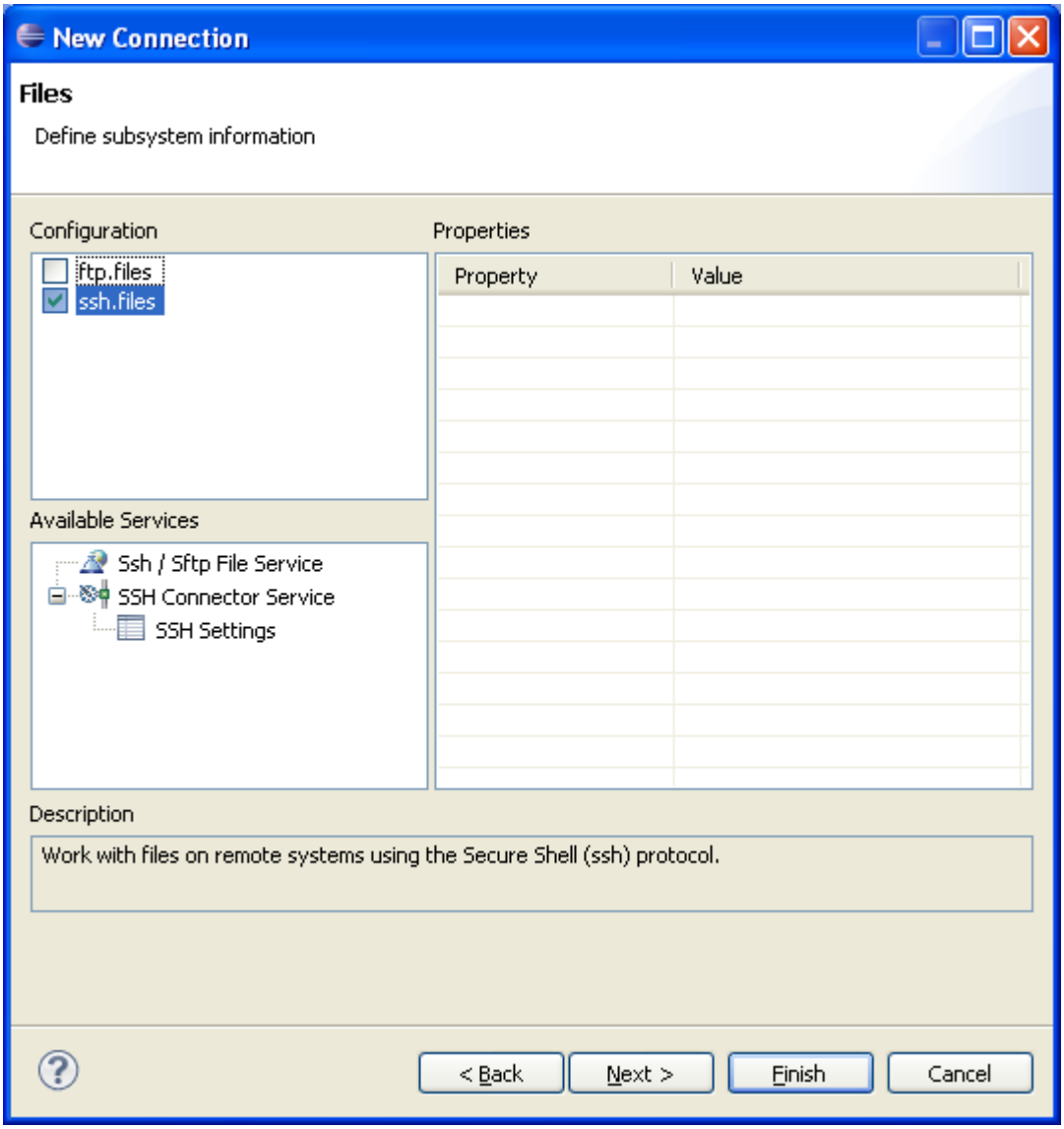


Figure 2-4 Defining the file system

7. Click **Next**.
8. Select the shell processes for Linux systems.

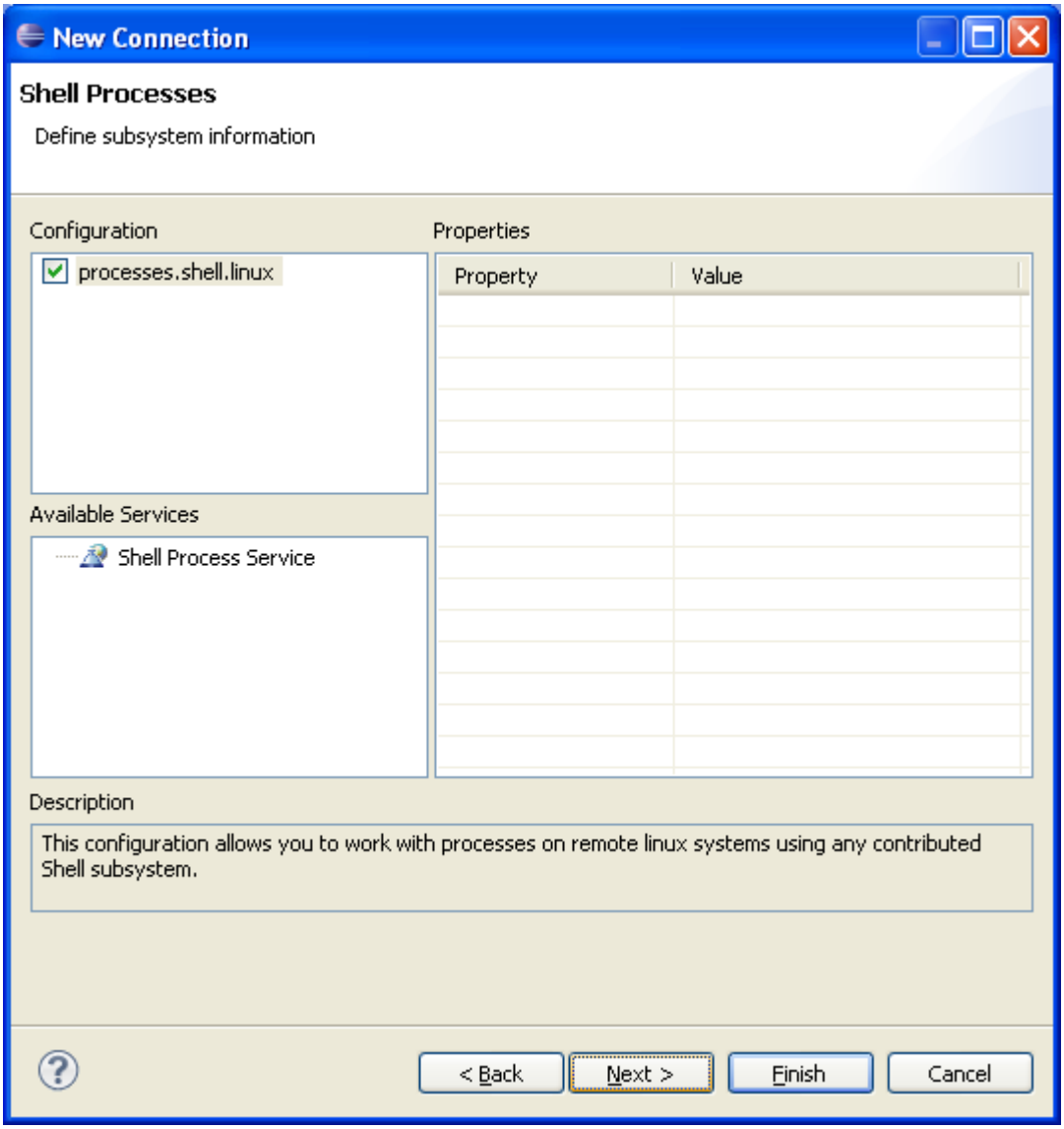


Figure 2-5 Defining the processes

9. Click **Next**.
10. Select SSH shells.

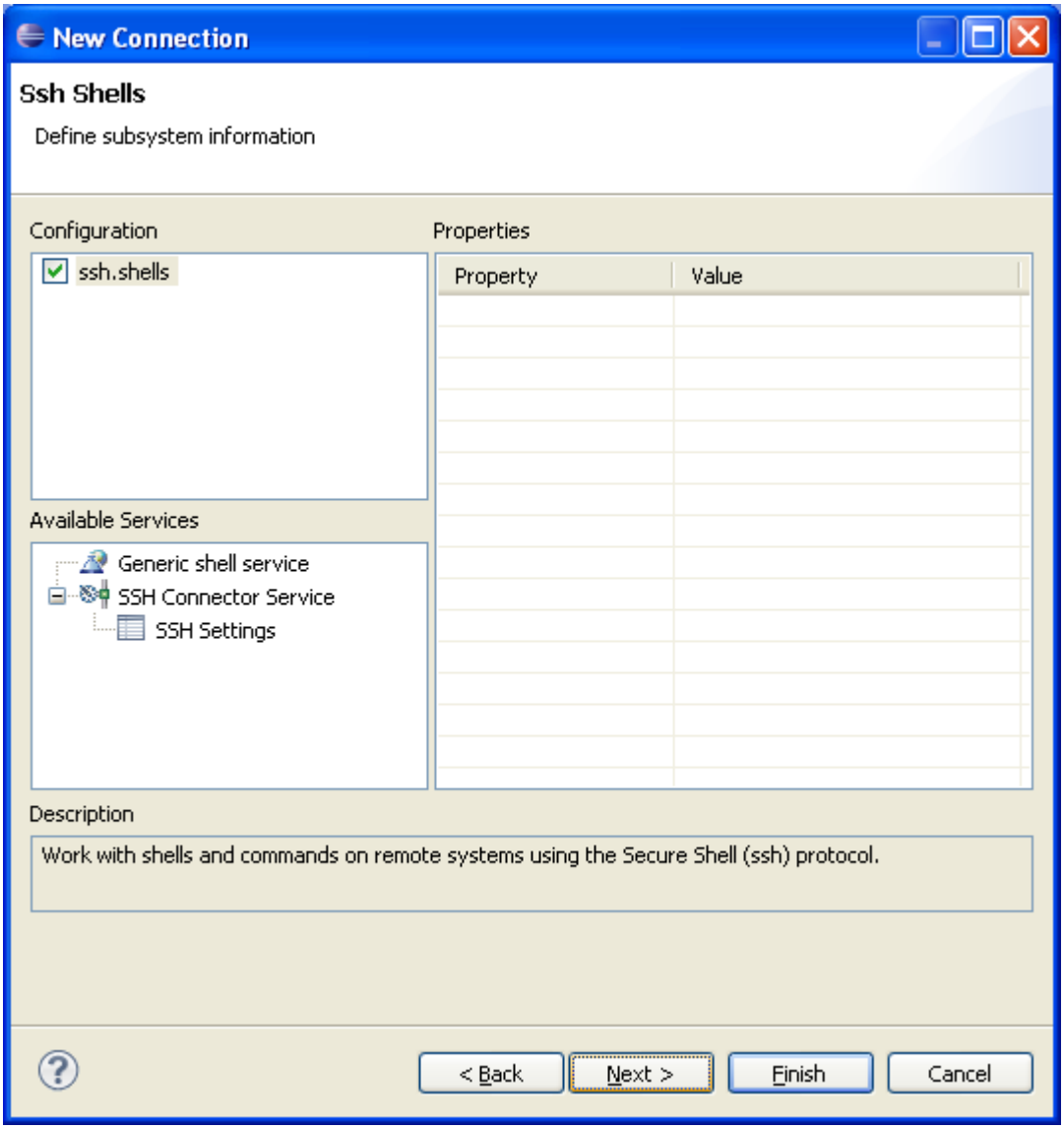


Figure 2-6 Defining the shell services

11. Click **Next**.
12. Select SSH terminals.

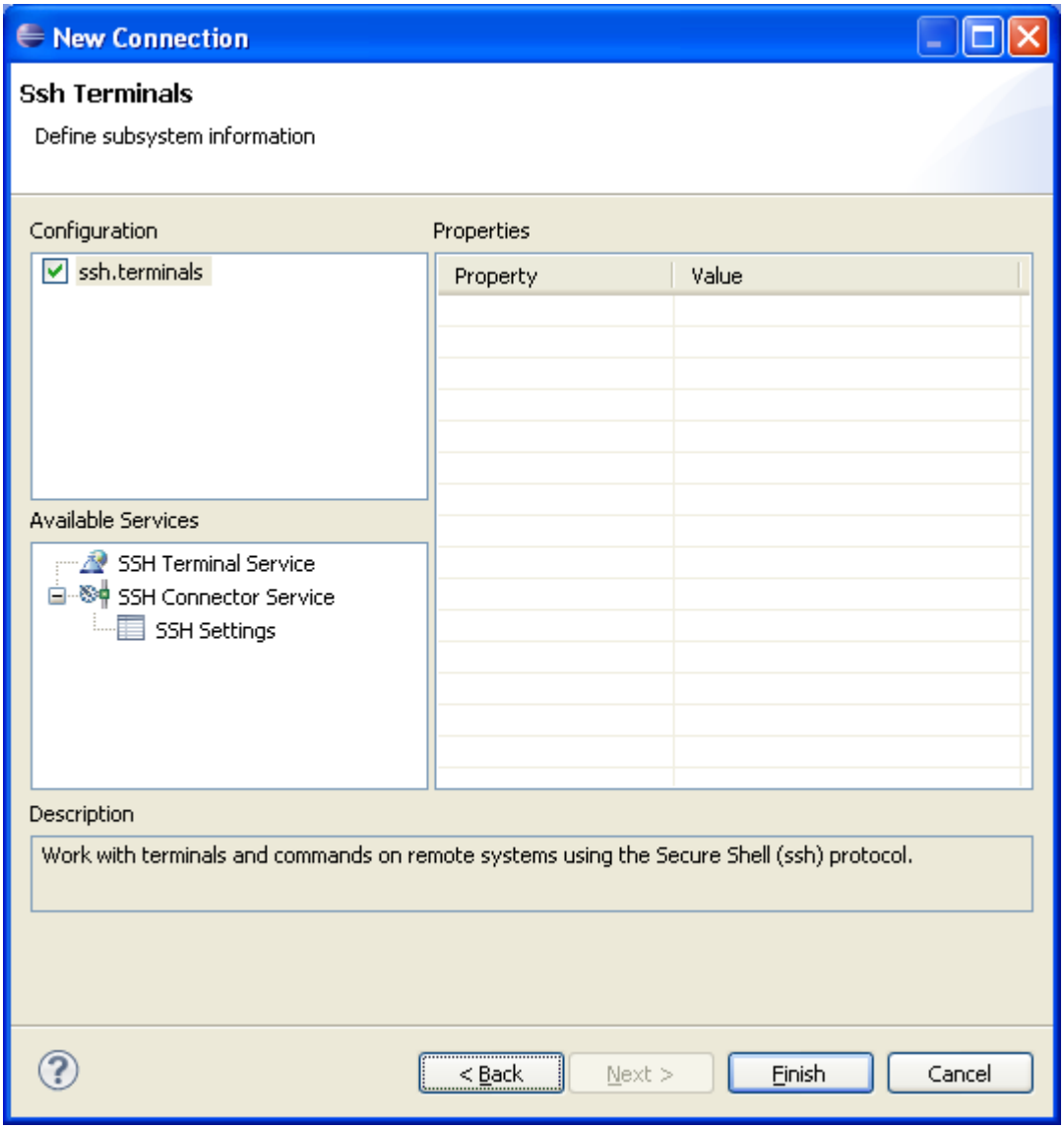


Figure 2-7 Defining the terminal services

13. Click **Finish**.
14. In the Remote Systems view:
    a) Right-click on the Linux target and select **Connect** from the context menu.
    b) In the Enter Password dialog box, enter a **UserID** and **Password** if required.
    c) Click **OK** to close the dialog box.
    d) Copy the required from the local file system on to the target file system. You can do this by dragging and dropping the relevant files in the Remote Systems view.

       This example uses Gnometris which requires copying the stripped version of the Gnometris application, `gnometris`,and the `libgames-support.so` library.
    e) Ensure that the files on the target have execute permissions. To do this, right-click on each file, select **Properties** from the context menu and select the checkboxes as required.

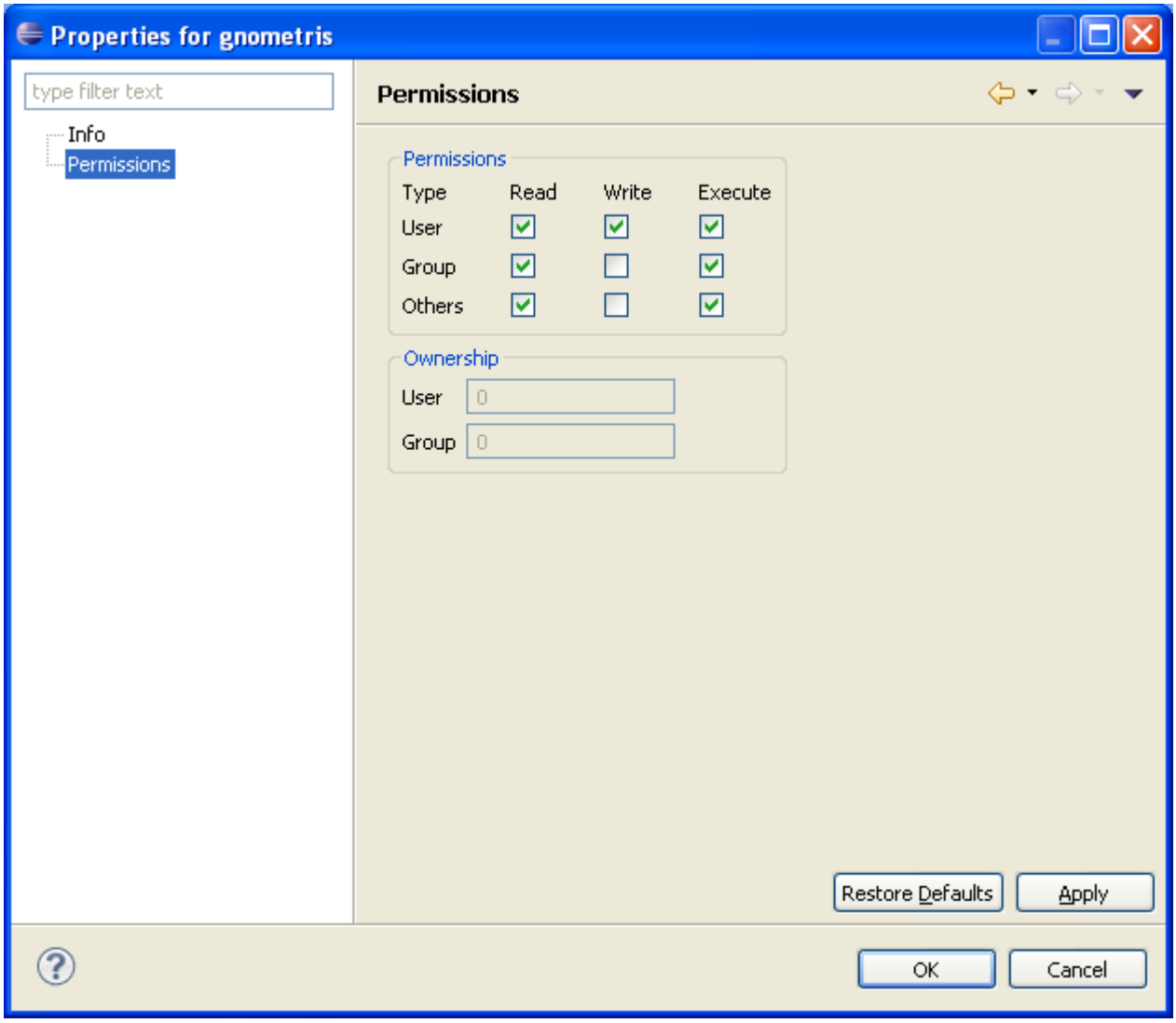


Figure 2-8 Modifying file properties from the Remote Systems view

15. Open a terminal shell that is connected to the target and launch `gdbserver` with the application:
    a) In the Remote Systems view, right-click on **Ssh Terminals**.
    b) Select **Launch Terminal** to open a terminal shell.
    c) In the terminal shell, navigate to the directory where you copied the application, then execute the required commands.

