Gateway SIDK Installation Guide

Firmware Version 9.4
Document Version 1.2

BRINGING TECHNOLOGY HOME
www.pace.com

Notice to Users

2WIRE PROVIDES NO WARRANTY WITH REGARD TO THIS MANUAL, THE SOFTWARE, OR OTHER INFORMATION CONTAINED HEREIN, AND HEREBY EXPRESSLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE WITH REGARD TO THIS MANUAL, THE SOFTWARE, OR SUCH OTHER INFORMATION, IN NO EVENT SHALL 2WIRE, INC. BE LIABLE FOR ANY INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, WHETHER BASED ON TORT, CONTRACT, OR OTHERWISE, ARISING OUT OF OR IN CONNECTION WITH THIS MANUAL, THE SOFTWARE, OR OTHER INFORMATION CONTAINED HEREIN OR THE USE THEREOF.

2Wire, Inc. reserves the right to make any modification to this manual or the information contained herein at any time without notice. The software described herein is governed by the terms of a separate user license agreement.

Updates and additions to software may require an additional charge. Subscriptions to online service providers may require a fee and credit card information. Financial services may require prior arrangements with participating financial institutions.

11192010

5100-001021-000

Contents

About This Guide

This document is a startup guide for installing and setting up the Software Integration Development Kit (SIDK) for the Pace gateway firmware. This document helps you get started set up your environment so that you can use the SIDK to customize or integrate your own or other third party software code and applications with the Pace gateway firmware.

Audience

This guide is intended for use by system integrators.

Prerequisites

The users of this document are expected to be proficient in:

- Networking concepts and technologies
- Internet gateway devices
- Firmware and framework development

Document Structure

This document has the following major topics:

- The Pace Gateway SIDK. Describes Pace's SIDK, which you can use to customize the gateway firmware and integrate with a variety of hardware platforms. as well as unique software requirements. It also lists a minimum set of requirements to install and run the SIDK.
- Installing CentOS 5.4. Provides a procedure to install CentOS.
- Installing SIDK Components. Provides procedures for installing the components that comprise the SIDK.
- Configuring Eclipse for the SIDK Project. Lists the steps to install and set up the Pace SIDK using Eclipse.
- Upgrading the Pace SIDK. Lists the steps to upgrade from an earlier version of the SIDK.

Style Conventions

The following style conventions are used in this guide:

Note	Notes contain incidental information about the subject. In this guide, they are used to provide additional information about the product and to call attention to exceptions.

Caution notes identify information that helps prevent damage to hardware or loss of data.

Warning notes identify information that helps prevent injury or death.

Typographical Conventions

Convention	Used For
Blue Text	Cross references
Bold	Interface elements that are clicked or selected
Italic	Emphasis, book titles, variables, list terms, user inputs
`Monospace`	Command syntax and code
`*Monospace Italic*`	Variables within command syntax and code

CHAPTER 1

The Pace Gateway SIDK

The Pace Gateway SIDK comprises various components and tool chains, which you can use to customize Pace's firmware to meet specific device requirements.

Using the development kit, you can write new code, modify the code available in the tree, and package the results into target images for all supported platforms. This allows rapid prototyping and releases of features across hardware.

A secondary benefit of the SIDK is that updates can be incorporated from Pace without impacting the external source code, since it is isolated into its own directory structure.

Development Kit Contents

The development kit consists of the following:

- Toolchain for developing against the target hardware
  - GCC 4.2.4
  - GDB 6.8
  - Binutils 2.19.1
  - Flex and Bison
  - Additional tools as required
- Kernel and Modules compiled for the target hardware
- Flash image utilities
  - SquashFS, TL, JFFS2 support
  - Gang Image Format for programming flash
- Libraries, Header files, and Man pages
  - Embedded C Runtime (libc, ld.so)
  - C++ Runtime (libstdc++)
  - Internationalization and UTF-8 support (libicuuc and libiconv)
  - Additional Libraries documented in Man pages
- Applications
  - Web Server (apache)
  - Command Line Interface (tcli)
  - CPE WAN Management through TR-069 (CWMP)
  - Additional applications as documented

System Requirements

To successfully set up and run the Pace SIDK, you must have the following:

System Requirements	
Computer	x86_64 or i386 (CPU must support virtualization capabilities)
Processor	Intel 64-bit or compatible -or- Intel 32-bit or compatible
RAM	2 GB (Minimum)

System Requirements	
Hard Disk Space	1 TB for CVS server 40 GB (Minimum) for CVS client computers
Operating System (OS)	CentOS 5.4 for x86_64 or i386 (As per the hardware)
Versioning System	CVS-1.11.22-7 (Package can be installed using CentOS 5.4 Package Manager)
Java Platform	Sun Java JDK Standard Edition 6
Integrated Development Environment (IDE)	Eclipse 3.5 with C/C++ Development Tools (CDT)

CHAPTER 2

Installing CentOS 5.4

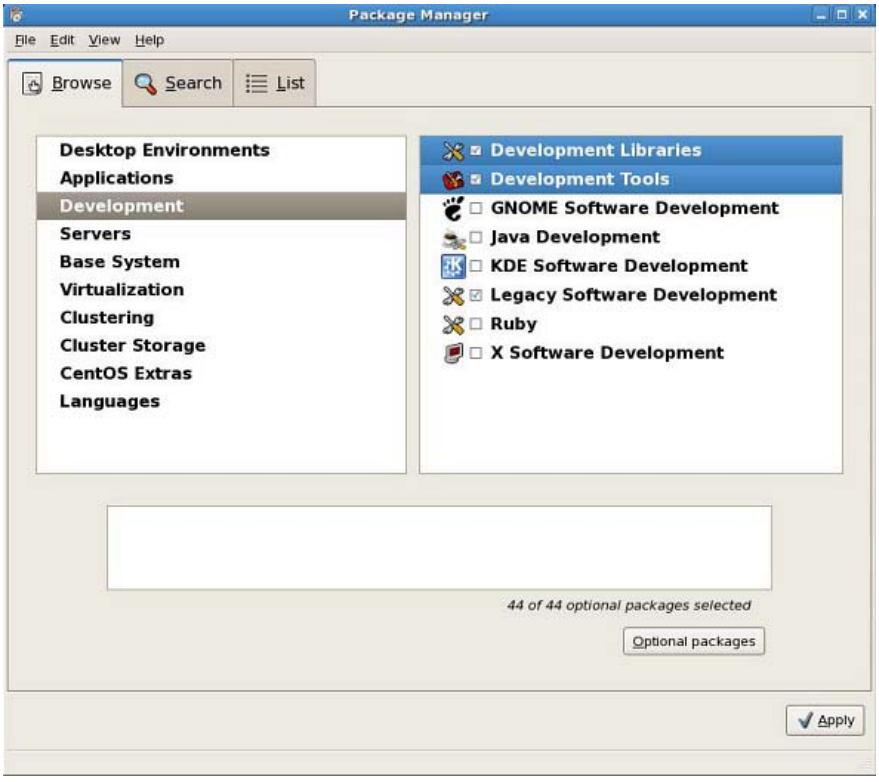
To run the Pace SIDK, you need to first install the Community Enterprise Operating System (CentOS) enterprise-class Linux distribution.

To install CentOS 5.4:

1. Obtain the latest copy of the CentOS 5.4.
2. Install the OS on the computer along with the following packages:
   - GNOME Desktop Environment
   - Virtualization

   Note Disable SELinux. (It is a workaround for the CVS pserver.)

3. Install all **Development Libraries** and **Development Tools** packages.



4. Run **Package Updater** to apply all the available updates after CentOS installation is complete.

5. Run **User Manager** to create a user account `user` in CentOS.

CHAPTER 3

Installing SIDK Components

The Pace SIDK has the following components:

- *Concurrent Versioning System (CVS)*. CVS is required to serve as a versioning system.
- *Java Development Kit (JDK)*. JDK is required to use Eclipse.
- *Eclipse*. Eclipse is an Integrated Development Environment (IDE) framework used to customize the gateway code.

Installing CVS (Server Only)

You must install and configure CVS in order to check in and check out the gateway source code, and build a gateway image to be installed on the device.

Note This step is required only if the CVS repository and the client is to be installed on the same computer. If only a CVS client is installed, refer to Configuring Eclipse for the SIDK Project on page 13.