    For example, to launch Gnometris:

    ```
    export DISPLAY=ip:0.0
    gdbserver :port gnometris
    ```

    where:

    `ip`
    : is the IP address of the host to display the Gnometris game

    `port`
    : is the connection port between `gdbserver` and the application, for example `5000`.

    Note

    If the target has a display connected to it then you do not need to use the `export DISPLAY` command.

This section contains the following subsections:

2.8.1 Launching gdbserver with an application

To launch `gdbserver` with the application:

Procedure

1. Open a terminal shell that is connected to the target.
2. In the Remote Systems view, right-click on **Ssh Terminals**.
3. Select **Launch Terminal** to open a terminal shell.
4. In the terminal shell, navigate to the directory where you copied the application, then execute the required commands.

   For example, to launch Gnometris:

   ```
   export DISPLAY=ip:0.0
   gdbserver :port gnometris
   ```

   where:

   `ip`
   : is the IP address of the host to display the Gnometris game

   `port`
   : is the connection port between `gdbserver` and the application, for example `5000`.

   Note

   If the target has a display connected to it then you do not need to use the `export DISPLAY` command.

2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target

Describes how to connect to the Gnometris application that is already running on a ARM Linux target.

Procedure

1. Select **Debug Configurations...** from the **Run** menu.
2. Select **DS-5 Debugger** from the configuration tree and then click on **New** to create a new configuration. Alternatively you can select an existing DS-5 Debugger configuration and then click on **Duplicate** from the toolbar.
3. In the Name field, enter a suitable name for the new configuration.
4. Click on the **Connection** tab to see the target and connection options.
5. In the Select target panel:
   a) Select the required platform, for example, **beagleboard.org- OMAP_3530**.
   b) Select **Connect to already running gdbserver** for the debug operation.
6. In the Connections panel, for the connection between `gdbserver` and the application:
   a) Enter the IP address of the target.
   b) Enter the port number.

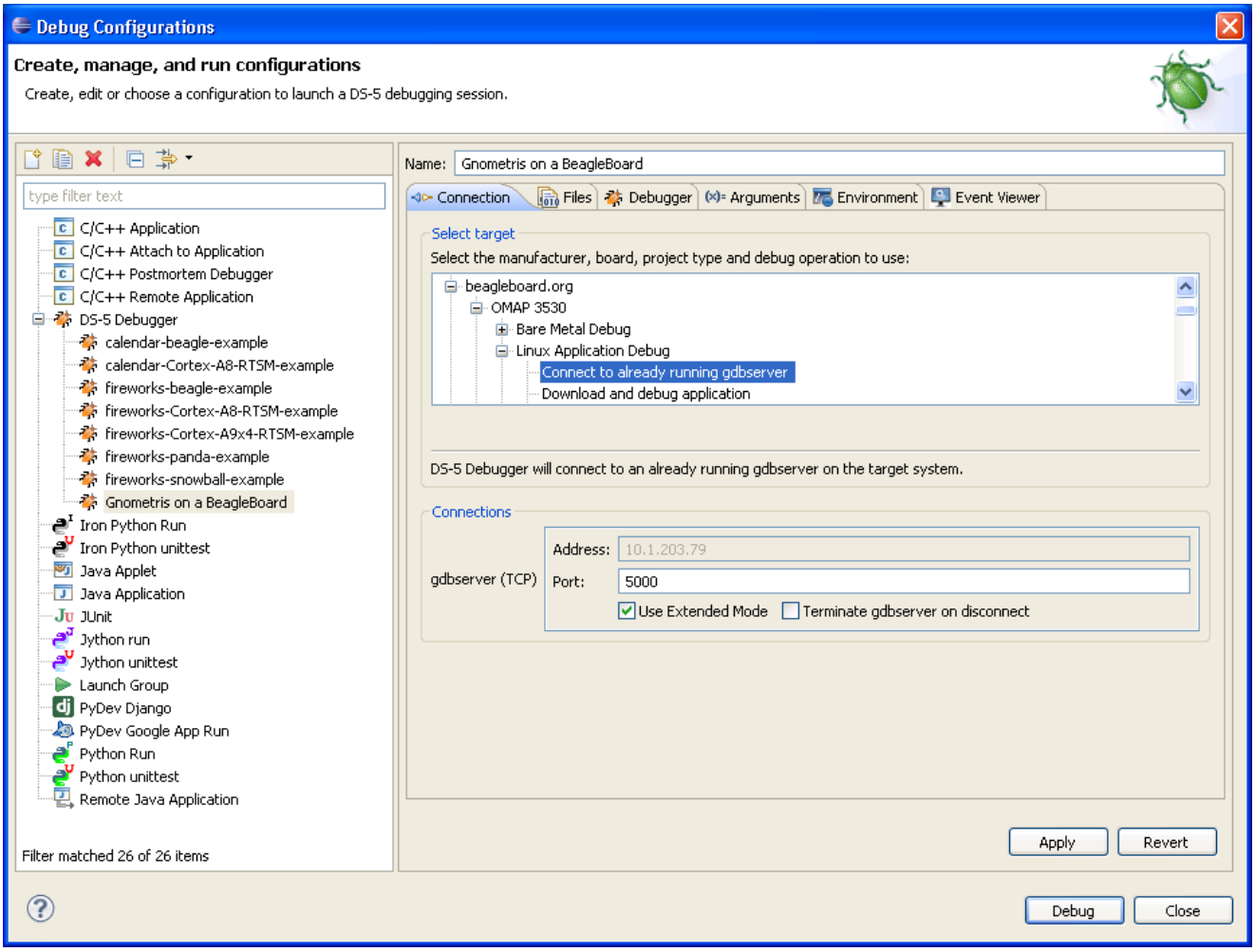


Figure 2-9 Typical connection configuration for a Beagle board

7. Click on the **Files** tab to see the file options.
8. In the Files panel:
   a) Select **Load symbols from file** and then select the application image containing debug information. For example: `H:\workspace\gnometris\gnometris`.
   b) Click **Add a new resource to the list** to add another file entry.
   c) Select **Load symbols from file** and then select the shared library that is required by the Gnometris application. For example: `H:\workspace\gnometris\libgames-support.so`.

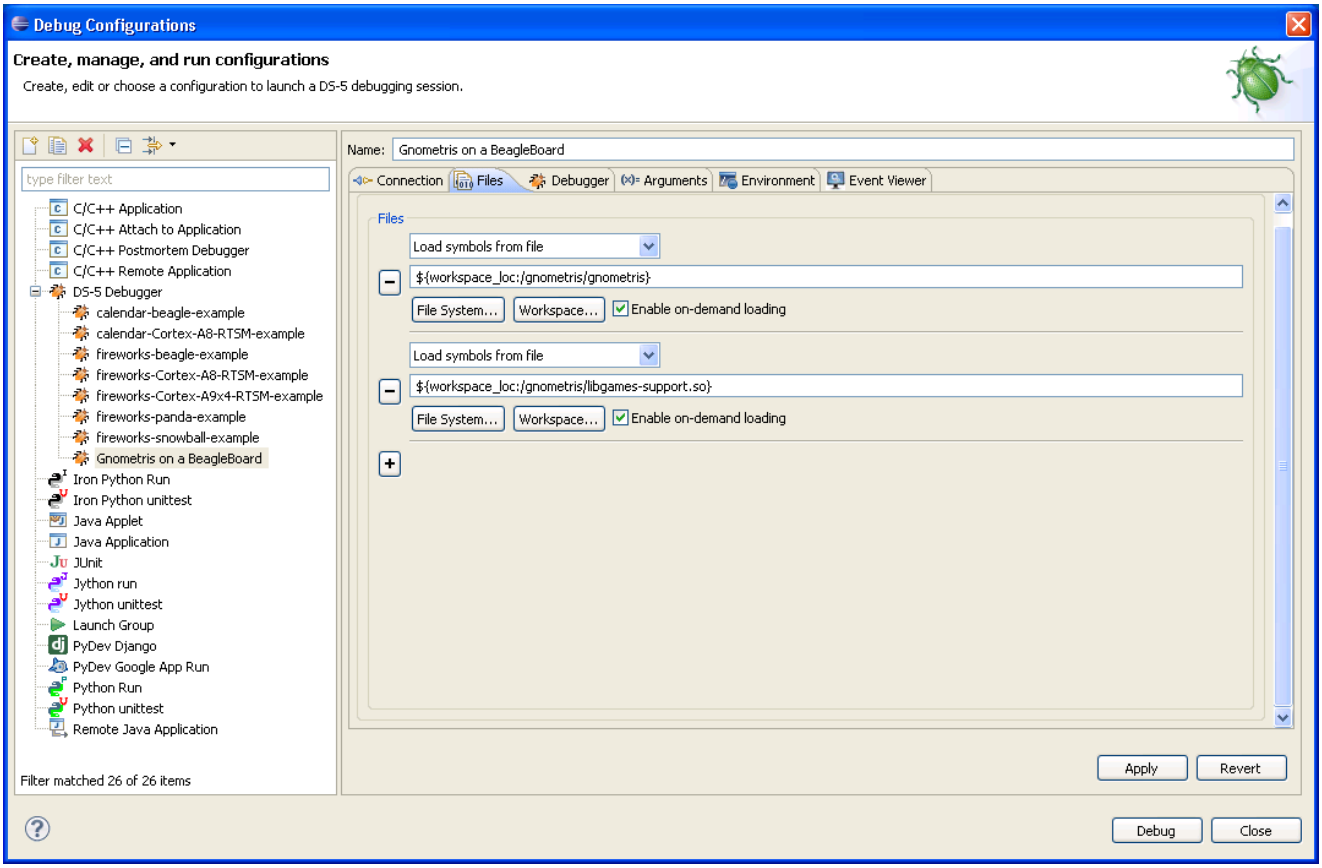


Figure 2-10 Typical file selection for a Beagle board

9. Click on the **Debugger** tab to see the debugging options for the configuration.
10. In the Run control panel:
    a) Select **Debug from symbol**.
    b) Enter **main** in the field provided.
11. In the Host working directory panel, select **Use default**.

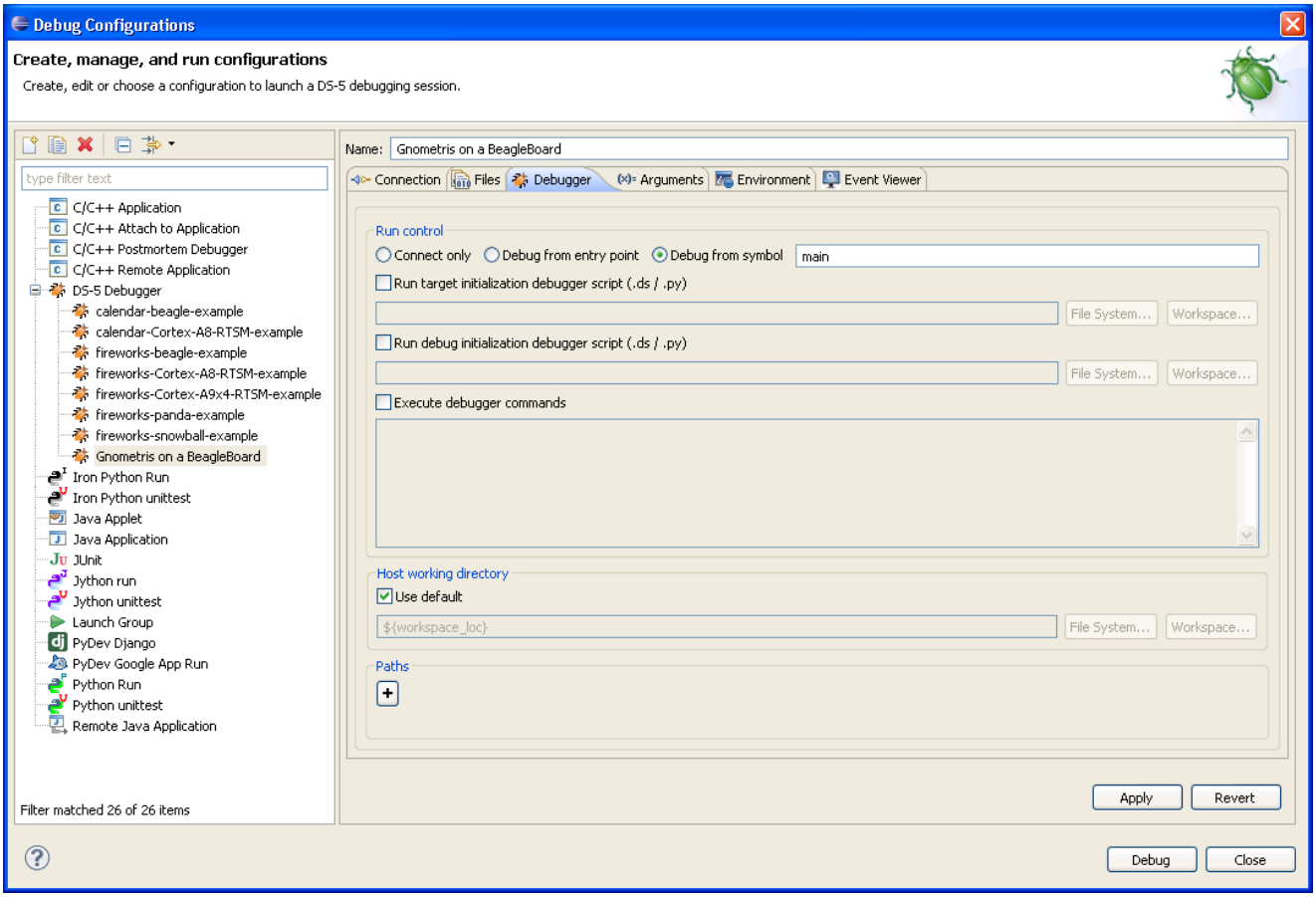


Figure 2-11 Typical debugger settings for a Beagle board

12. Click on **Debug** to start the debugger and run to the `main()` function.
13. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.
2.4 Building the Gnometris project from Eclipse on page 2-28.
2.5 Building the Gnometris project from the command-line on page 2-29.
2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.
2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.
2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.
2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Debug Configurations - Connection tab.
Debug Configurations - Files tab.
Debug Configurations - Debugger tab.
Debug Configurations - Environment tab.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.

2.4 Building the Gnometris project from Eclipse on page 2-28.
2.5 Building the Gnometris project from the command-line on page 2-29.
2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.
2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.
2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.
2.9 Debugging Gnometris on page 2-44.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Debug Configurations - Connection tab.
Debug Configurations - Files tab.
Debug Configurations - Debugger tab.
Debug Configurations - Environment tab.
Target management terminal for serial and SSH connections.
Remote Systems view.

2.9 Debugging Gnometris

Debugging the Gnometris application using the example project containing the image binaries and libraries provided with DS-5.

Procedure

1. Ensure that you are connected to the target, Gnometris is running, and the debugger is waiting at the `main()` function.
2. In the Project Explorer view, open the Gnometris directory to see a list of all the source files.
3. Double-click on the file `blockops-noclutter.cpp` to open the file.
4. In the `blockops-noclutter.c` file, find the line `BlockOps::rotateBlock()`, and double click in the vertical bar on the left-hand side of the C/C++editor to add a breakpoint. A marker is placed in the vertical bar of the editor and the Breakpoints view updates to display the new information.
5. Click on **Continue** in the Debug Control view to continue running the program.
6. Start a new Gnometris game on the target. When a block arrives, press the up cursor key to hit the breakpoint.
7. Select the Registers view to see the values of the registers.
8. Select the Disassembly view to see the disassembly instructions. You can also double click in the vertical bar on the left-hand side of this view to set breakpoints on individual instructions.
9. In the Debug Control view, click on **Step Over Source Line** to move to the next line in the sourcefile. All the views update as you step through the source code.
10. Select the History view to see a list of all the debugger commands generated during the current debug session. You can select one or more commands and then click on **Exports the selected lines as a script** to create a script file for future use.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.

2.4 Building the Gnometris project from Eclipse on page 2-28.

2.5 Building the Gnometris project from the command-line on page 2-29.

2.6 Loading the Gnometris application on a Fixed Virtual Platform (FVP) on page 2-30.

2.7 Loading the Gnometris application on to an ARM® Linux target on page 2-31.

2.8 Configuring an RSE connection to work with an ARM® Linux target on page 2-32.

2.8.2 Connecting to the Gnometris application that is already running on a ARM® Linux target on page 2-39.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

C/C++ editor.

Debug Control view.

Registers view.

2.10 Debugging a loadable kernel module

You can use DS-5 to develop and debug a loadable kernel module. Loadable modules can be dynamically inserted and removed from a running kernel during development without the need to frequently recompile the kernel.

This tutorial uses a simple character device driver `modex.c` which is part of the ARMv7 Linux application examples available in DS-5.

You can use `modex.c` to compile, run, and debug against your target. The `readme.html` in the `<DS-5 installation folder>/DS-5/examples/docs/kernel_module` contains information about customising this for your target.

——— **Note** ———

If you are working with your own module, before you can debug it, you must ensure that you:

- Unpack kernel source code and compile the kernel against exactly the same kernel version as your target.
- Compile the loadable module against exactly the same kernel version as your target.
- Ensure that you compile both images with debug information. The debugger requires run-time information from both images when debugging the module.

———————————

Procedure

1. Create a new **Debug Configuration**.
   a) From the main DS-5 menu, select **Run** > **Debug Configurations**.
   b) In the Debug Configurations dialog box, create a **New Launch Configuration** and give it a name. For example, `my_board`.
   c) In the **Connection** tab, select the target and platform and set up your target connection.

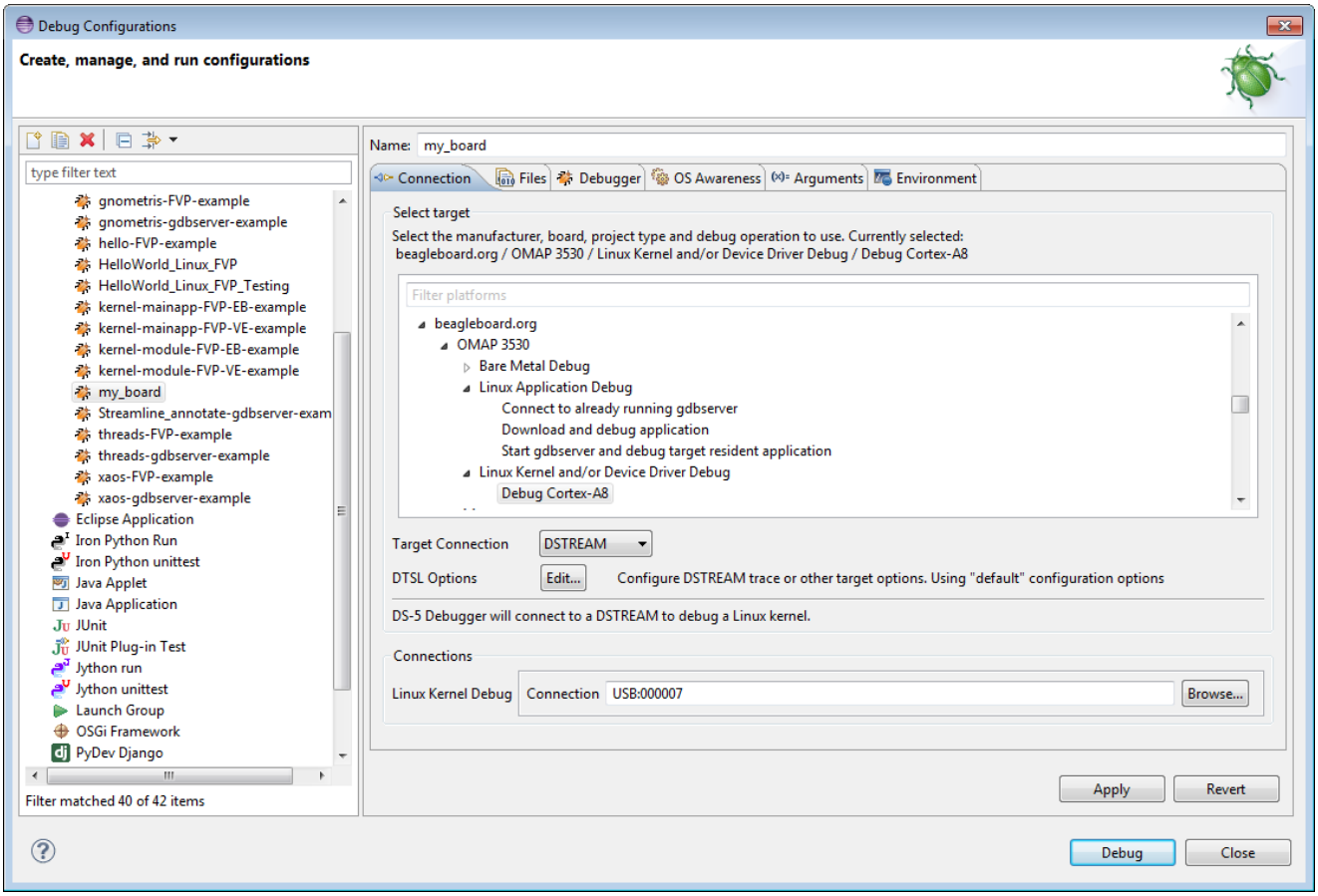


Figure 2-12 Typical connection settings for a Linux kernel/Device Driver Debug

d) In the **Files** tab, set up the debugger settings to load debug information for the Linux kernel and the module.

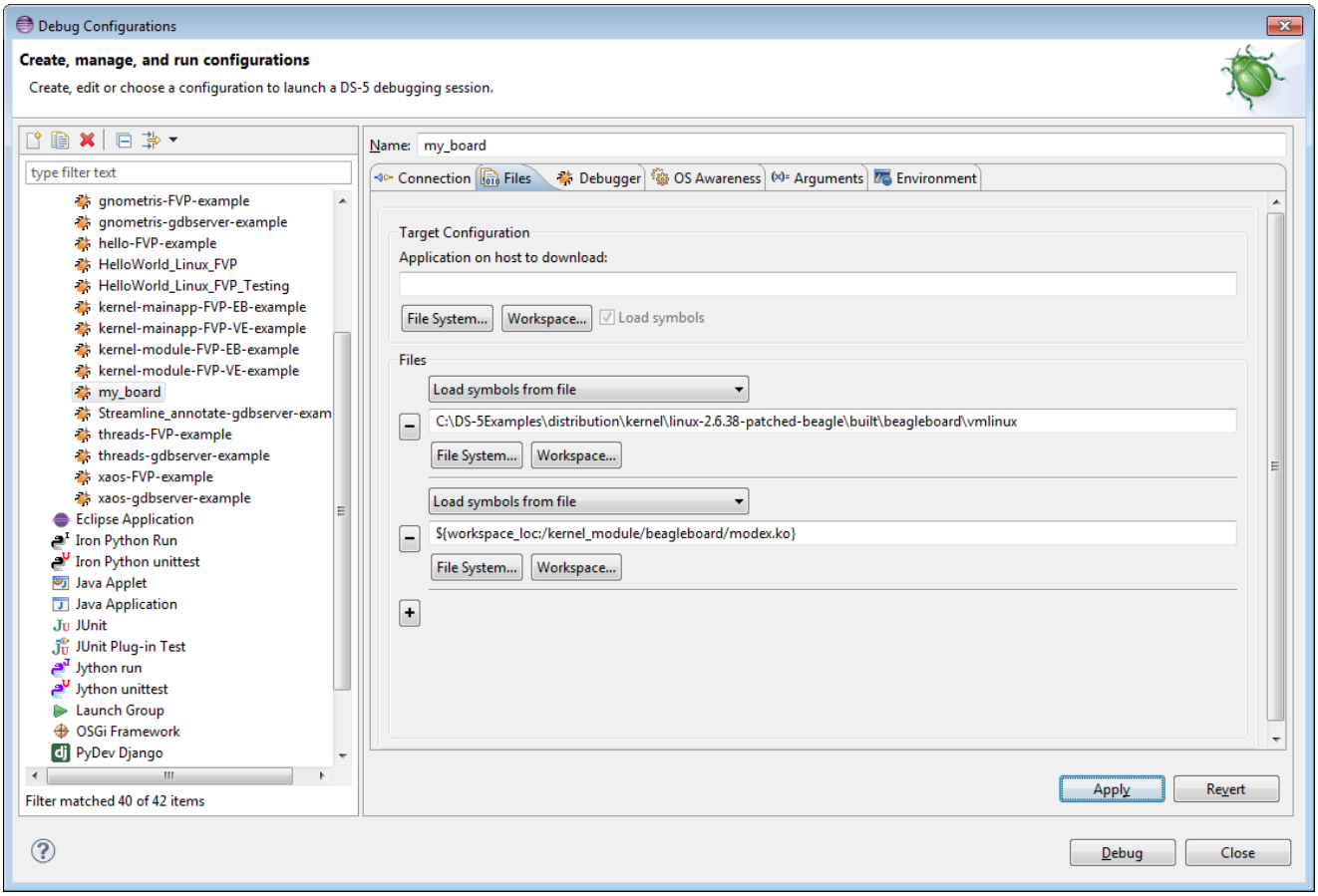


Figure 2-13 Typical Files settings for a Linux kernel/Device Driver Debug

e) In the **Debugger** tab, select **Connect only** in the **Run control** panel.

f) Click **Debug** to connect the debugger to the target.

2. Configure and connect a terminal shell to the target. You can use the *Remote System Explorer* (RSE) provided with DS-5.
3. Using RSE, copy the compiled module to the target:
   a) On the host workstation, navigate to `.../linux_system/kernel_module/stripped/modex.ko` file.
   b) Drag and drop the module to a writeable directory on the target.
4. Using the terminal shell, insert the `modex.ko` kernel module.
   a) Navigate to the location of the kernel module.
   b) Execute the following command: `insmod modex.ko`
   The **Modules** view updates to display details of the loaded module.
5. To debug the module, set breakpoints, run, and step as required.
6. To modify the module source code:
   a) Remove the module using commands as required in the terminal shell. For example: `rmmod modex`
   b) Recompile the module.
   c) Repeat steps *3* to *5* as required.

Note

When you insert and remove a module, the debugger stops the target and automatically resolves memory locations for debug information and existing breakpoints. This means that you do not have to stop the debugger and reconnect when you recompile the source code.

Useful terminal shell commands:

`lsmod`
: Displays information about all the loaded modules.

`insmod`
: Inserts a loadable module.

`rmmod`
: Removes a module.

Useful DS-5 Debugger commands:

`info os-modules`
: Displays a list of OS modules loaded after connection.

`info os-log`
: Displays the contents of the OS log buffer.

`info os-version`
: Displays the version of the OS.

`info processes`
: Displays a list of processes showing ID, current state and related stack frame information.

`set os-log-capture`
: Controls the capturing and printing of *Operating System* (OS) logging messages to the console.

OS modules loaded after connection are displayed in the Modules view.

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Configuring a connection to a Linux Kernel.
Controlling execution.
Examining the target.
About debugging a Linux kernel.
About debugging Linux kernel modules.
ARM Linux problems and solutions.
Target connection problems and solutions.
Operating System (OS) DS-5 debugger commands.
Target management terminal for serial and SSH connections.

2.11 Performance analysis of threads application running on ARM® Linux

ARM Streamline is a graphical performance analysis tool. It provides timeline and analysis reports that highlight problem areas at system, process, and thread level, in addition to hot spots in the applications.

Prerequisites

Before capturing the analysis data, ensure that:

1. You obtain the IP address of the target. You can use the `ifconfig` application in a Linux console. The IP address is denoted by the **inet addr**.
2. The ARM Linux Kernel is configured for Streamline.
3. The threads application is copied to the target.
4. The gator daemon is running on the target.

Procedure

1. Launch Eclipse.
2. Launch a terminal shell and connect it to the target. You can use the terminal shell provided with *Remote Systems Explorer* (RSE).
3. In the terminal shell, navigate to the directory where you copied the `threads` application.
4. Ensure that you are in the C/C++ Perspective.
5. Create a target connection:
   a) Select the **Change capture options...** toolbar icon in the Streamline Capture Data view.
   b) In the Name field, enter a suitable name for the new configuration.
   c) In the Connection panel, enter the IP address or name and the associated port number for the connection between the host workstation and the target.
   d) In the Capture panel, accept the default settings or customize as required.
   e) Click on **Add Program...** or **Add program from Workspace...** in the Program Images panel to open a dialog box where you can select the application image.
   f) Navigate to the threads application and click on **Open** or **OK** to close the dialog box.
   g) Click on **Apply** to save the settings.
   h) To start capturing the data, click on the **Start capture** toolbar icon in the Streamline Capture Data view.
6. In the terminal shell, execute the following command to run the threads application:

```
./threads
```

7. After you have completed running the threads application, return to the C/C++ Perspective in Eclipse.
8. Click on the report in the Streamline Capture Data view to analyze the graphical data.

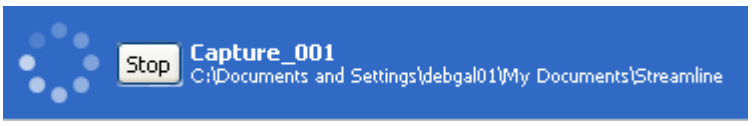


Figure 2-14 Streamline Capture Data file

A Streamline Analysis Data file is generated automatically when you stop capturing the data or you can double-click on an existing analysis file to view it in the Editor.

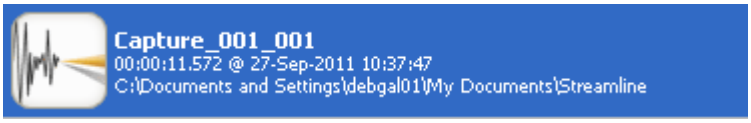


Figure 2-15 Streamline Analysis Data file

Related references

3.6 Examples provided with DS-5 on page 3-84.

Related information

Using ARM Streamline.

2.12 Setting up the Android tools for use with DS-5

This tutorial describes the steps required for Android development using DS-5, the Android *Native Development Kit* (NDK), Eclipse with *C/C++ Development Tooling* (CDT), and the Android *Software Development Kit* (SDK) Platform 2.2.

It does not describe how to install any of the Android tools. See the Android Developers website for more information.

——— **Note** ———

Use of the Android development kits are subject to their own terms and conditions.

———————————

Before you can debug an Android package containing native C/C++ code you must:

Procedure

1. Download and install the Android SDK tools and required platforms. This enables you to build Java applications together with any native C/C++ code into an Android package with a .apk file extension. This example uses the Android SDK Platform 2.2.
2. Download and install the Android NDK. This is a companion tool to the Android SDK that enables you to build performance-critical parts of your applications in native code such as C and C++ languages.

   ——— **Note** ———

   On Windows, you must also download and install `cygwin`, including the `make` package so that you can run the scripts inside the Android NDK.

   ———————————
3. Update the version of `gdbserver` in the Android NDK `...\toolchains\...\prebuilt` directory with the required version of `gdbserver` provided by DS-5. You can locate this file by selecting **Help** > **ARM Extras...** from the main menu. Ensure that you rename it to `gdbserver`.
4. Add the`...\android-sdk\platform-tools` folder to your `PATH` environment variable. If it is already configured then you can skip this step.
5. Set up the Eclipse plug-in for Android:
   a) Launch Eclipse.
   b) Install the DS-5 Eclipse plug-ins. If you have a full DS-5 installation then you can skip this step.
   c) Install the *Android Development Tools* (ADT) Eclipse plug-ins.
   d) Select **Window** > **Preferences** > **Android** and click on **Browse...** to set the location of the Android SDK.
   e) Open the Android SDK and AVD Manager dialog box by selecting **Window** > **Android SDK and AVD Manager**.
   f) Expand the **Available packages** group and add SDK platforms as required. For example, Android SDK Platform Android 2.2.

Related tasks

2.2 Installing DS-5 into a custom Eclipse environment on page 2-25.
2.13 Loading the hello-neon application on to an Android target on page 2-52.
2.14 Connecting to an application that is already running on an Android target on page 2-56.

Related information

DS-5 Knowledge Articles.
Eclipse.
Cygwin.
Android Developers.

2.13 Loading the hello-neon application on to an Android target

Describes how to load the hello-neon application on to an Android target.

Procedure

1. Launch Eclipse.
2. Start the target.
3. Create a new Android project:
   a) Select **File** > **New** > **Project...**
   b) Expand the Android group and select **Android Project**.
   c) Click **Next**.
   d) Enter a suitable project name. For example, `HelloNeon`.
   e) Select **Create project from your existing source**.
   f) Click **Browse...** to locate the `hello-neon` folder.
   g) Click **Next**.
   h) Select the required Build Target. For example, **Android 2.2**.
   i) Click **Finish**.
4. Ensure that the application builds with debug information. You can do this by:
   a) Open the `AndroidManifest.xml` file.
   b) Click on the **Application** tab.
   c) Select **true** in the Debuggable field.
   d) Save the changes and close the file.
5. Build the `hello-neon` source files with debug information using the scripts provided by the Android NDK. This tutorial uses the NDK revision 6b. For example:

   ```
   ./ndk-build -C samples/hello-neon NDK_TOOLCHAIN=arm-linux-androideabi-4.4.3 NDK_DEBUG=1
   ```

6. Select the Android project and build the `HelloNeon.apk`.
7. Create a new *Android Virtual Device* (AVD) and start the emulator.
8. Select **Debug Configurations...** from the **Run** menu.
9. Select **DS-5 Debugger** from the configuration tree and then click on **New** to create a new configuration. Alternatively you can select an existing DS-5 Debugger configuration and then click on **Duplicate** from the toolbar.
10. In the Name field, enter a suitable name for the new configuration. For example, `HelloNeon`.
11. Click on the **Connection** tab to see the target and connection options.
12. In the Select target panel:
    a) Select **Android - Generic platform**.
    b) Select **Download and debug an Android application** for the debug operation.
    c) Select the emulator in the Connections field.