Perform the following CVS-related tasks before customization of firmware:

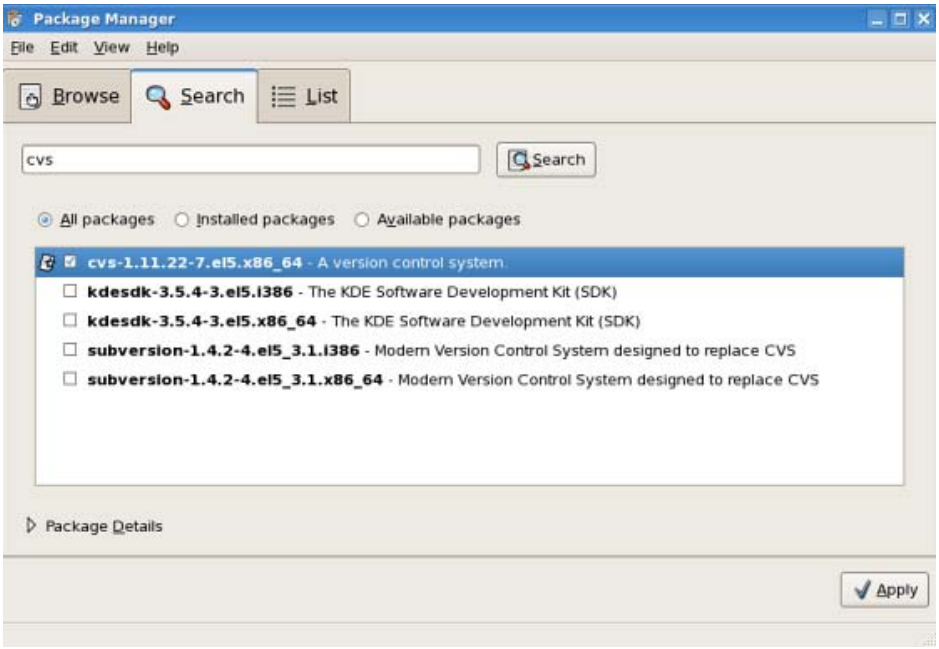
- CVS Installation in CentOS 5.4
- CVS Configuration

CVS Installation in CentOS 5.4

To install CVS:

1. On the menu bar, click **Applications**, and select **Add/Remove Software**.
2. On the **Package Manager** window, click the **Search** tab.
3. Enter **CVS** in the **Search** field, and click **Search**.
4. Select the **CVS-1.11.22** check box, and click **Apply**.

5. Confirm and import the key to complete the CVS installation.



CVS Configuration

Configuring CVS enables you to integrate the firmware code operations in the CVS.

To configure CVS, you need to set up the repository:

1. Log in as `root` and add a `cvs` user:

   ```
   [root@localhost ~]# /usr/sbin/useradd –r –n –d /var/cvsroot –c “CVS Repository” –m cvs
   ```

2. Log in as `cvs` and set environment variable:

   ```
   [root@localhost ~]# su –l cvs
   [cvs@ localhost ~]$ pwd
   /var/cvsroot
   [cvs@localhost ~]$ cvs -d /var/cvsroot init
   ```

3. Log in as `root` and set up pserver for CVS on the local host.
   a. Edit the cvs file located in /etc/xinetd.d/:

      ```
      [root@localhost ~]# vi /etc/xinetd.d/cvs
      ```

Edit the cvs file to set inetd to start the CVS server and run as user `cvs`:

```
# default: off
# description: The CVS service can record the history of your source files.
CVS stores all the versions of a file in a single file in a clever way that
only stores the differences between versions.
service cvspserver
{
    disable = no
    port = 2401
    socket_type = stream
    protocol = tcp
    wait = no
    user = cvs
    passenv = PATH
    server = /usr/bin/cvs
    env = HOME=/var/cvsroot
    server_args = -f --allow-root=/var/cvsroot pserver
    # bind = 127.0.0.1
}
```

b. Install and start xinetd. An Internet connection is required.

```
[root@localhost ~]# yum install xinetd
[root@localhost ~]# /etc/init.d/xinetd start
```

c. Create and configure the CVS writers file and associated password:

- check out CVSROOT:
  ```
  [root@localhost ~]# cvs –d /var/cvsroot co CVSROOT
  [root@localhost ~]# cd CVSROOT
  ```
- Create and edit the writers file using vi or gedit:
  ```
  [root@localhost CVSROOT]# vi writers
  ```
  In the editor, add a single line in the writers file: `user`
- Check in and commit the new file into CVSROOT:
  ```
  [root@localhost CVSROOT]# cvs –d /var/cvsroot add writers
  [root@localhost CVSROOT]# cvs –d /var/cvsroot commit
  ```
  CVS opens an editor and adds a comment.
- Create the password file which contains the password for the user account:
  ```
  [root@localhost CVSROOT]# htpasswd –c passwd user
  ```
  Enter and confirm the password.
- Edit passwd file and adapt for CVS:

  Edit the passwd file, and append `:cvs` to the `user` entry.

  For example, `user:DC2fgt5hyhdf:cvs`

  Save the file.

- check in and commit the new file into CVSROOT:
  ```
  [root@localhost CVSROOT]# cvs –d /var/cvsroot add passwd
  [root@localhost CVSROOT]# cvs –d /var/cvsroot commit
  ```
- Copy passwd file into the repository:
  ```
  [root@localhost CVSROOT]# cp passwd /var/cvsroot/CVSROOT
  ```

Note Copying the passwd file into the repository helps commit the previous step.

d. Configure the config file from the checked-out CVSROOT:

- Edit config file located in /root/CVSROOT using vi or gedit:
  ```
  [root@localhost CVSROOT]# vi config
  ```

  In the config file, uncomment and set the value of `SystemAuth` to `no`:

```
# Set this to "no" if pserver should not check system users/passwords:
SystemAuth=no

# Put CVS lock files in this directory rather than directly in the
repository.
#LockDir=/var/lock/cvs

# Set `TopLevelAdmin' to `yes' to create a CVS directory at the top
# level of the new working directory when using the `cvs checkout' command.
#TopLevelAdmin=no

# Set `LogHistory' to `all' or `TOEFWUPCGMAR' to log all transactions to
the # history file, or a subset as needed (i.e. `TMAR' logs all write
operations)
#LogHistory=TOEFWUPCGMAR

# Set `RereadLogAfterVerify' to `always' (the default) to allow the
verifymsg
# script to change the log message.  Set it to `stat' to force CVS to
verify
# that the file has changed before reading it (this can take up to an extra
# second per directory being committed, so it is not recommended for large
# repositories.  Set it to `never' (the previous CVS behavior) to prevent
# verifymsg scripts from changing the log message.
#RereadLogAfterVerify=always
```

- Check in the config file:
  ```
  [root@localhost CVSROOT]# cvs –d /var/cvsroot ci config
  ```

  CVS opens an editor and adds a comment for the applied change.

e. Test the pserver connection:

```
[root@localhost ~]# cvs –d :pserver:user@localhost:/var/cvsroot login
```

Enter the password associated to the user account.

If no error message is displayed, then the connection is successful.

4. Import the SIDK into CVS.
   a. Create a sidk directory and extract sidk.tar:

      ```
      [user@localhost ~]$ mkdir sidk
      [user@localhost ~]$ cd sidk/
      [user@localhost sidk]$ tar xf ~/<locationOfSIDKfile>/sidk.tar
      ```
      (can take up to 60s)
      ```
      [user@localhost sidk]$ cd ..
      ```
   b. Import the SIDK into CVS:
      - Log in as `root` and change mode for /var/cvsroot directory:
        ```
        [root@localhost var~]$ chmod –R 775 cvsroot
        [root@localhost var~]$ chmod –R a+s cvsroot
        ```
      - Add a `cvs` group and change the owner and group of /var/cvsroot to `cvs`:
        ```
        [root@localhost var~]$ groupadd –r –f cvs
        [root@localhost var~]$ chown –R cvs.cvs cvsroot
        ```
      - Log in as `user` and import sidk into the CVS:
        ```
        [user@localhost ~]$ cd sidk/
        [user@localhost sidk]$ cvs -d /var/cvsroot import sidk TWOWIRE-9_4 SIDK-9_4
        ```
        CVS will start an editor and prompt for a message.

        Enter `Initial import` and quit the editor.

Check out SIDK Directories

The section provides an example on how to check out using shell commands. Repository check out is done from Eclipse within the SIDK project.