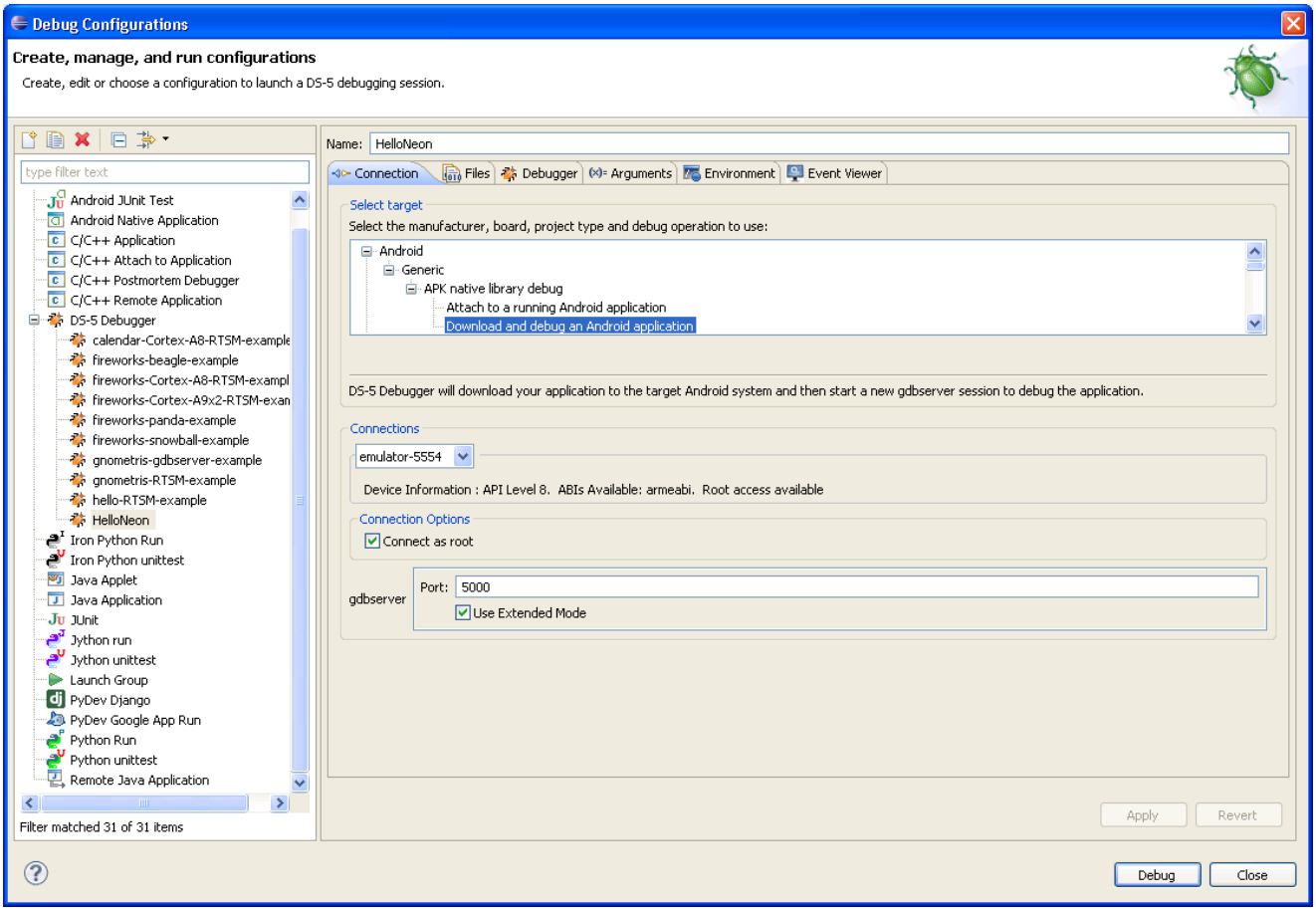


Figure 2-16 Typical Connection tab settings for an Android application

13. In the Files tab, click on **Workspace...** for the Project directory field and select the `hello-neon` directory. This automatically populates the other fields.

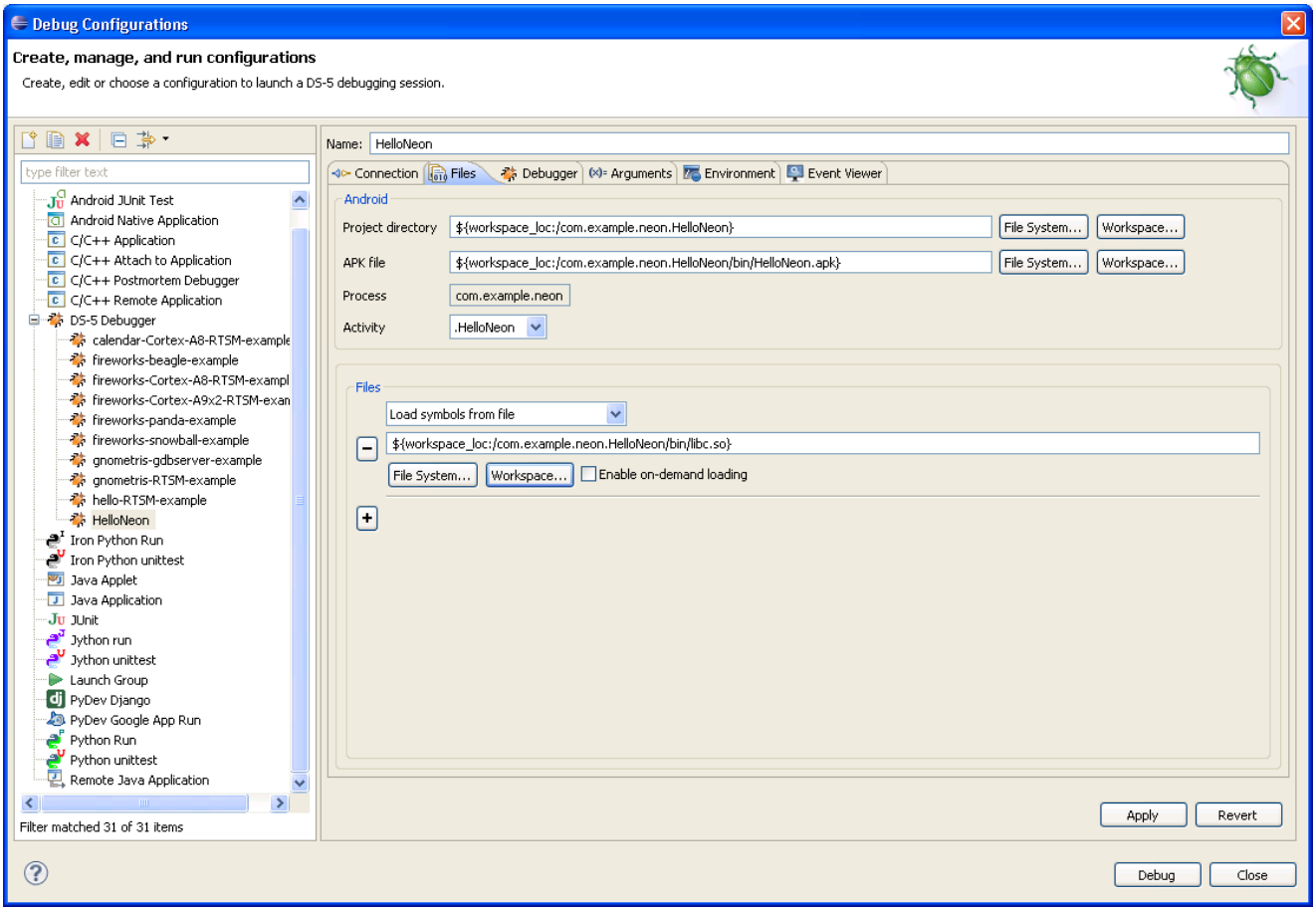


Figure 2-17 Typical Files tab settings for an Android application

14. Click on the **Debugger** tab and select **Connect only** in the Run control panel.
15. In the Paths panel, click on **Workspace...** and select the `hello-neon` directory.

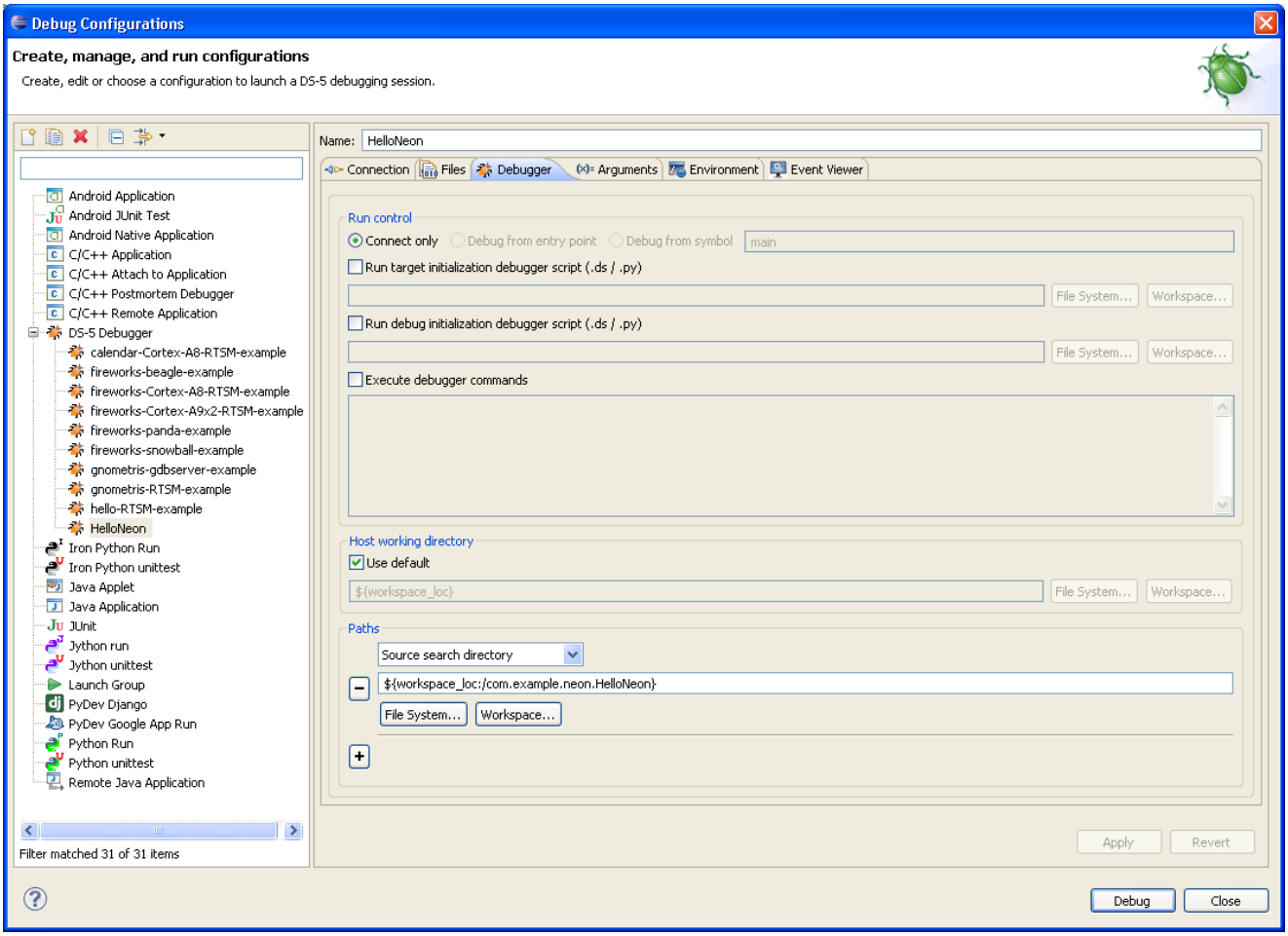


Figure 2-18 Typical Debugger tab settings for an Android application

16. Click on **Debug** to connect to the target.
17. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.
18. To debug the application, set breakpoints, run, and step as required.

Note

If the application exits before NDK can attach `gdbserver` to the native library then you might need to add a delay before launching the Java native libraries.

Related tasks

2.12 Setting up the Android tools for use with DS-5 on page 2-51.

2.14 Connecting to an application that is already running on an Android target on page 2-56.

Related information

DS-5 Knowledge Articles.

Eclipse.

Cygwin.

Android Developers.

2.14 Connecting to an application that is already running on an Android target

This tutorial describes how to connect to and debug an application that is already running on an Android target.

——— **Note** ———

Use of the Android development kits are subject to their own terms and conditions.

———————————

Procedure

1. Launch Eclipse.
2. Connect the host workstation to the target.
3. Ensure that the application is already installed and running on the target.
4. Select **Debug Configurations...** from the **Run** menu.
5. Select **DS-5 Debugger** from the configuration tree and then click on **New** to create a new configuration. Alternatively you can select an existing DS-5 Debugger configuration and then click on **Duplicate** from the toolbar.
6. In the Name field, enter a suitable name for the new configuration. For example, `HelloNeon`.
7. Click on the **Connection** tab to see the target and connection options.
8. In the Select target panel:
   a) Select **Android - Generic platform**.
   b) Select **Attach to a running Android application** for the debug operation.
   c) Select the target in the Connections field.

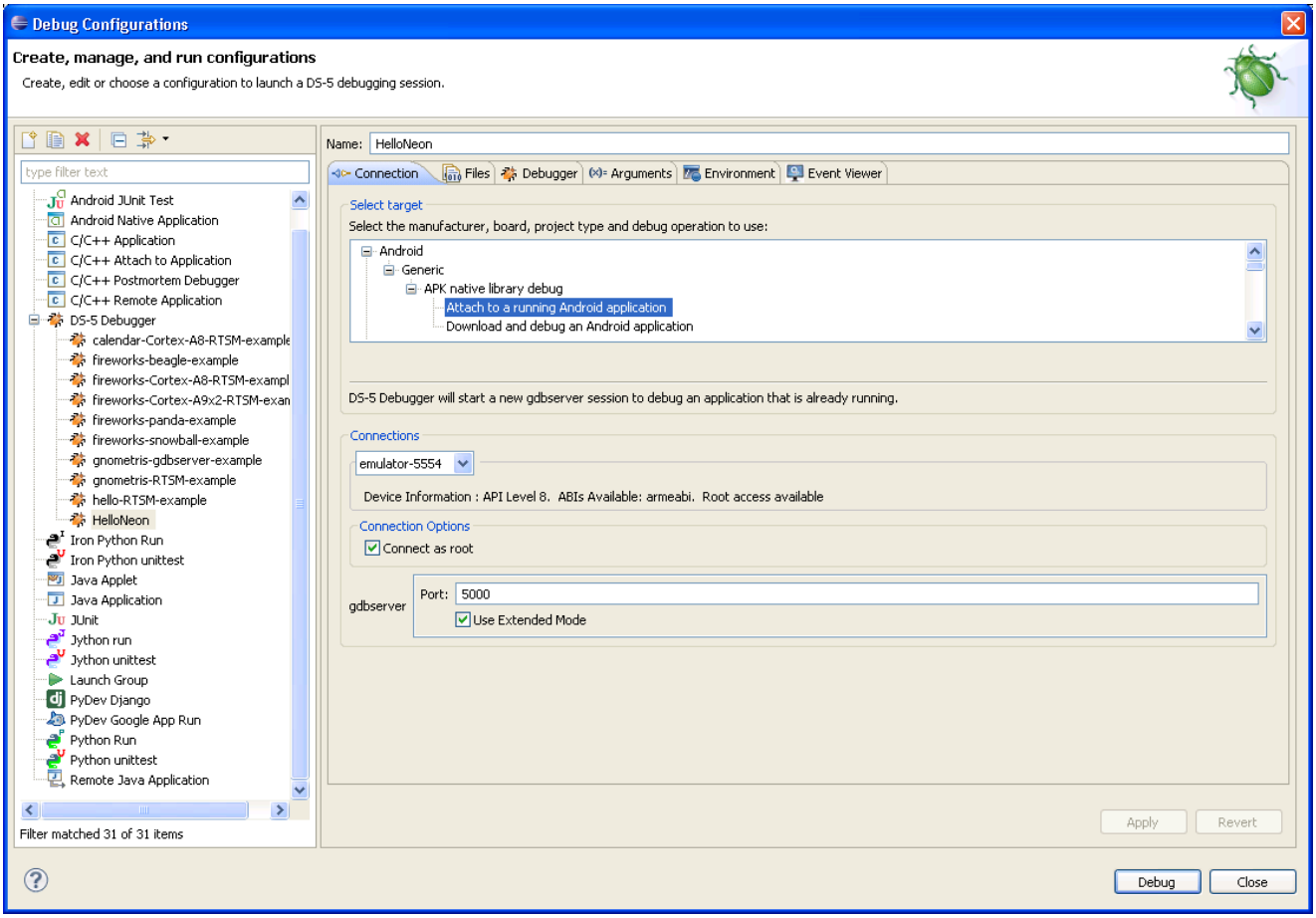


Figure 2-19 Typical Connection tab settings for an Android application

9. In the Files tab, click on **Workspace...** for the Project directory field and select the `hello-neon` directory. This automatically populates the other fields.

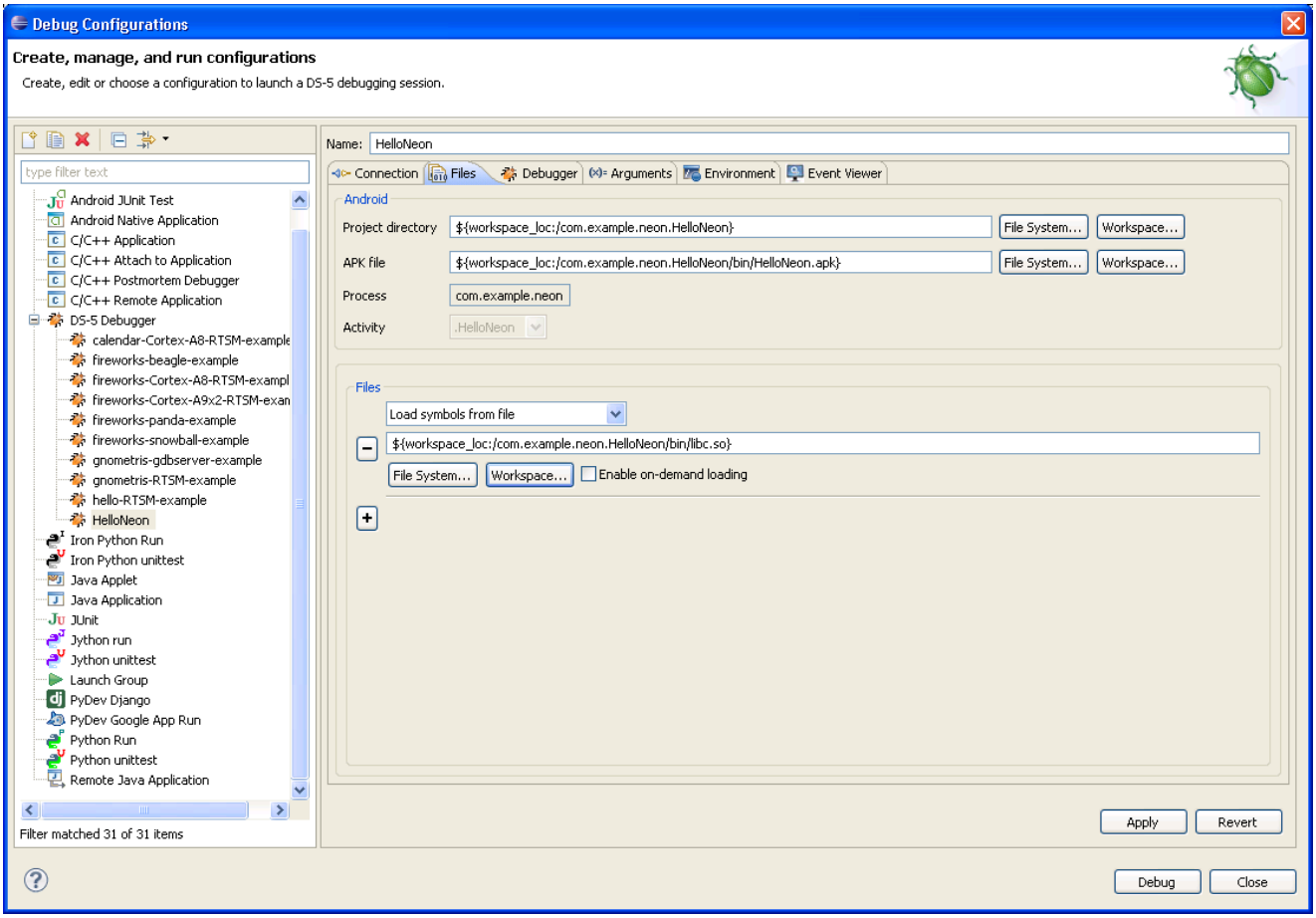


Figure 2-20 Typical Files tab settings for an Android application

10. Click on the **Debugger** tab and select **Connect only** in the Run control panel.
11. In the Paths panel, click on **Workspace...** and select the `hello-neon` directory.

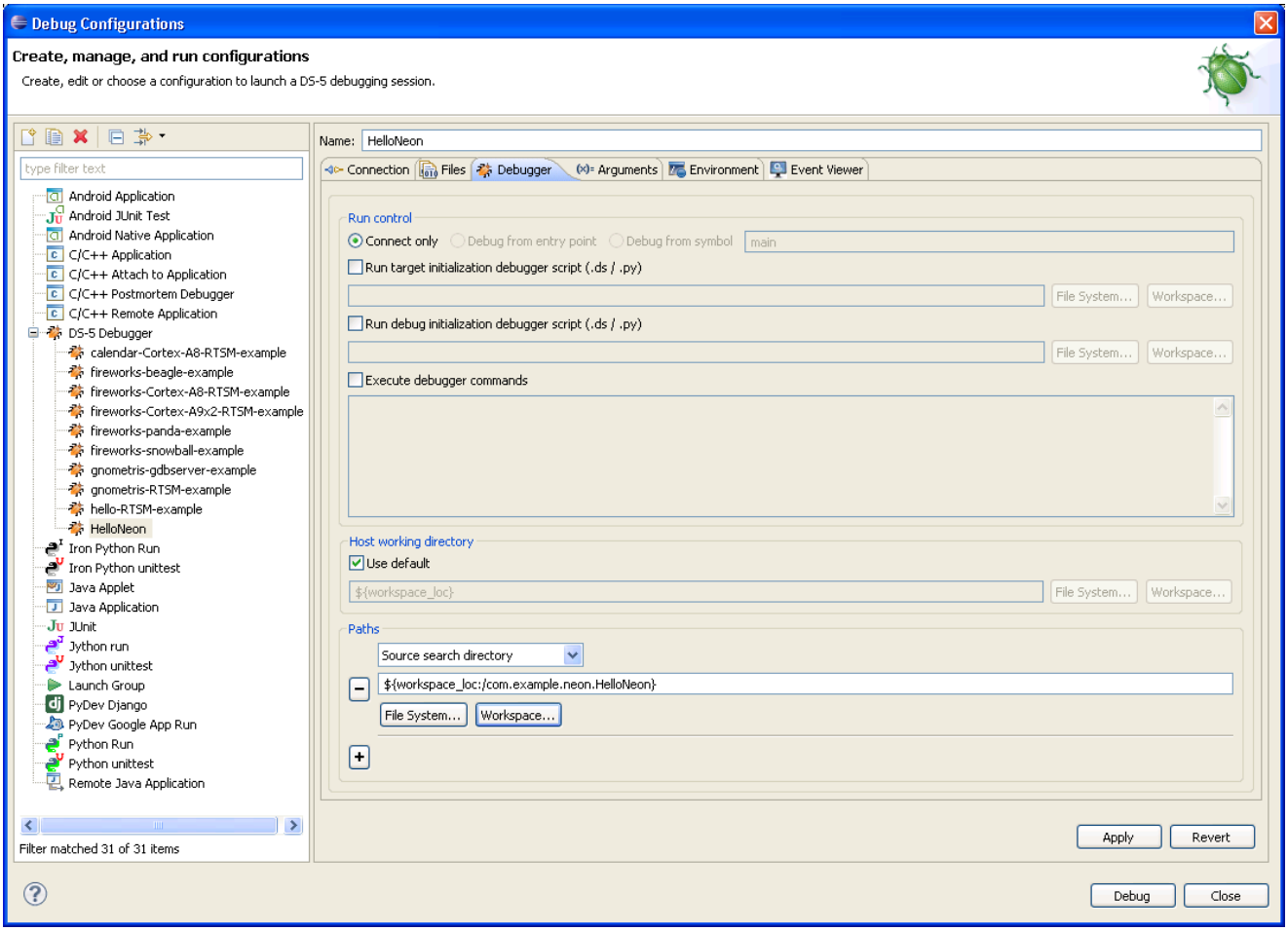


Figure 2-21 Typical Debugger tab settings for an Android application

12. Click on **Debug** to connect to the target.
13. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.
14. To debug the application, set breakpoints, run, and step as required.

Note

If the application exits before NDK can attach `gdbserver` to the native library then you might need to add a delay before launching the Java native libraries.

Related tasks

2.12 Setting up the Android tools for use with DS-5 on page 2-51.
2.13 Loading the hello-neon application on to an Android target on page 2-52.

Related information

DS-5 Knowledge Articles.
Eclipse.
Cygwin.
Android Developers.

2.15 Managing DS-5 licenses

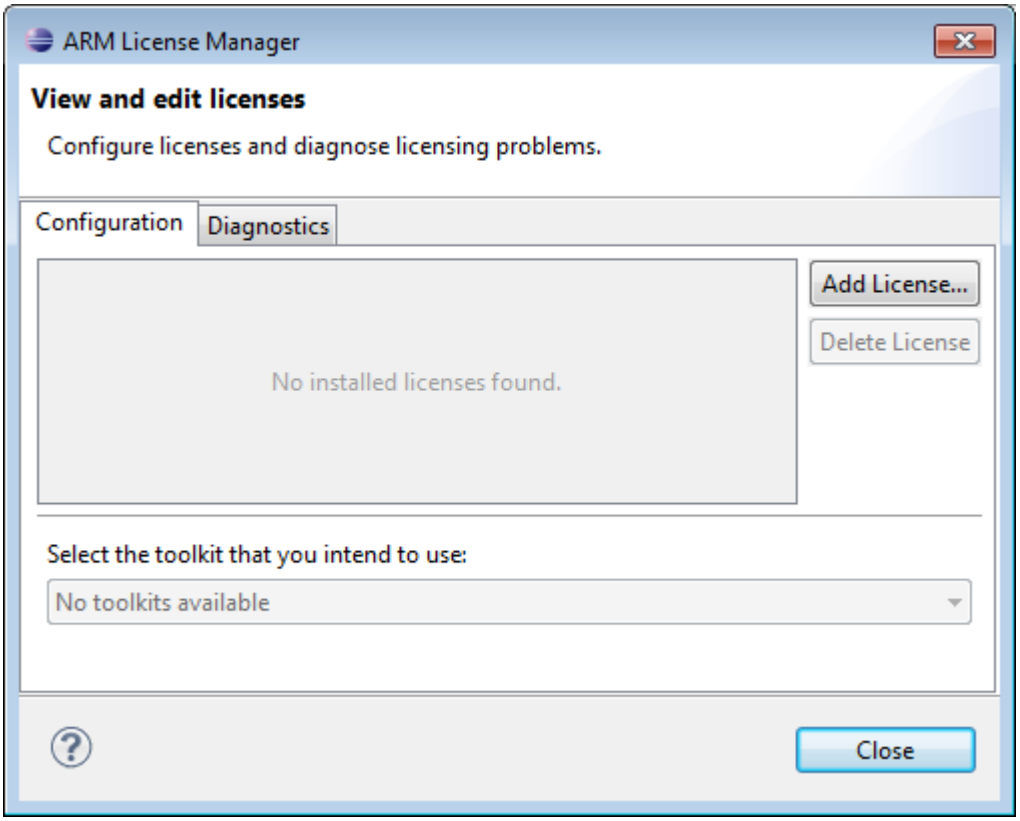
Describes how to manage DS-5 licenses using the ARM Licence Manager within the Eclipse environment.
This section contains the following subsections:

2.15.1 Viewing and editing licenses using the ARM License Manager

You can view and edit DS-5 licenses using the **ARM License Manager**.

Procedure

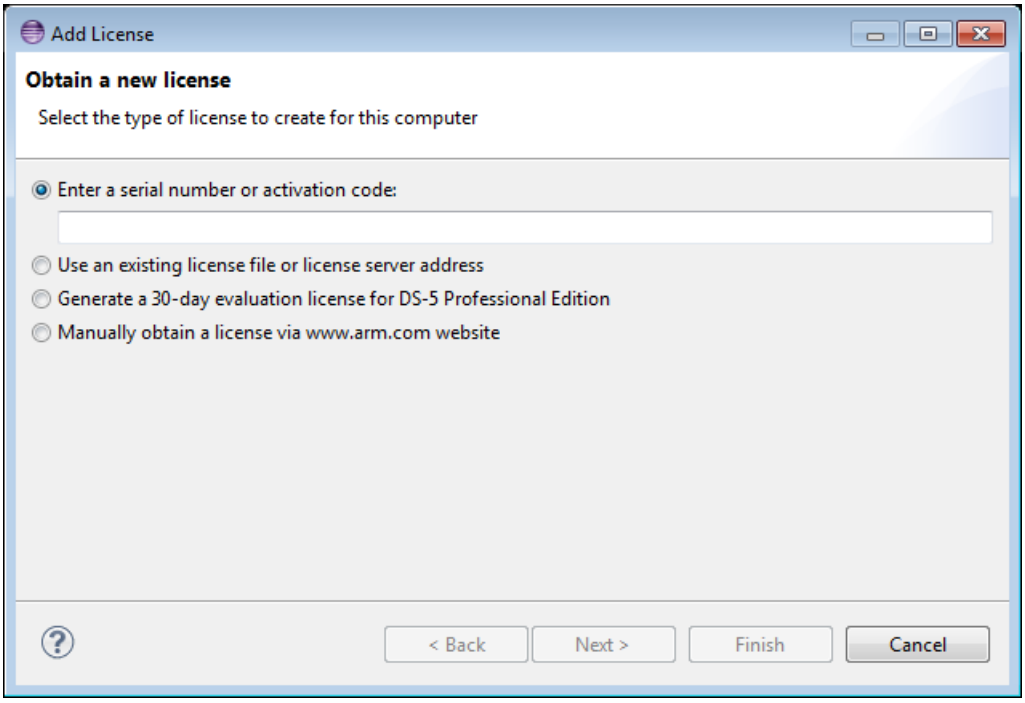
1. To view the **ARM License Manager**, in Eclipse, select **Help** > **ARM License Manager...**.



Note

Installed licenses are displayed in the **Configuration** tab of the ARM License Manager dialog box.

2. To add a license to DS-5, click **Add License...**. Use the options in the Add License dialog box to obtain a new license.

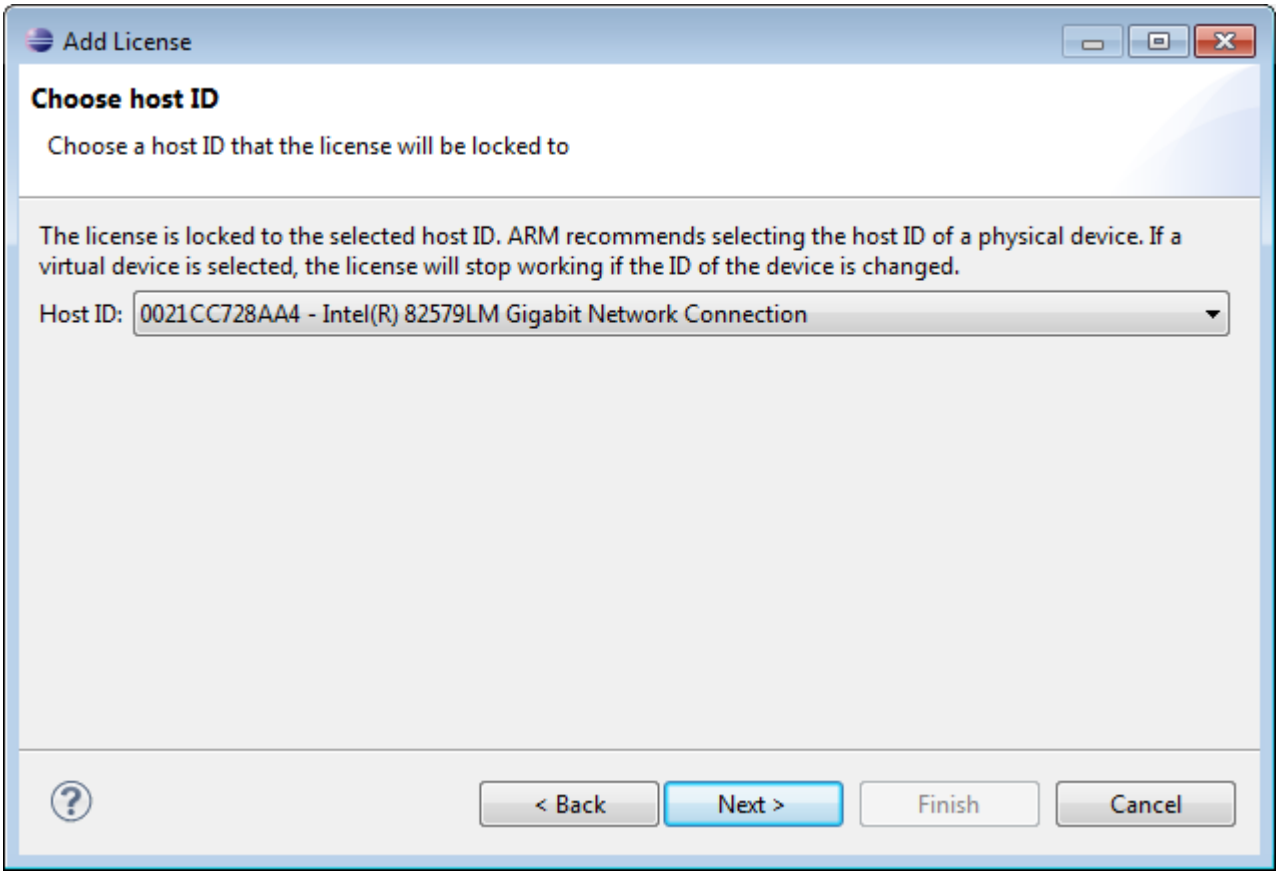


2.15.2 Using a serial number or activation code to obtain a license

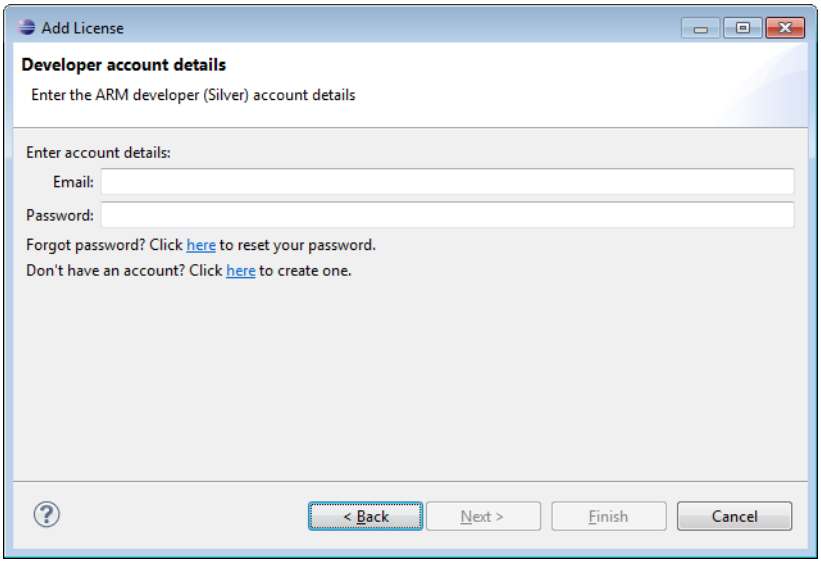
You can use a serial number or activation code to obtain a license.

Procedure

1. In the Add License dialog box, select **Enter a serial number or activation code to obtain a license**.
2. Enter the Serial number in the field.
3. Click **Next**.
4. In the Choose host ID dialog box, select a **Host ID** from the drop-down list.



5. Click **Next**.
6. Enter your ARM developer account details in the ARM Self-Service Portal or if you do not have an account, you can create one.