To check out the entire SIDK file from the repository to the local work environment:

1. Create a Work Environment:
   ```
   [cvs@localhost ~]$ mkdir Work_Env
   [cvs@localhost ~]$ cd Work_Env
   ```
2. Check out the SIDK:
   ```
   [cvs@localhost Work_Env]$ cvs -d ~/cvsroot co sidk
   ```

Check in an SIDK File

This section provides information on how to check in a modified file, how to check the differences between the original and modified file, and how to check the log. It provides an example on how to check in using shell commands. File check in is done using Eclipse within the SIDK project.

The dfa.c file located in /home/*username*/Work_Env/sidk/cross/gawk/ is modified.

To modify the dfa.c file:

1. Edit dfa.c file using vi or gedit.
2. Add the following comment:
   ```
   /* ADD comment for test */.
   ```
3. Save the file.
4. Check the differences:
   ```
   [cvs@localhost gawk]$ cvs diff dfa.c
   Index: dfa.c
   ===================================================================
   RCS file: /home/cvs/cvsroot/sidk/cross/gawk/dfa.c,v
   retrieving revision 1.1.1.1
   diff -r1.1.1.1 dfa.c
   ```

```
22a23,25
> /* ADD comment for test */
>
>
[1]+  Done                    kedit dfa.c
```

5. Check in the modified file:

```
[cvs@localhost gawk]$ cvs ci dfa.c
Checking in dfa.c;
/home/cvs/cvsroot/sidk/cross/gawk/dfa.c,v  <--  dfa.c
new revision: 1.2; previous revision: 1.1
done
```

6. Check the log:

```
[cvs@localhost gawk]$ cvs log dfa.c
RCS file: /home/cvs/cvsroot/sidk/cross/gawk/dfa.c,v
Working file: dfa.c
head: 1.2
branch:
locks: strict
access list:
symbolic names:
        SIDK-9_4: 1.1.1.1
        TWOWIRE-9_4: 1.1.1
keyword substitution: kv
total revisions: 3;     selected revisions: 3
description:
----------------------------
revision 1.2
date: 2010/03/09 22:51:01;  author: cvs;  state: Exp;  lines: +3 -0
I added a comment line
----------------------------
revision 1.1
date: 2010/03/09 22:23:47;  author: cvs;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2010/03/09 22:23:47;  author: cvs;  state: Exp;  lines: +0 -0
Initial Import
=============================================================================
```

Installing Java JDK

To install Java JDK:

1. Obtain a copy of the Java JDK either through the add-on SIDK CD or go to: http://java.sun.com/javase/downloads/widget/jdk6.jsp
2. Create and copy the Java JDK rpm in addons/java/ :

```
[user@localhost ~]$ mkdir addons
[user@localhost ~]$ cd addons/
[user@localhost addons]$ mkdir java
[user@localhost addons]$ cp /media/XXXX/<java jdk rpm> /home/user/addons/java
```

3. Log in as `root` and install Java JDK:

```
[root@localhost java]# chmod a+x <java jdk rpm>
[root@localhost java]# ./<java jdk rpm>
```

Note You can also use the `yum install java` command to install Java JDK.

Eclipse

To install and run the Eclipse framework:

1. Obtain a copy of the Eclipse CDT either through the add-on the SIDK CD or go to: http://www.gtlib.gatech.edu/pub/eclipse/technology/epp/downloads/release/galileo/SR2/
2. Create an eclipse directory and copy the eclipse tarball:

```
[user@localhost ~]$ mkdir eclipse
[user@localhost ~]$ cd eclipse/
[user@localhost eclipse]$ cp /media/XXXX/<eclipse tar> /home/user/eclipse
```

3. Install Eclipse:

```
[user@localhost eclipse]$ tar xf <eclipse tar>
```

Note You may need to run the extract after logging in as a root user.

4. Edit the eclipse.ini file to modify the Xmx value located on the last line of the file (replace 256m by 2g) before starting Eclipse:

```
-startup
plugins/org.eclipse.equinox.launcher_1.0.201.R35x_v20090715.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.0.200.v20090519
-product
org.eclipse.epp.package.cpp.product
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
-vmargs
-Dosgi.requiredJavaVersion=1.5
-XX:MaxPermSize=256m
-Xms40m
-Xmx2g
```

5. Run Eclipse:
   `[user@localhost eclipse]$ ./eclipse`

CHAPTER 4

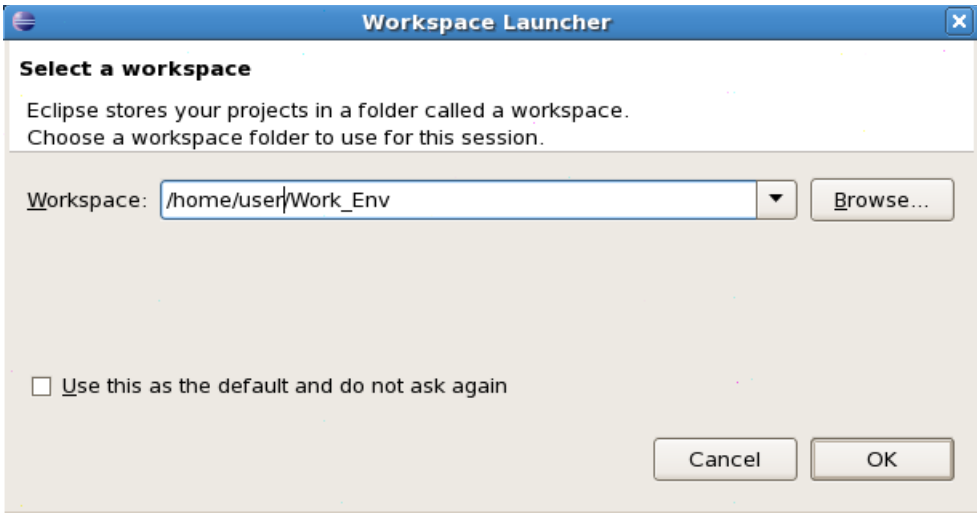
Configuring Eclipse for the SIDK Project

This chapter describes the procedure for checking out the SIDK on a client using Eclipse over the network.

Creating a New Project

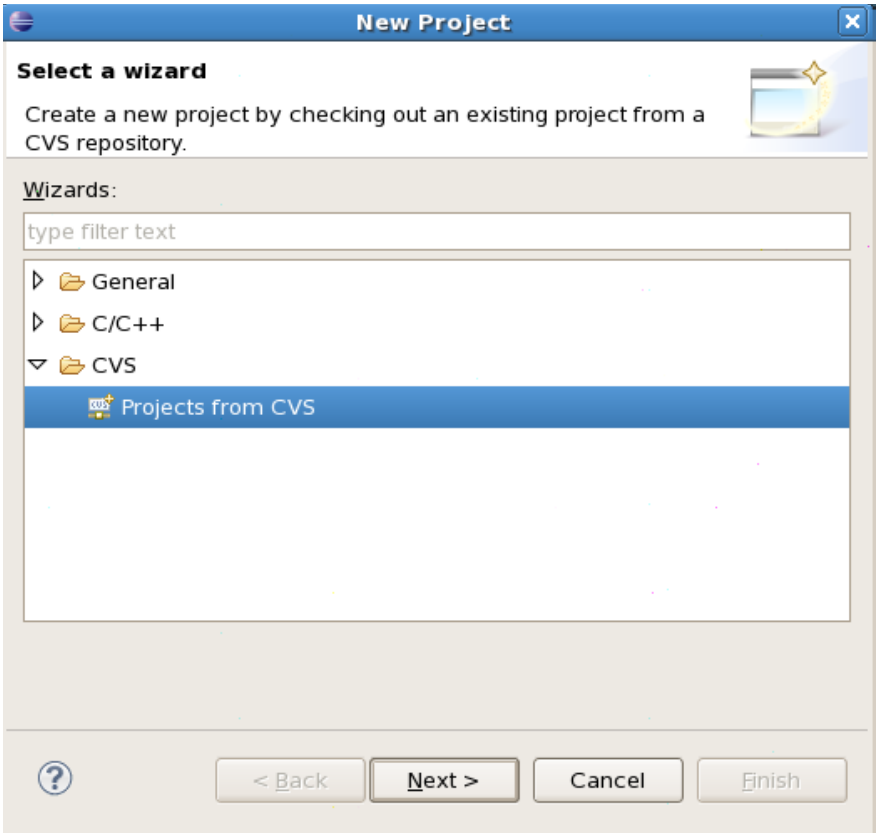
To create a new project in Eclipse:

1. On the **Workspace Launcher** window, select a workspace using **Browse** to check out the SIDK project. You can also enter the path of the workspace in the **Workspace** field.