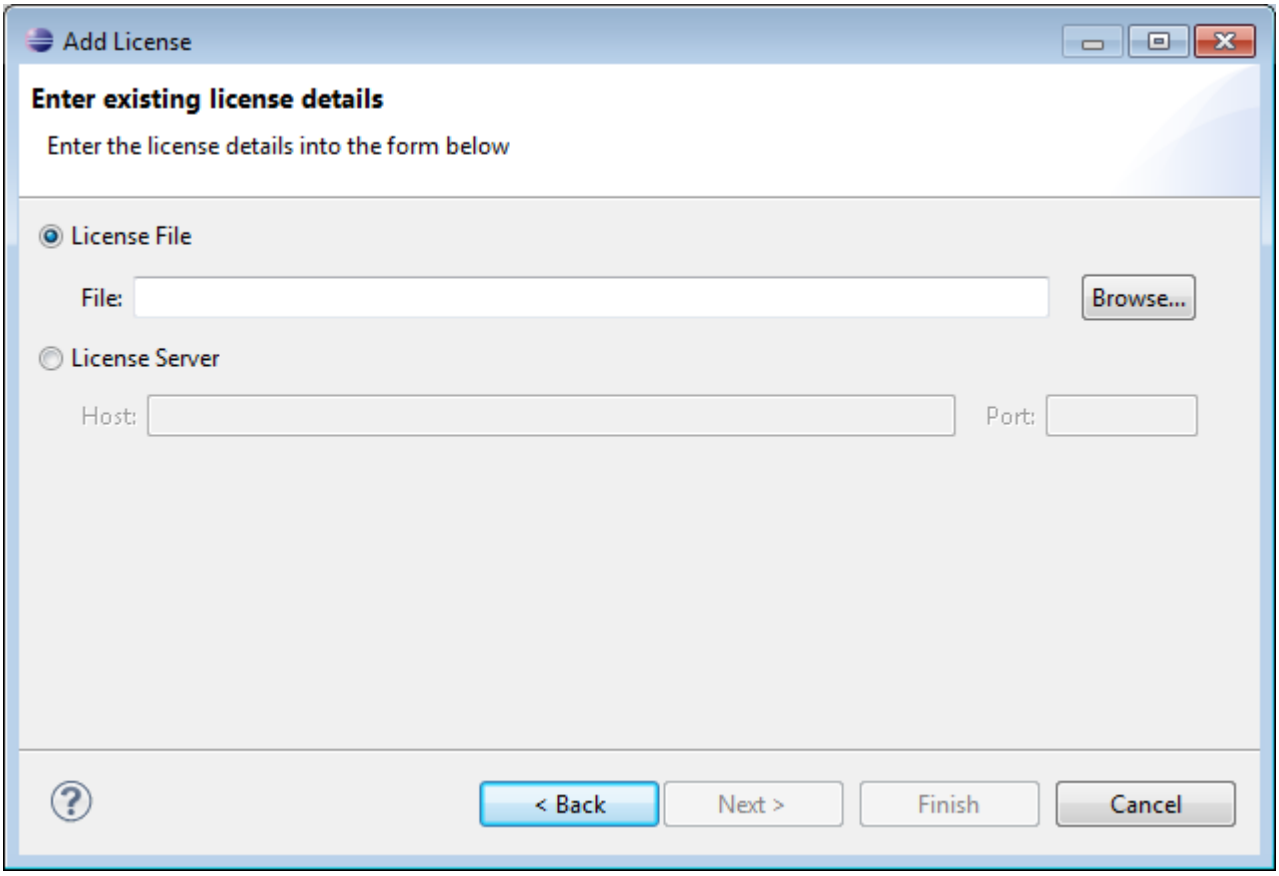
7. Click **Finish**.

2.15.3 Using an existing license file or license server to obtain a license

You can obtain a license using an existing license file or license server.

Procedure

1. In the Add License dialog box, select **Use an existing license file or license server address**.
2. Click **Next**.
3. In the Enter existing license details dialog, if you have a license file, select **License File** or if you have a server to administer the license, select **License Server**.



Note

For server licenses, instead of entering the host and port information separately in their respective fields, you can enter them in the format `port@host` in the Host field.

4. Click **Finish** to add the license to the ARM License Manager.
   In Windows, license files are copied into the `%APPDATA%\ARM\DS-5\licenses` folder. In Linux, the license files are copied into the `$HOME/.ds-5/licenses` folder.

2.15.4 Evaluating DS-5 Professional

To evaluate DS-5, you can generate a license that allows you evaluate DS-5 Professional for 30 days.

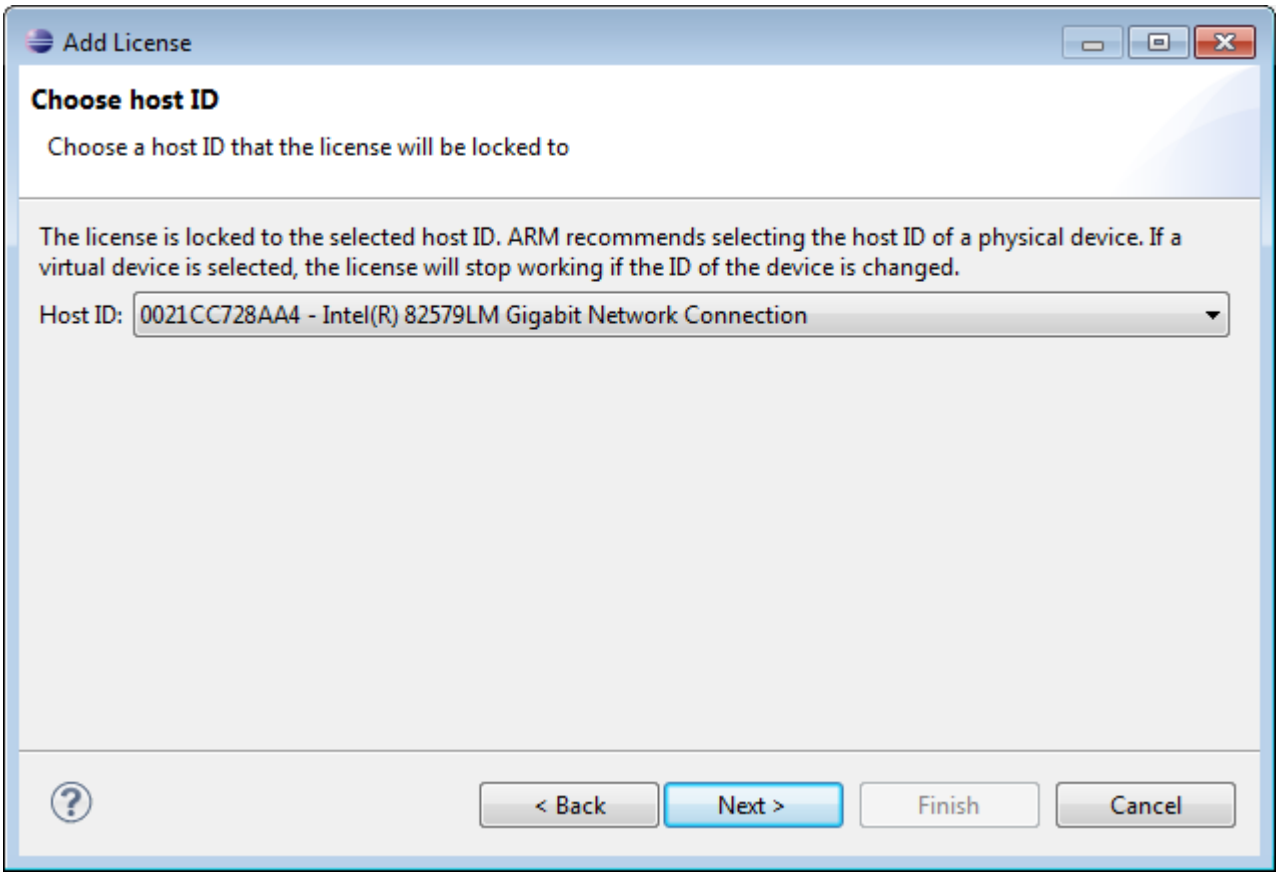
Procedure

1. In the Add License dialog box, select **Generate a 30-day evaluation license for DS-5 Professional Edition**.

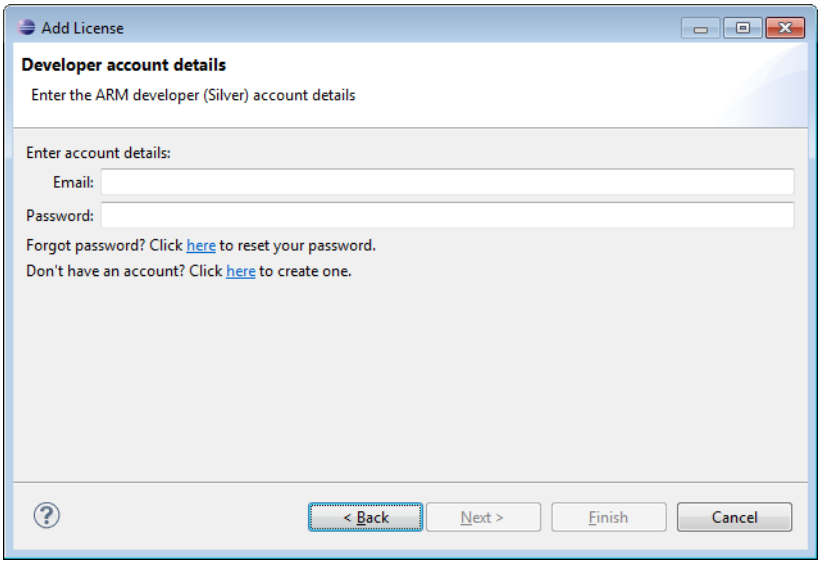
Note

Evaluation licenses are restricted to one 30-day evaluation license per machine. Contact your support team for extending your license.

2. Click **Next**.
3. In the Choose host ID dialog box, select a **Host ID** from the drop-down list.



4. Click **Next**.
5. Enter your ARM developer account details in the ARM Self-Service Portal or if you do not have an account, you can create one.



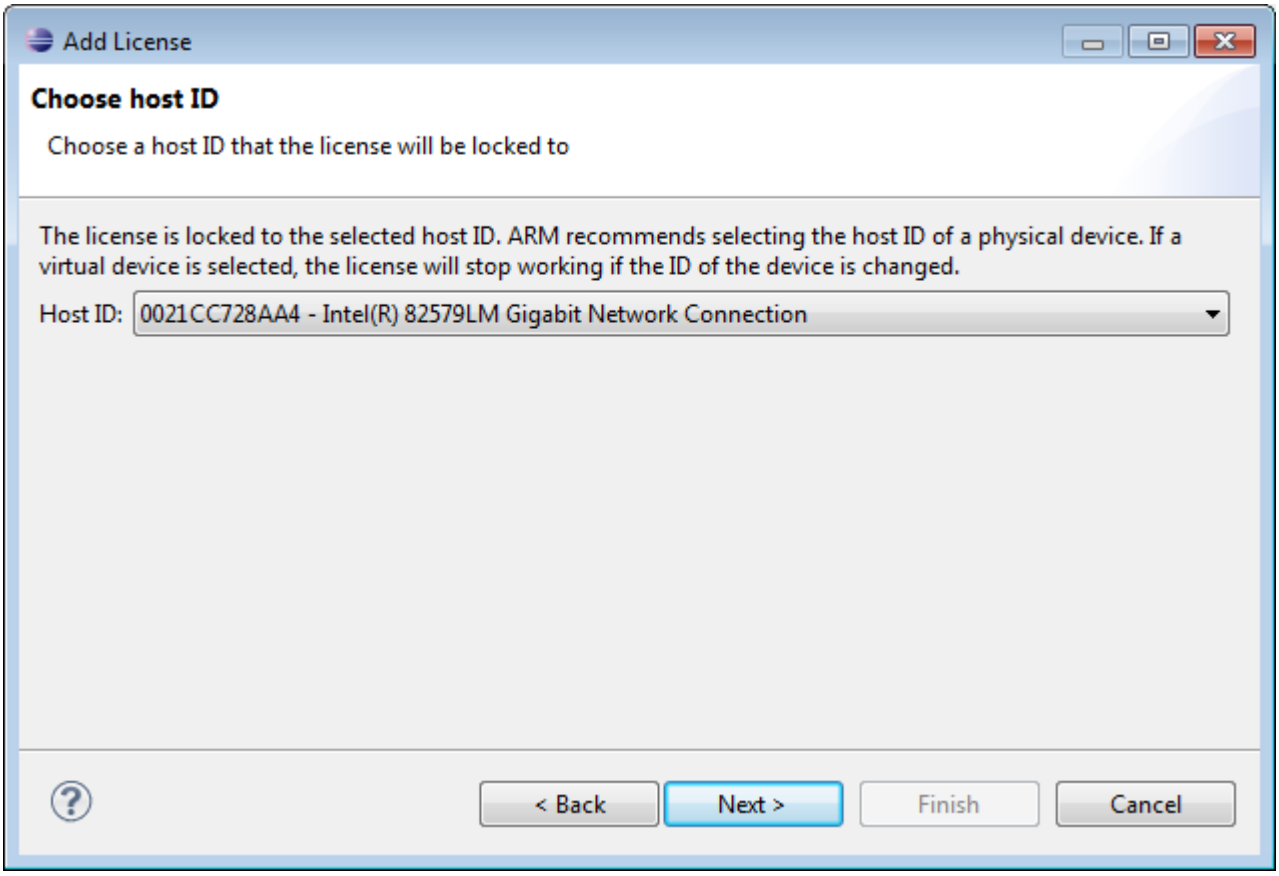
6. Click **Finish**.

2.15.5 Obtaining a license manually via the ARM website

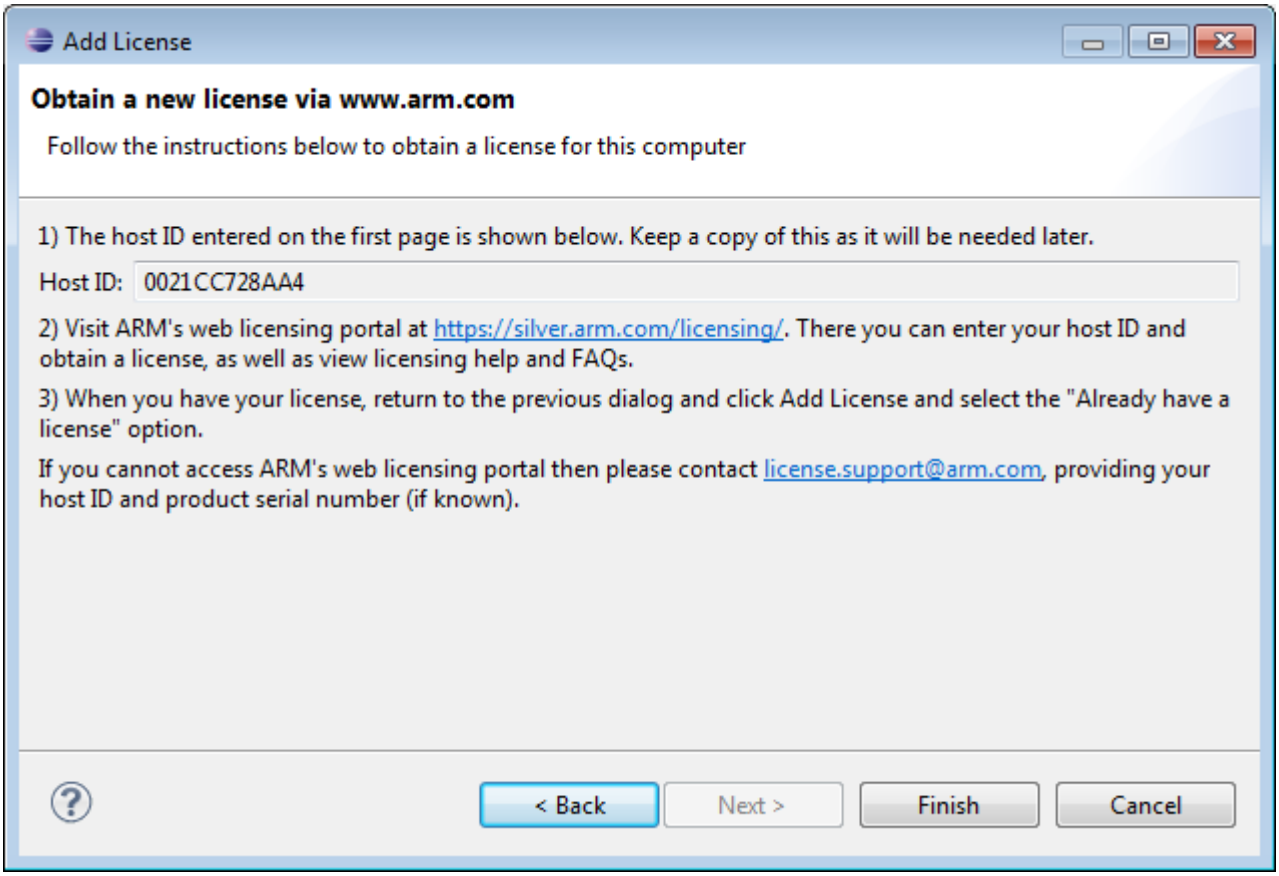
You can manually obtain a license from the ARM website.

Procedure

1. In the Add License dialog box, select **Manually obtain a license via www.arm.com website**.
2. Click **Next**.
3. In the Choose host ID dialog box, select a **Host ID** from the drop-down list.



4. Click **Next**
5. Follow steps 1 to 3 in the Obtain a new license via *http://www.arm.com* dialog box.



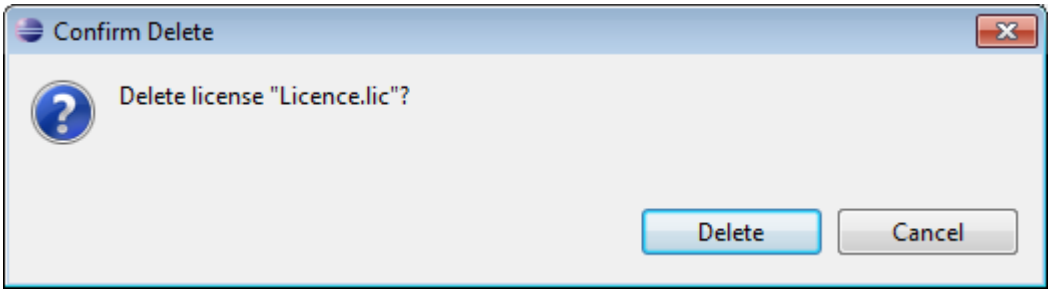
6. Click **Finish**.

2.15.6 Deleting a license

You can use the **Delete** option to delete a license.

Procedure

1. To view the **ARM License Manager**, in Eclipse, select **Help** > **ARM License Manager...**.
2. In the **Configuration** tab of the ARM License Manager dialog box, select the license to be deleted.
3. Click **Delete License**.
4. In the Confirm Delete dialog box, click **Delete** to uninstall and remove the license file from the DS-5 license folder.



2.15.7 Viewing detailed license and system information

You can view system and DS-5 license information using the **Diagnostics** tab available in the ARM License Manager dialog box. Use this information to investigate licensing issues or to provide additional information to your support team.

Procedure

1. To view the **ARM License Manager**, in Eclipse, select **Help** > **ARM License Manager...**.
2. Select the **Diagnostics** tab to view system and license information.
3. Click **Copy to Clipboard** to copy the information to the clipboard and send to your support team.
4. Click **Close** to close the dialog box.

Related tasks

2.16 Changing the Toolkit on page 2-69.

Related references

3.4 Licensing and product updates on page 3-82.

Related information

ARM DS-5 License Management Guide.

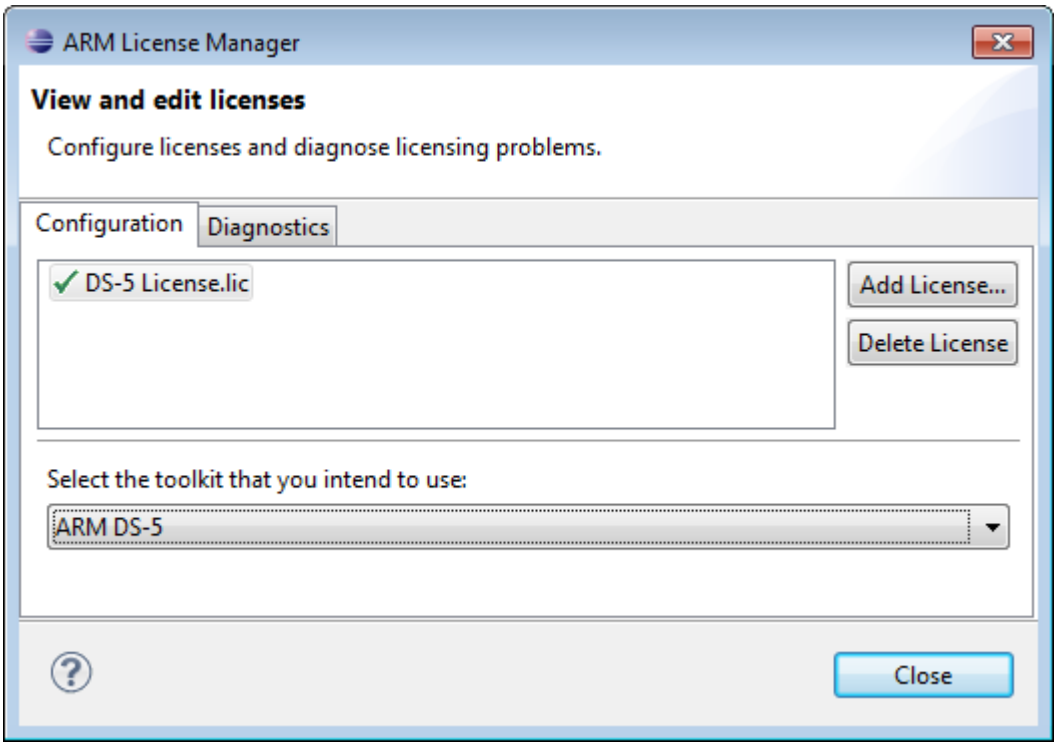
ARM Self-Service Portal.

2.16 Changing the Toolkit

You can change the toolkit for DS-5 using the ARM License Manager.

Procedure

1. Start Eclipse for DS-5.
2. Select **Help** > **ARM License Manager...**.
3. Click **Add License...** and follow the steps to add a license.
4. To change the toolkit, select it from the **Toolkit** drop-down menu.



Note

Some toolkit features are dependent on the installed license.

5. Click **Close** to close the dialog box.
6. Restart Eclipse.

Related references

2.15 Managing DS-5 licenses on page 2-60.
3.4 Licensing and product updates on page 3-82.

Related information

ARM DS-5 License Management Guide.
ARM Self-Service Portal.

2.17 Registering a compiler toolchain from the DS-5 command prompt

Use the `add_toolchain` utility from the DS-5 command prompt to register a new toolchain.

To register a toolchain using the DS-5 command prompt:

Procedure

1. Enter `add_toolchain` *`path`*, where *`path`* is the directory containing the toolchain binaries. The utility automatically detects the toolchain properties.

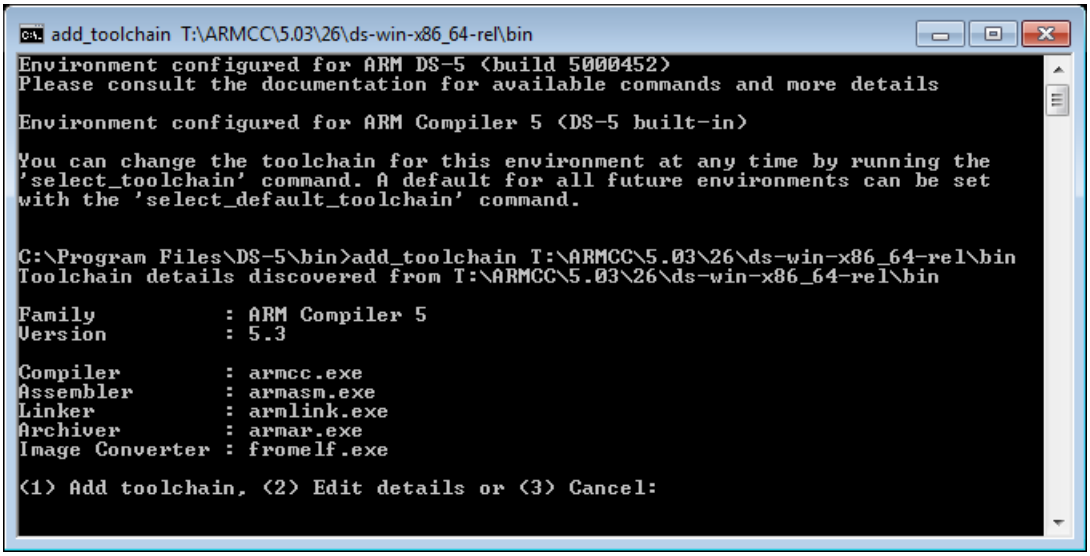


Figure 2-22 Registering a new toolchain

2. The utility prompts whether you want to register the toolchain with the details it has detected. If you want to change the details, the utility prompts for the details of the toolchain.

Note

- The toolchain type must be one of ARM Compiler 4, ARM Compiler 5, ARM Compiler 6, or GCC.
- The toolchain target only applies to GCC toolchains. It indicates what target platform the GCC toolchain builds for. For example, if your compiler toolchain binary is named `arm-linux-gnueabihf-gcc`, then the target name is the prefix `arm-linux-gnueabihf`. The target field allows DS-5 to distinguish different toolchains that otherwise have the same version.

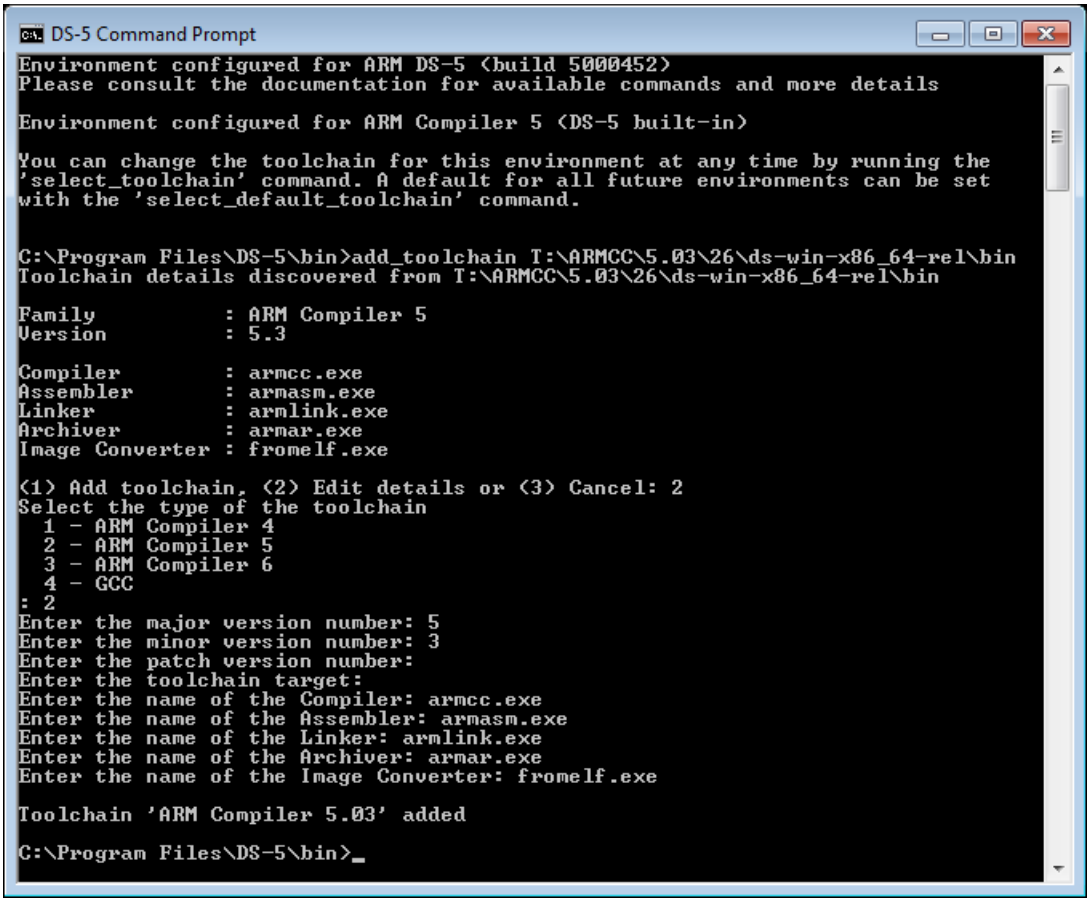


Figure 2-23 Registering a new toolchain

Note

You must manually enter the toolchain properties if:

- The toolchain properties were not autodetected.
- The type, major version, and minor version of the new toolchain are identical to a toolchain that DS-5 already knows about.

3. After you register a new toolchain, you must restart DS-5 before you can use the toolchain in the DS-5 environment.
4. When you create a new project, DS-5 shows the new toolchain in the available list of toolchains. In this example, `ARMCCv5.01` is the newly registered toolchain.

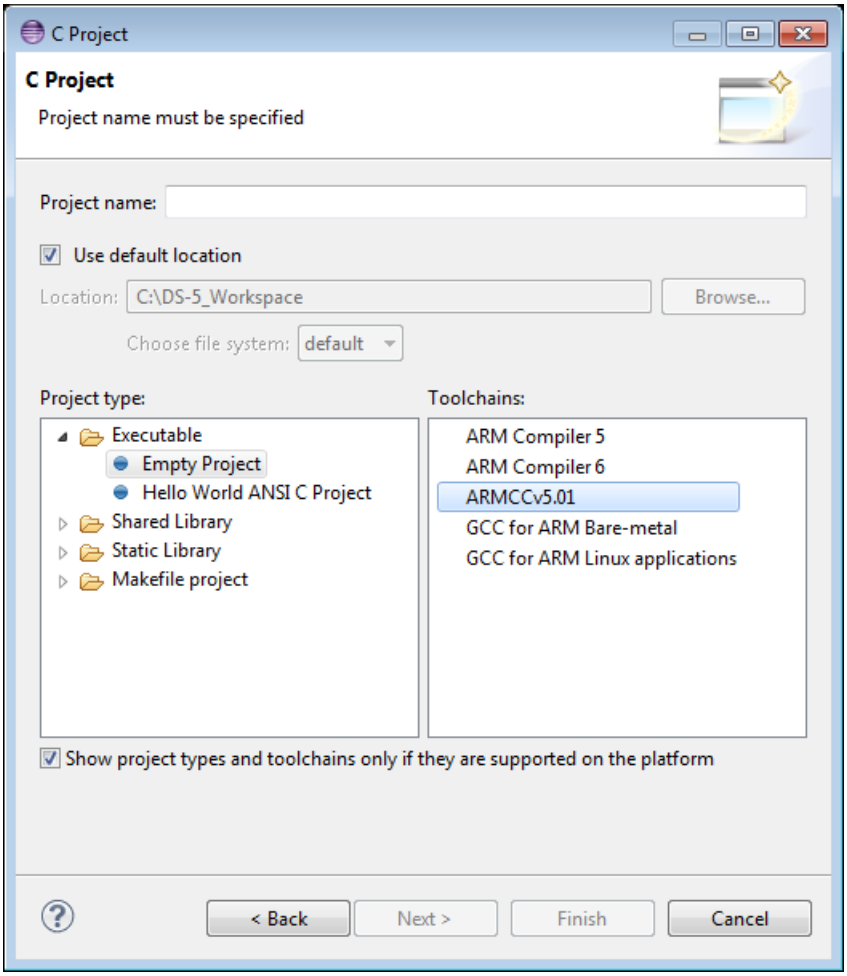


Figure 2-24 Using a new toolchain for a new project

For an existing project, if you want to change the toolchain to the newly registered toolchain, use the **Tool Chain Editor** dialog.

- Right-click the project and select **Properties** to show the **Properties** dialog.
- Select **C/C++ Build** > **Tool Chain Editor**

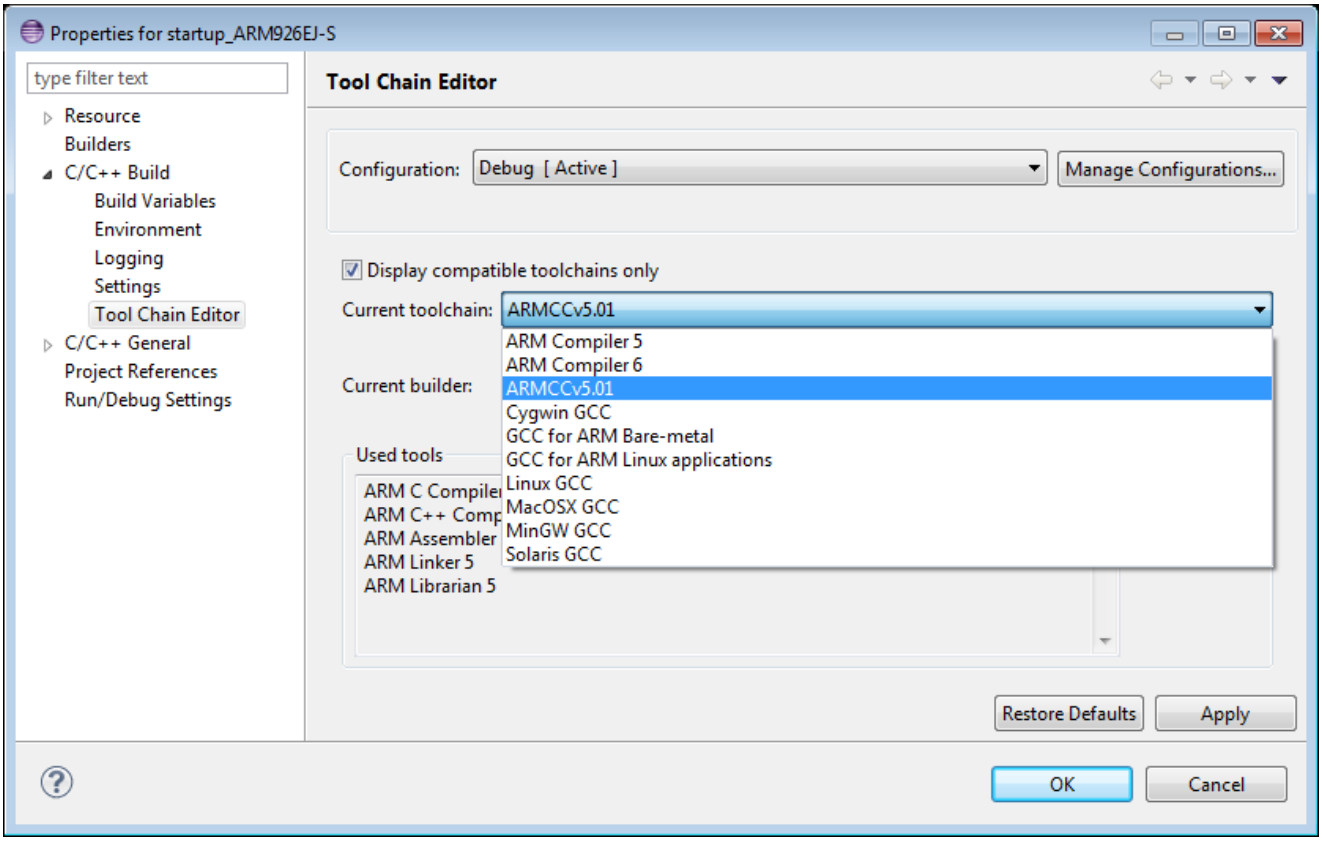


Figure 2-25 Changing the toolchain for a project

2.18 Registering a compiler toolchain from Eclipse

You can register compiler toolchains using the **Preferences** dialog in Eclipse for DS-5.

Procedure

1. To view the compiler toolchains that DS-5 currently knows about, select **Windows** > **Preferences**. And then select **DS-5** > **Toolchains**.