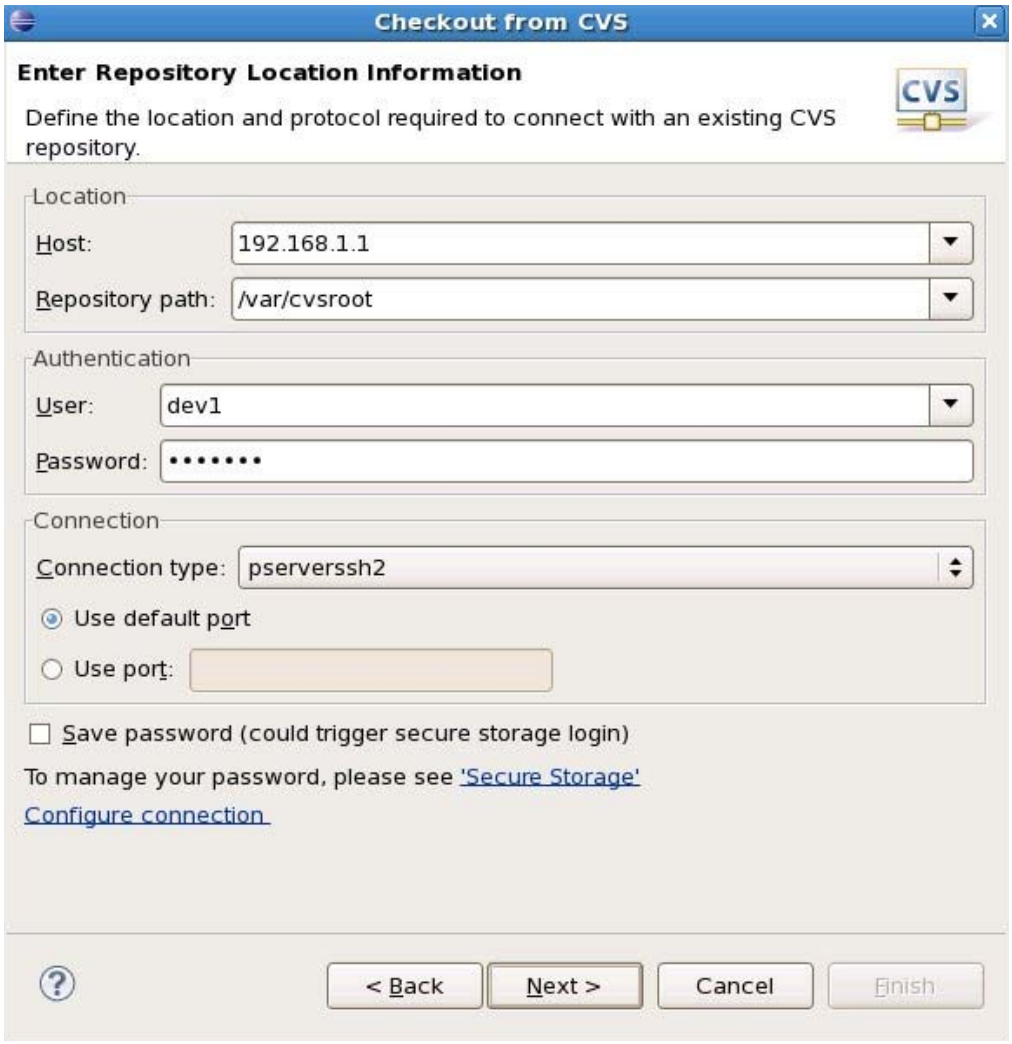
2. Click **OK**.
3. Click the **File** menu, select **New**, and then select **Project**.

4. On the **New Project** window, expand **CVS**, and select **Projects from CVS**.



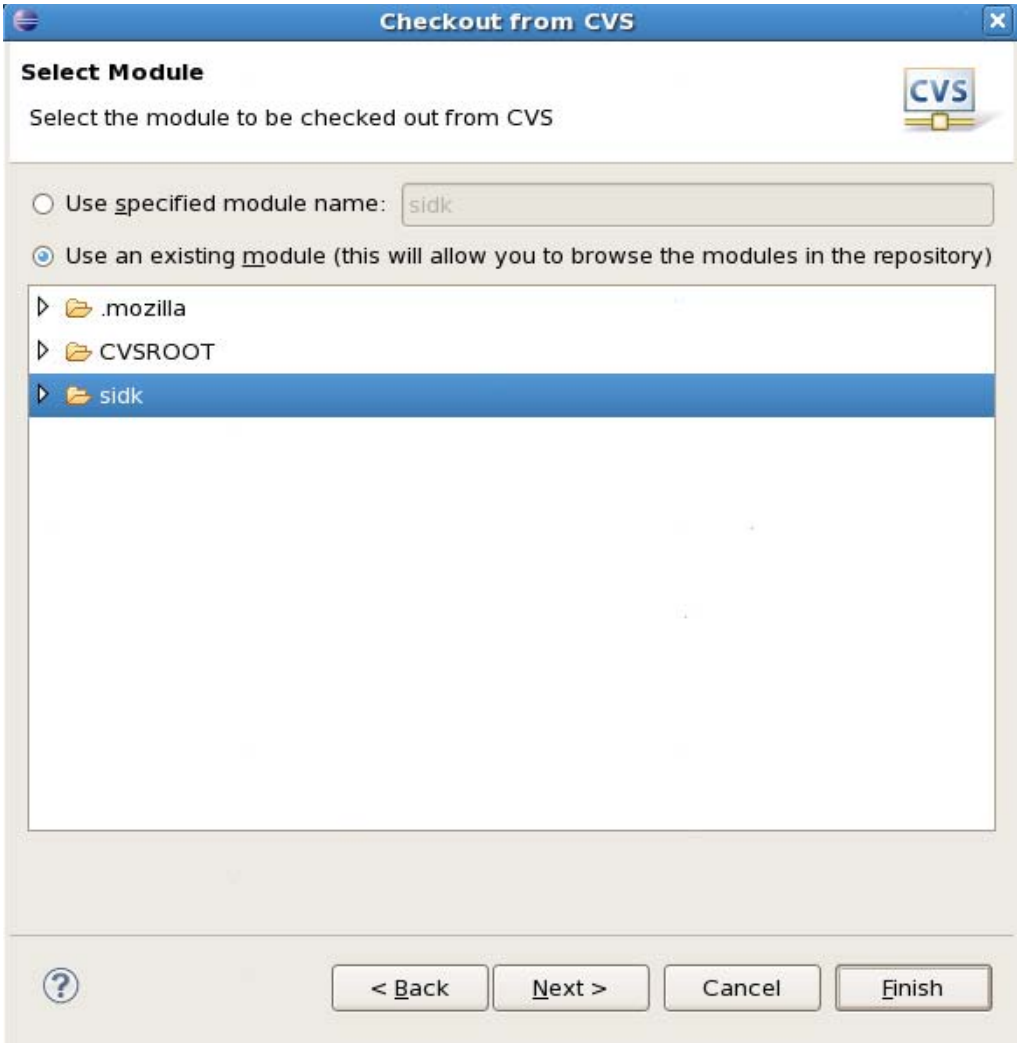
5. Click **Next**.
6. On the **Checkout from CVS** window, configure the following repository details:
   a. Enter the IP address of the CVS server in the **Host** field.
   b. Enter the path of the CVS repository `/var/cvsroot` in the **Repository path** field.
   c. Enter the name of the CVS user in the **User** field.
   d. Enter the password associated to the CVS user in the **Password** field.

e. Enter the type of connection `pserverssh2` in the **Connection type** field.