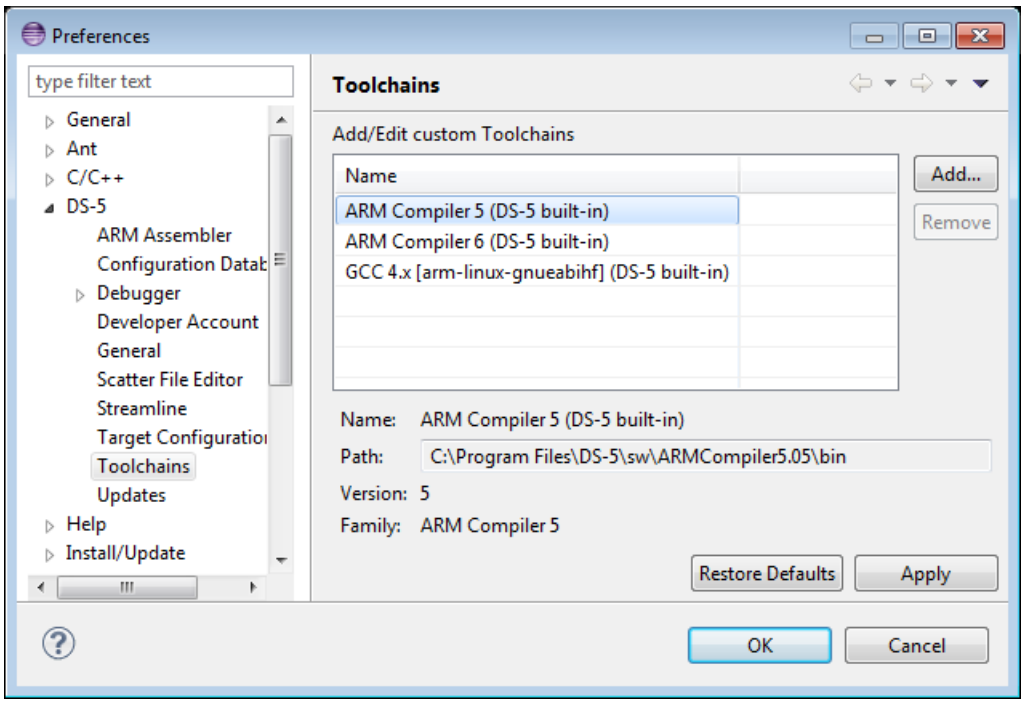


Figure 2-26 Toolchains preferences dialog

2. To add a toolchain, select **Add**. This displays the **Add a new Toolchain** dialog.
3. Enter the path to the toolchain binaries that you want to use. Then click **Next** to autodetect the toolchain properties.
4. When the toolchain properties have been autodetected, you can select **Finish** to register the toolchain. Alternatively, select **Next** to manually enter or change the toolchain properties, and then select **Finish**.

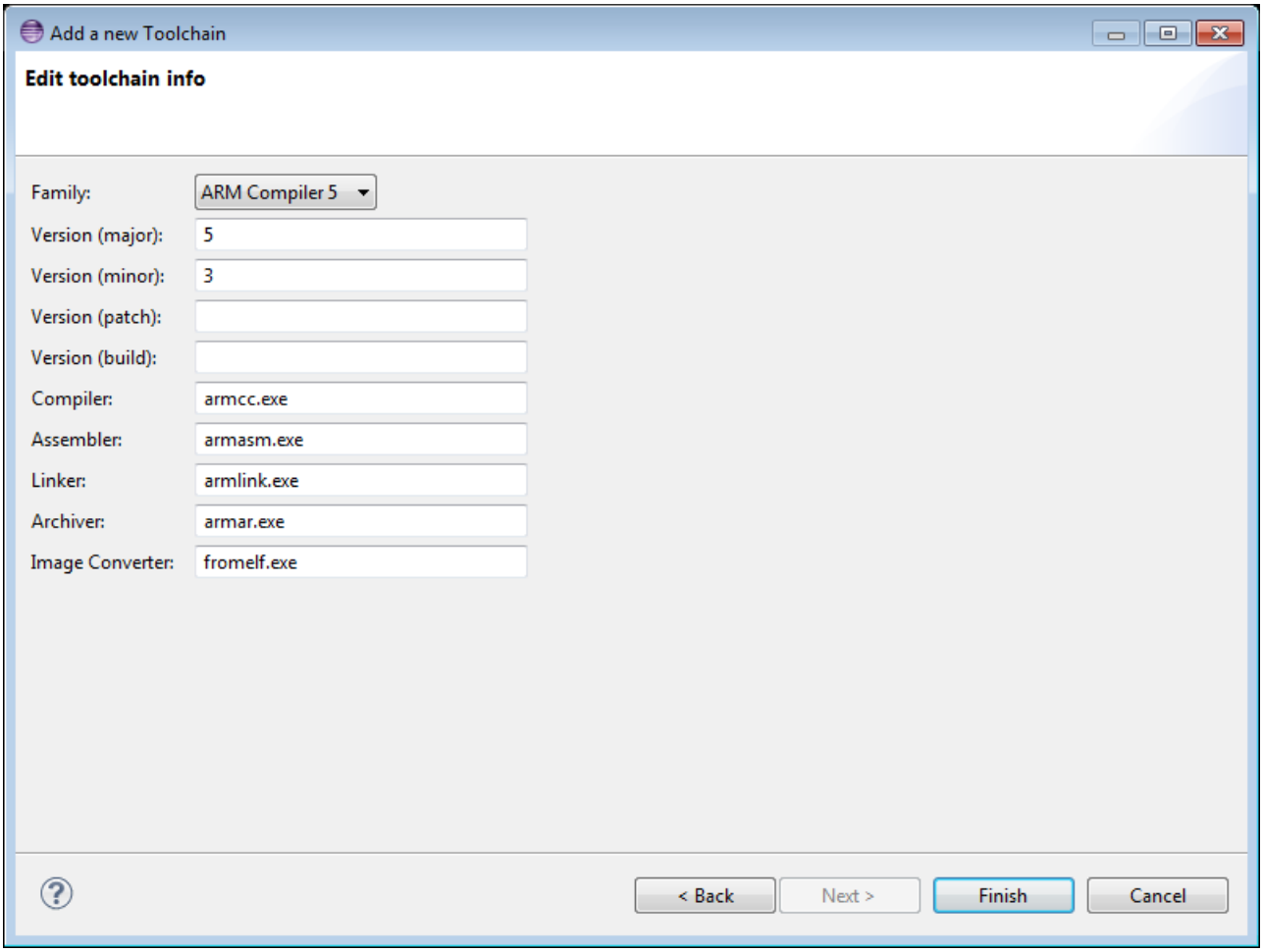


Figure 2-27 Properties for the new toolchain

Note

You must manually enter the toolchain properties if:

- The toolchain properties were not autodetected.
- The family, major version, and minor version of the new toolchain are identical to a toolchain that DS-5 already knows about.

5. Select **Apply** from the **Toolchains** preferences dialog. The new toolchain has now been registered into DS-5. You must restart DS-5 before you can use the new toolchain in the DS-5 environment.

Related tasks

2.17 Registering a compiler toolchain from the DS-5 command prompt on page 2-70.

Chapter 3
ARM® DS-5 Installation and Examples

Describes the installation and licensing requirements and provides information on the examples provided with ARM DS-5.

It contains the following sections:

3.1 System requirements

To install and use DS-5, your workstation must have a minimum specificiation of a dual core 2GHz processor (or equivalent) and 2GB of RAM.

4GB of RAM, or more, is recommended to improve performance when debugging large images, using models with large simulated memory maps, or when using ARM Streamline Performance Analyzer.

A full installation also requires approximately 3GB of hard disk space.

This section contains the following subsections:

3.1.1 Host platform requirements

DS-5 is supported (except where specified) on 32-bit and 64-bit versions of the following platforms (and service packs).

- Windows 7 Professional Service Pack 1
- Windows 7 Enterprise Service Pack 1
- Windows Server 2008 R2 (ARM Compiler 5 toolchain only)
- Windows Server 2012 (ARM Compiler 5 and 6 toolchains only)
- Red Hat Enterprise Linux 5 Desktop with Workstation option
- Red Hat Enterprise Linux 6 Workstation
- Ubuntu Desktop Edition 12.04 LTS.
- Ubuntu Desktop Edition 14.04 LTS (64-bit only).

Deprecated platforms in this release:

- Windows XP Professional Service Pack 3 (32-bit only).

——— **Note** ———

ARM Compiler 6 is only supported on 64-bit host platforms and is only supplied with the 64-bit version of DS-5.

———————————

DS-5 can coexist with ARM RVDS if they are installed into separate directories. Multiple versions of DS-5 cannot coexist on the same Windows host machine.

——— **Note** ———

All line drawings in the online help use SVG format. To view these graphics, your browser must support the SVG format.

———————————

3.1.2 Debug system requirements

Android and ARM Linux application debug require `gdbserver` to be available on your target.

The recommended version of `gdbserver` is 7.0 or later. Executables for ARM Linux and Android that are compatible with DS-5 Debugger are provided in the *`install_directory`*`/DS-5/arm` directory. You can locate these files by selecting **Help** > **ARM Extras...** from the main menu.

——— **Note** ———

DS-5 Debugger is unable to provide reliable multi-threaded debug support with `gdbserver` versions prior to 6.8.

———————————

Android and Linux application rewind require *undodb-server* to be available on your target. DS-5 Debugger copies *undodb-server* to the target for you in the **Download and Debug** connection type, but for all other connection types, you must copy it yourself. The *undodb-server* binary is located in the *install_directory*\DS-5\arm\undodb\linux directory within your installation.

Note

- Application rewind does not follow forked processes.
- When debugging backwards, you can only view the contents of recorded memory, registers, or variables. You cannot edit or change them.

DS-5 support for Android and Linux depends upon infrastructure and features introduced in specific kernel versions:

- DS-5 Debugger supports debugging native C/C++ applications and libraries on Android versions 2.2.x, 2.3.x, 3.x.x, and 4.0.
- DS-5 Debugger supports debugging ARM Linux kernel versions 2.6.28 and later.
- ARM Streamline Performance Analyzer supports ARM Linux kernel versions 2.6.32 and later.
- Application debug on *Symmetric MultiProcessing* (SMP) systems requires ARM Linux kernel version 2.6.36 or later.
- Access to VFP and NEON registers requires ARM Linux kernel version 2.6.30 or later and `gdbserver` version 7.0 or later.

3.1.3 Additional tools for Linux kernel and bare-metal debugging

ARM Linux kernel and bare-metal debugging require the use of additional tools (not supplied with DS-5) to connect to your target system.

DSTREAM, RVI, ULINKpro, and ULINKpro D, and ULINK2 debug units enable connection to physical hardware targets.

VSTREAM enables connection to RTL simulators and hardware emulators.

Note

You must use DSTREAM for ARMv8 development.

Managing firmware updates

- For DSTREAM and RVI it is recommended to use the supplied debug hardware update tool to check the firmware and update it if necessary. Updated firmware is available in the *install_directory*/DS-5/sw/debughw/firmware directory.
- For VSTREAM, the firmware is delivered as part of the VSTREAM software. To update the firmware, you must install a newer version of VSTREAM.
- For ULINK2 target connection probe to work with DS-5 Debugger, it must be upgraded with CMSIS-DAP compatible firmware. The `UL2_Upgrade.exe` program (Windows only) can upgrade your ULINK2 unit for you. The program and instructions are available in the *install_directory*/DS-5/sw/debughw/ULINK2 directory.
- For ULINKpro and ULINKpro D, DS-5 manages the firmware.

Related references

3.3 Installation directories on page 3-81.
3.4 Licensing and product updates on page 3-82.
3.5 Documentation provided with DS-5 on page 3-83.
3.6 Examples provided with DS-5 on page 3-84.

Related information

Setting up the ARM DSTREAM Hardware.

Setting up the ARM RVI Hardware.
DS-5 Knowledge Articles.
Adobe Viewer.

3.2 Installing DS-5

DS-5 32-bit and 64-bit install packages are available for Windows and Linux platforms.

The main advantage of using a 64-bit version of DS-5 is that the binaries provided with 64-bit versions are capable of processing larger data sets before hitting per-process memory limits. On Linux, 64-bit tools have fewer operating system compatibility issues.

Note

Although you can install 32-bit versions of DS-5™ on 64-bit platforms, it is recommended to install 64-bit versions of DS-5 on 64-bit operating systems.

Installing on Linux

To install DS-5 on Linux, run (not source) `install.sh` and follow the on-screen instructions.

Installing device drivers and desktop shortcuts is optional. The device drivers allow USB connection to DSTREAM and RVI debug hardware units. The desktop menu is created using the *http://www.freedesktop.org/* menu system on supported Linux platforms. If you want to install these features post-install, using root privileges, run `run_post_install_for_ARM_DS-5.sh` script available in the install directory.

Note

Tools installed by both the 32-bit and 64-bit installers have dependencies on 32-bit system libraries. You must ensure that 32-bit compatibility libraries are installed when using DS-5 on 64-bit Linux host platforms. DS-5 tools may fail to run or report errors about missing libraries if 32-bit compatibility libraries are not installed.

There are known issues when running DS-5 32-bit binaries on 64-bit Ubuntu host platforms.

The ARM Knowledgebase contains information which may help you troubleshoot these issues: *http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka14522.html*

Note

On Linux, you can use `suite_exec` to configure the environment variables correctly for DS-5. For example, run `<DS-5 install_directory>/bin/suite_exec <shell>` to open a shell with the PATH and other environment variables correctly configured. Run `suite_exec` with no arguments for more help.

Installing on Windows

To install DS-5 on Windows, run `setup.exe` and follow the on-screen instructions.

During installation, you may be prompted to install device drivers. These drivers allow USB connections to DSTREAM, RVI, and Energy Probe hardware units. They also support networking for the simulation models. It is recommended to install these drivers if you intend to use these features.

Note

You can safely ignore warnings displayed when these drivers are installed and continue with the installation.

3.3 Installation directories

Various directories are installed with DS-5 that contain example code and documentation. The DS-5 documentation refers to these directories as required.

The main installation, examples, and documentation directories are identified in the following table. The *install_directory* shown is the default installation directory. If you installed the product in a different directory, then the path names are relative to your chosen directory.

Table 3-1 DS-5 default directories

Directory	Windows	Linux
install_directory	For 32-bit version of Windows: `C:\Program Files\DS-5` For 64-bit version of Windows with 64-bit version of DS-5 installed: `C:\Program Files\DS-5` For 64-bit version of Windows with 32-bit version of DS-5 installed: `C:\Program Files (x86)\DS-5`	`~/DS-5`
arm_directory	*install_directory*`\arm\...`	*install_directory*`/arm/...`
examples_directory	*install_directory*`\examples\...`	*install_directory*`/examples/...`
tools_directory	*install_directory*`\bin\...`	*install_directory*`/bin/...`
documents_directory	*install_directory*`\documents\...`	*install_directory*`/documents/...`

Related references

3.1 System requirements on page 3-77.
3.4 Licensing and product updates on page 3-82.
3.5 Documentation provided with DS-5 on page 3-83.
3.6 Examples provided with DS-5 on page 3-84.

3.4 Licensing and product updates

DS-5 is a licensed product that uses the Flex*Net* license management software to enable features corresponding to specific editions.

To compare DS-5 editions, see: *http://ds.arm.com/ds-5/compare-ds-5-editions/*

To request a license or to access the latest DS-5 product information and updates, go to the ARM Self-Service Portal.

You can access the license management software by selecting **ARM License Manager...** from the Help menu in Eclipse for DS-5.

Related tasks

2.16 Changing the Toolkit on page 2-69.

Related references

3.1 System requirements on page 3-77.
3.3 Installation directories on page 3-81.
3.5 Documentation provided with DS-5 on page 3-83.
3.6 Examples provided with DS-5 on page 3-84.
2.15 Managing DS-5 licenses on page 2-60.

Related information

ARM Forums.
ARM DS-5 License Management Guide.
ARM Self-Service Portal.

3.5 Documentation provided with DS-5

DS-5 includes example projects and documentation.

To view a list of documentation available with DS-5, see: *http://ds.arm.com/developer-resources/ds-5-documentation/*

To access the documentation from within DS-5, from the main menu, select **Help** > **Help Contents** and navigate to **ARM DS-5 Documentation**.

Documentation on using the examples is available in *install_directory*`\DS-5\examples\docs`.

Related references

3.1 System requirements on page 3-77.

3.3 Installation directories on page 3-81.

3.4 Licensing and product updates on page 3-82.

3.6 Examples provided with DS-5 on page 3-84.

Related information

DS-5 documentation.

3.6 Examples provided with DS-5

DS-5 provides a selection of examples to help you get started:

- Bare-metal software development examples for ARMv7 and earlier that illustrate compilation with the ARM Compiler 5 and ARMv7 bare-metal debug. The code is located in the archive file `<examples_directory>\Bare-metal_examples_ARMv7.zip`.
- Bare-metal software development examples for ARMv8 that illustrate compilation with the ARM Compiler 6 and ARMv8 bare-metal debug. The code is located in the archive file `<examples_directory>\Bare-metal_examples_ARMv8.zip`.

  Note

  ARMv8 features are available only in the DS-5 Ultimate Edition.

- Bare-metal example projects for supported boards that demonstrate board connection and basic debug into on-chip RAM. The files are located in the archive file, `examples_directory\Bare-metal_boards_examples.zip`.
- ARM Linux examples that illustrate build, debug, and performance analysis of simple C/C++ console applications, shared libraries, and multi-threaded applications. These examples run on a*Fixed Virtual Platform* (FVP) that is preconfigured to boot ARM Linux. The files are located in the archive file, `examples_directory\Linux_examples.zip`.
- The *RTX Real-Time Operating System* (RTX-RTOS) source files and examples demonstrate the RTX-RTOS applications. The files are located in the archive file, `examples_directory\CMSIS_RTOS_RTX.zip`.
- Optional packages with source files, libraries, and prebuilt images for running the examples.

  These can be downloaded from: *https://silver.arm.com/browse/DS500* (Registration required).

  — Linux distribution project with header files and libraries for the purpose of rebuilding the ARM Linux examples.
  — Legacy Linux SD card image for the BeagleBoard configured for DS-5.
  — Legacy Linux SD card image for the BeagleBoard-xM configured for DS-5.

You can extract these examples to a working directory and build them from the command-line, or you can import them into Eclipse using the import wizard. All examples provided with DS-5 contain a preconfigured Eclipse launch script that enables you to easily load and debug example code on a target.

Each example provides instructions on how to build, run, and debug the example code. You can access the instructions from the main index, `examples_directory\docs\index.html.`

Related concepts

1.4 About Fixed Virtual Platform (FVP) on page 1-16.

Related tasks

2.3 Importing the example projects into Eclipse on page 2-27.

Related references

3.1 System requirements on page 3-77.
3.3 Installation directories on page 3-81.
3.4 Licensing and product updates on page 3-82.
3.5 Documentation provided with DS-5 on page 3-83.

Related information

Using the welcome screen.
ARM Development Studio 5 (DS-5).