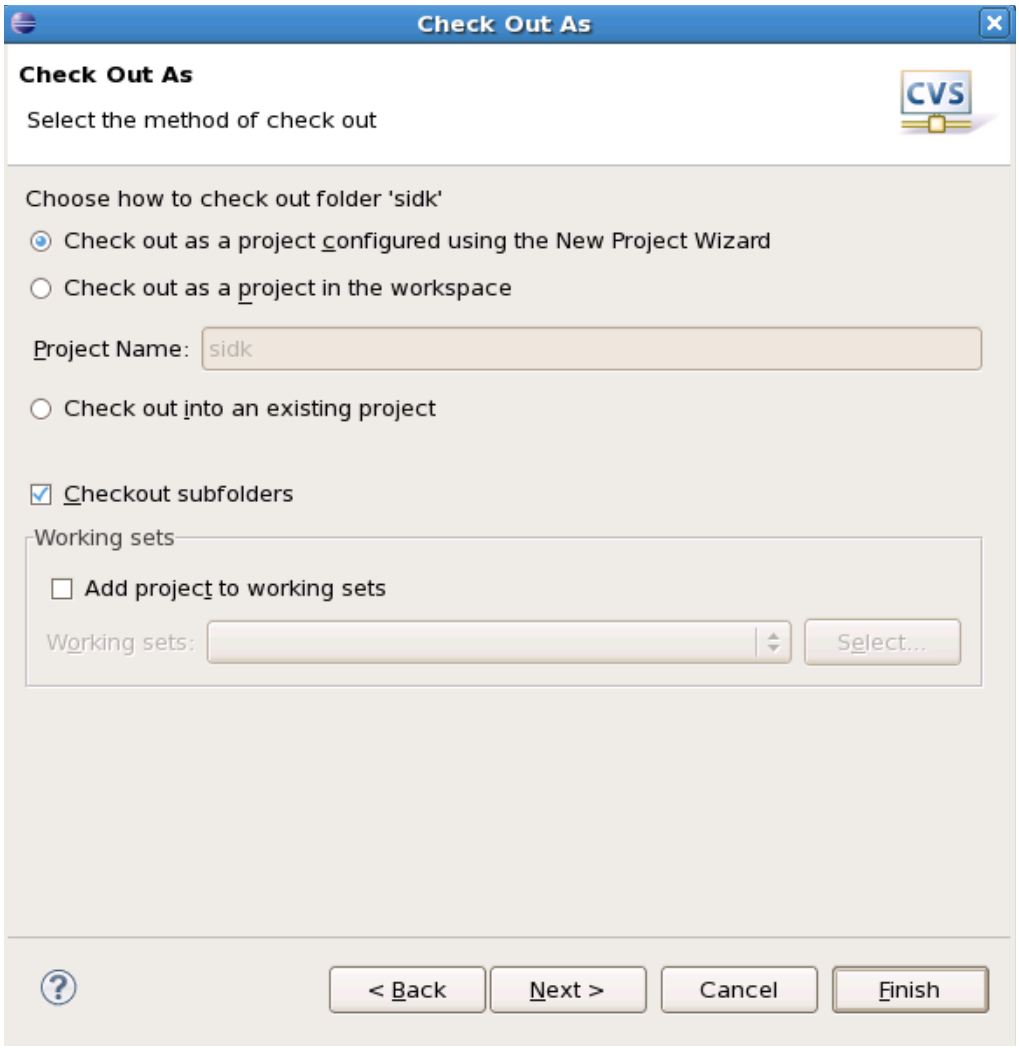
7. Click **Next**.

8. On the **Checkout from CVS** window, click **Use an existing module**. The **sidk** module appears.



9. Select **sidk**, and click **Next**. The system will prompt you to select a check out method.

10. Click **Check out as a project configured using the New Project Wizard**.



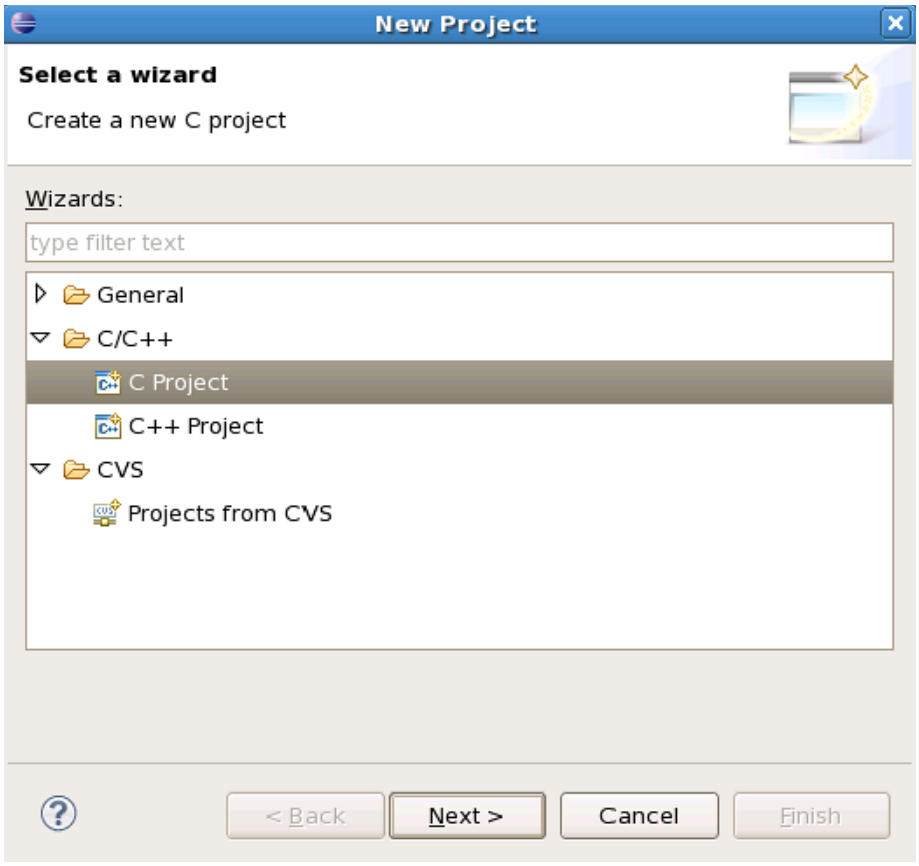
11. Click **Next**.

    The **Check Out As** window to select tag appears.

12. Leave the **Select a Tag** field empty, and click **Finish**.

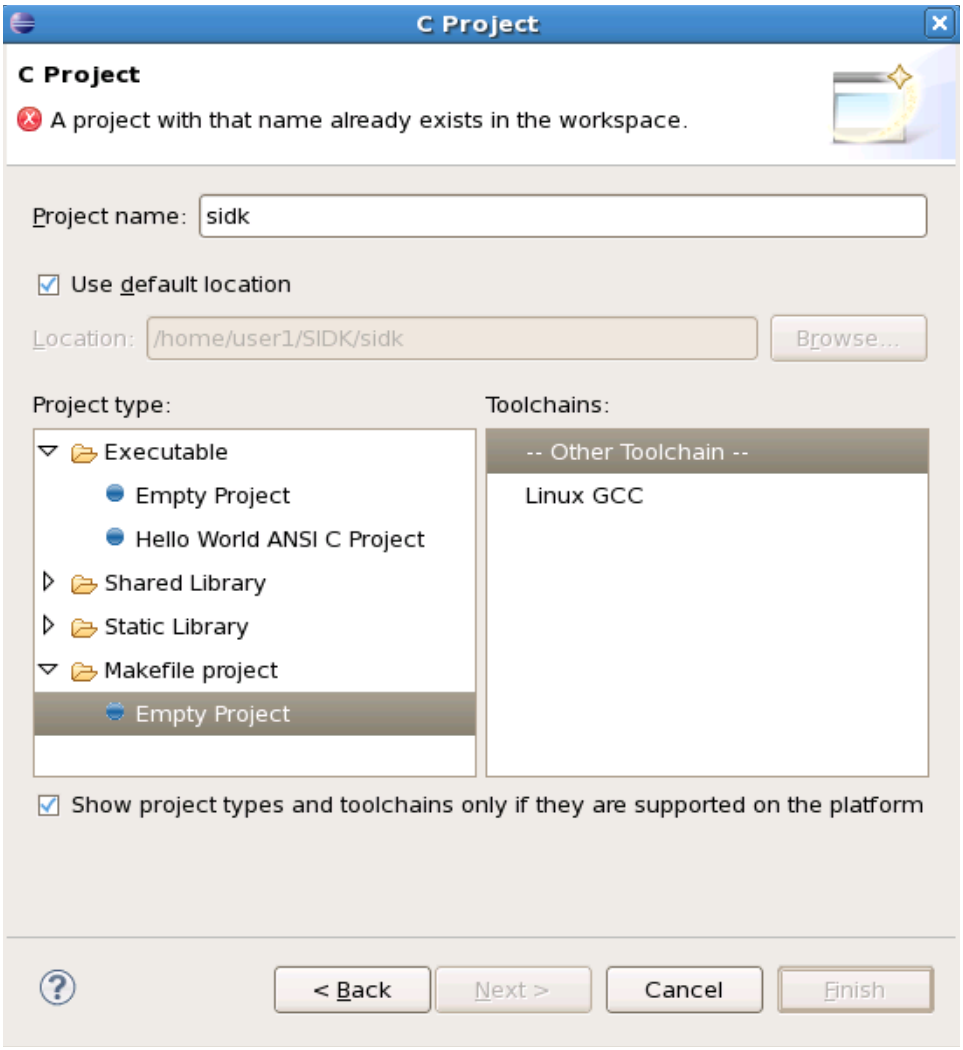
The **New Project** window appears.

13. On the **New Project** window, expand **C/C++** and select **C Project**.



14. Click **Next**.
15. On the **C Project** window, enter the name of the C Project `sidk` in the **Project Name** field.

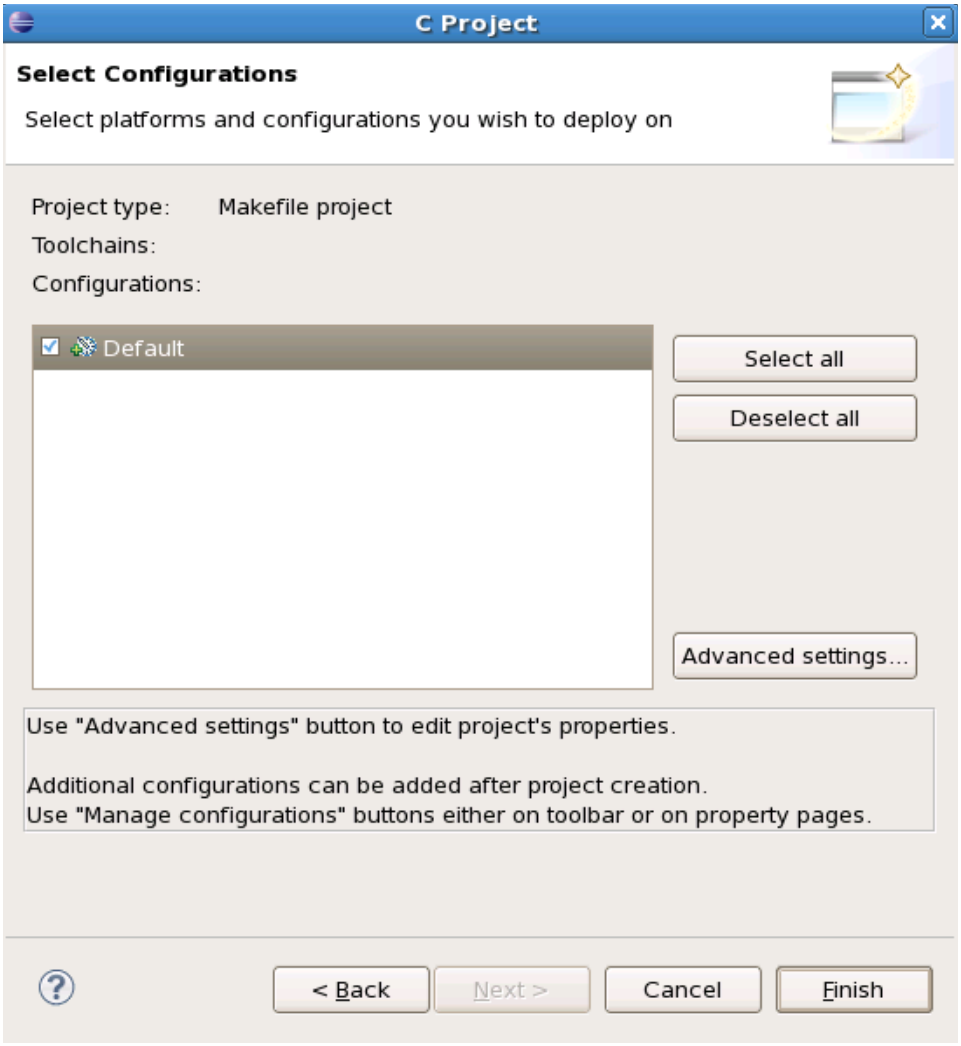
16. Expand **Makefile project** and select **Empty Project**.



17. Select **Other Toolchain** in the **Toolchains** section and click **Next**.

    The **C Project** window to select configurations appears.

18. Ensure that the **Default** check box is selected.



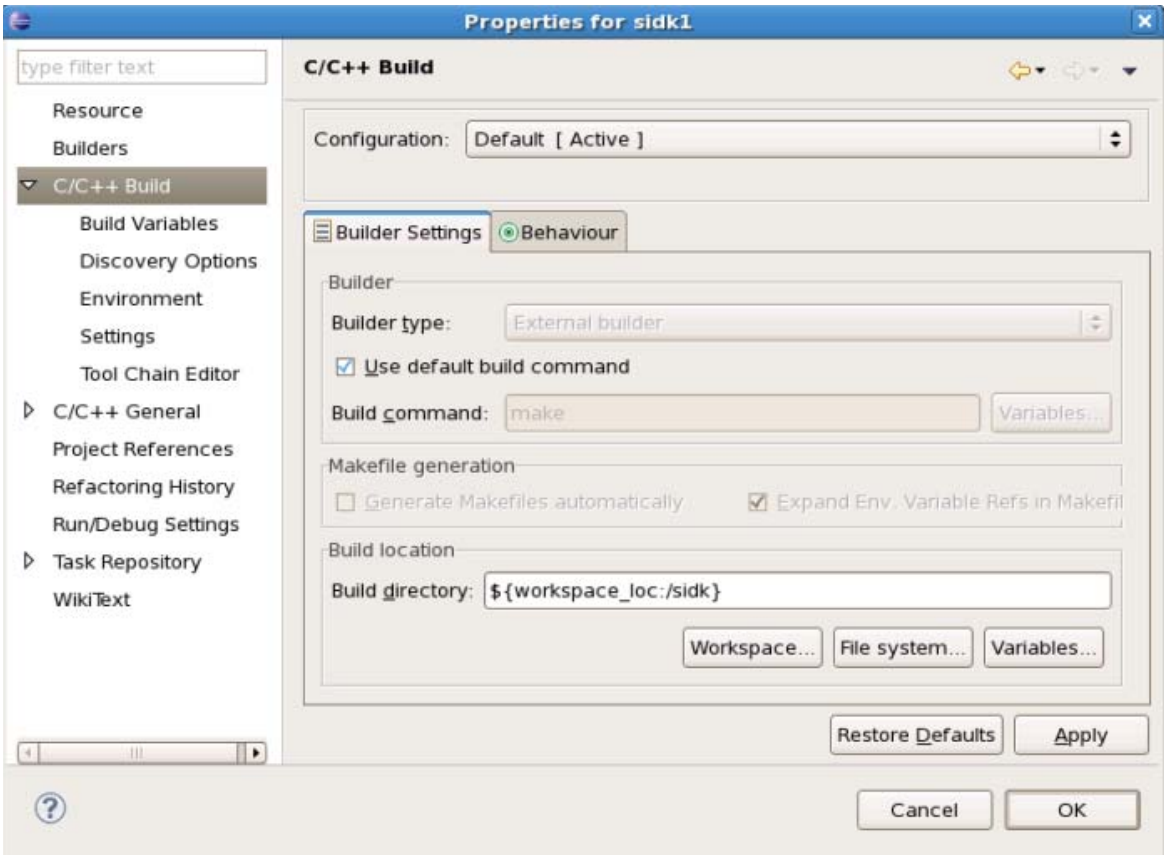
19. Click **Advanced Settings**.

    The **Properties for sidk** window appears. The window displays the properties for sidk project.

20. Click **C/C++ Build** on the left pane.

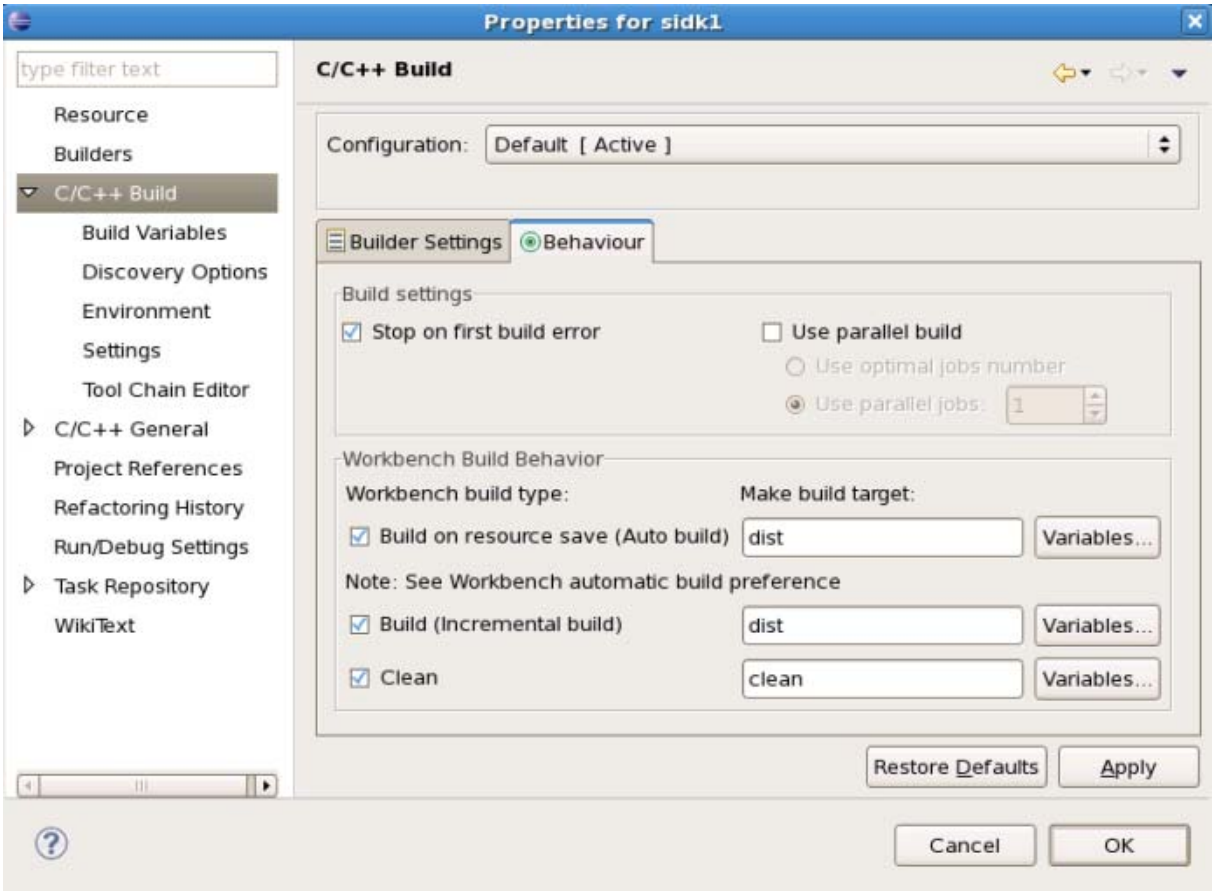
    The **Builder Settings** tab appears by default.

21. Ensure that the **Use default build command** check box is selected.



22. Click the **Behaviour** tab, and modify the values of **Make Build target** field by selecting the required values in the **Build Settings** and **Workbench Build Behaviour** sections:
    a. Select the **Build on resource save (Auto build)** check box.
    b. Click the **Make build target** field corresponding to **Build on resource save (Auto build)**.
    c. Delete `all` and enter `dist`.
    d. Click the **Make build target** field corresponding to **Build (Incremental build)**.

e. Delete `all` and enter `dist`.



23. Click **Apply** and then click **OK**.

    The **C Project** window to select configuration appears.

24. Click **Finish**.

The SIDK project is ready for editing.

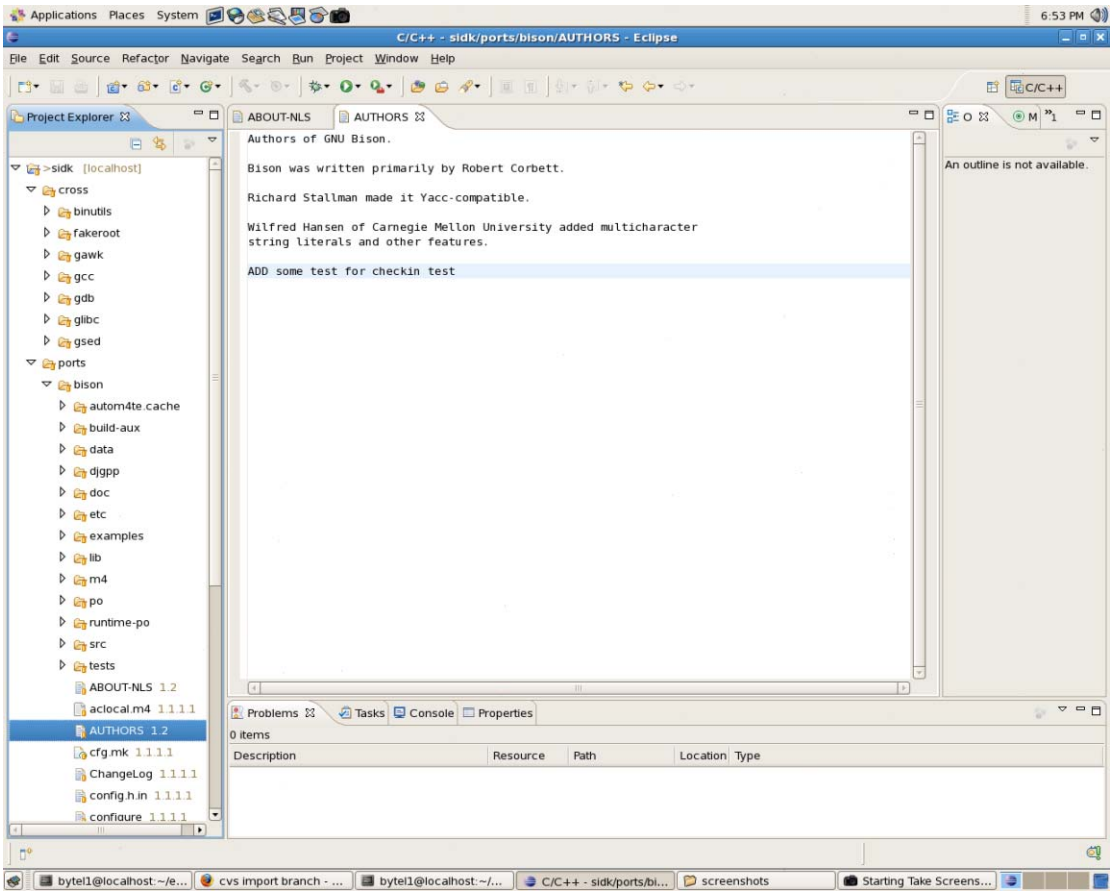
Checking in a Modified File

To check in a modified file:

1. Edit a file and apply a comment.

   For example, the file sidk/port/bison/tests/AUTHORS has been modified. The following text is added:

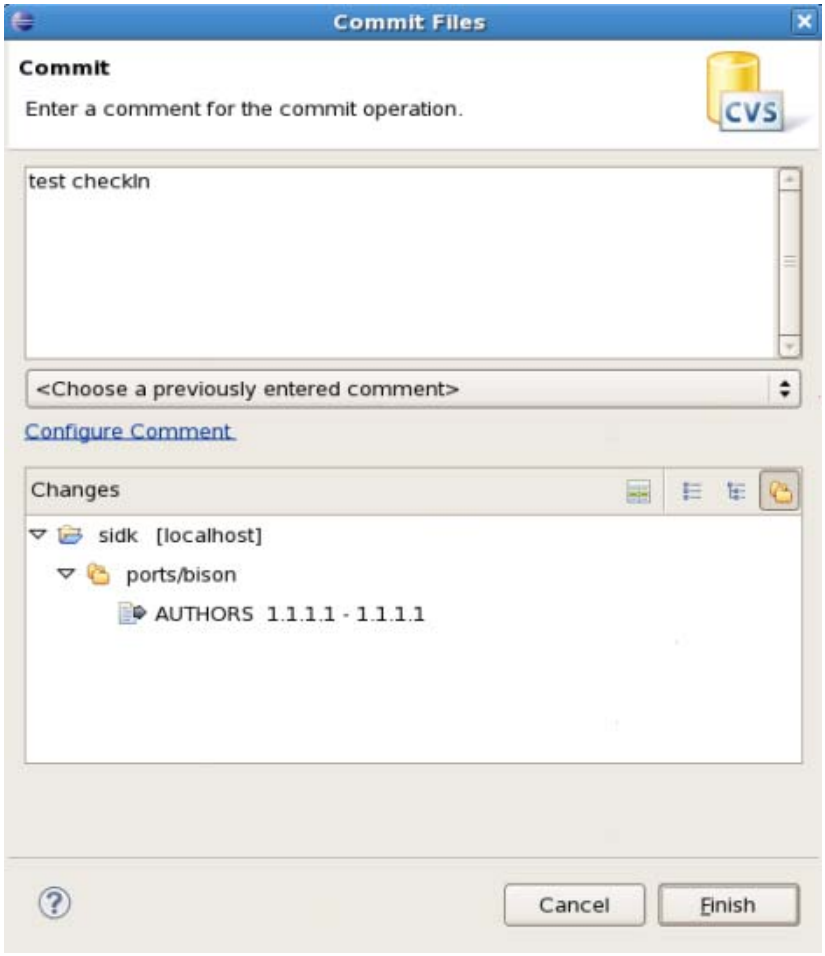
   ```
   ADD some test for checkin test.
   ```



2. Commit the change.
   a. On the **Project Explorer** panel, select the file you have edited.
   b. Right-click and select **Team**.

c. Select **Commit**.

The **Commit Files** window appears.



d. Enter a comment on the change applied.

3. Click **Finish**.

The revision number changes from 1.1.1.1 to 1.2.

CHAPTER 5

Upgrading the Pace SIDK

Whenever a new version of SIDK is available, Pace provides a link to download the new SIDK package. The package must be downloaded, imported, and checked-in on the CVS server.

The package includes:

- *sidk-common.tar.bz2*. All common Pace features.
- *sidk-xen.tar.bz2*. Rules to build for the xen virtual machine.
- *cvs-import.sh*. A script that aids the CVS import process.
- *cvswrappers*. Used by the cvs-import.sh script.
- *SIDK Installation Guide*. A startup document for installing and setting up the Pace SIDK.
- *SIDK Developer Guide*. A document for building the SIDK image using the Eclipse IDE.

The package may include additional tar files as the SIDK is able to build for other hardware platforms.

Upgrading the SIDK

To upgrade an earlier version of the SIDK, you need to perform upgrade procedures on the CVS server and client machines.

CVS Server

To upgrade on the CVS server:

- On the CVS server, run the "cvs-import.sh" script:

```
[–d <cvsroot>] [-b <branch>] –c <common> <addon> …
[user@sidk-package ~]$ bash ./cvs-import.sh –d /var/cvsroot –c ./sidk-
common.tar.bz2  sidk-xen.tar.bz2
```

CVS Client

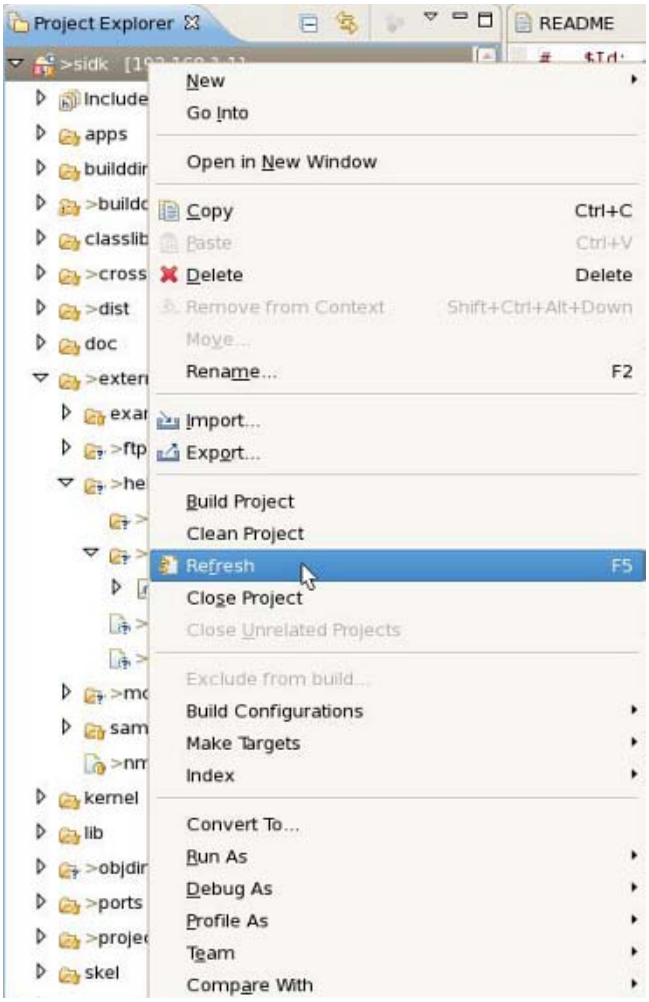
To upgrade on the CVS client:

1. On the CVS client, delete all builddir.* directories contained in the SIDK workspace directory:

   ```
   [user@sidk-workspace ~]$ rm –rf builddir.*
   ```

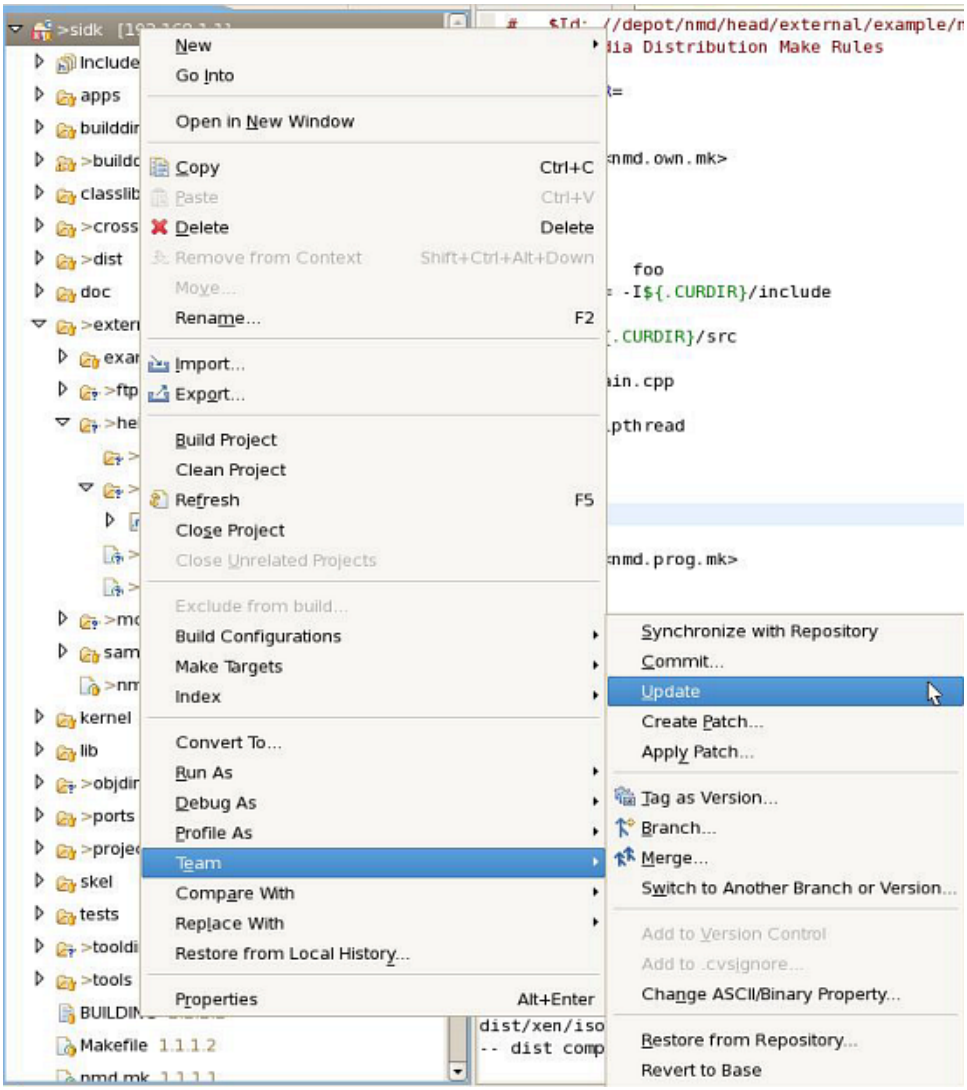
2. Start the sidk workspace on Eclipse.
3. On the **Project Explorer** panel, click the **sidk** project.

4. Right-click and select **Refresh**.



5. On the **Project Explorer** panel, click the **sidk** project.
6. Right-click and select **Team**.

7. Select **Update**.



After the update process is successful, you can start rebuilding the SIDK.