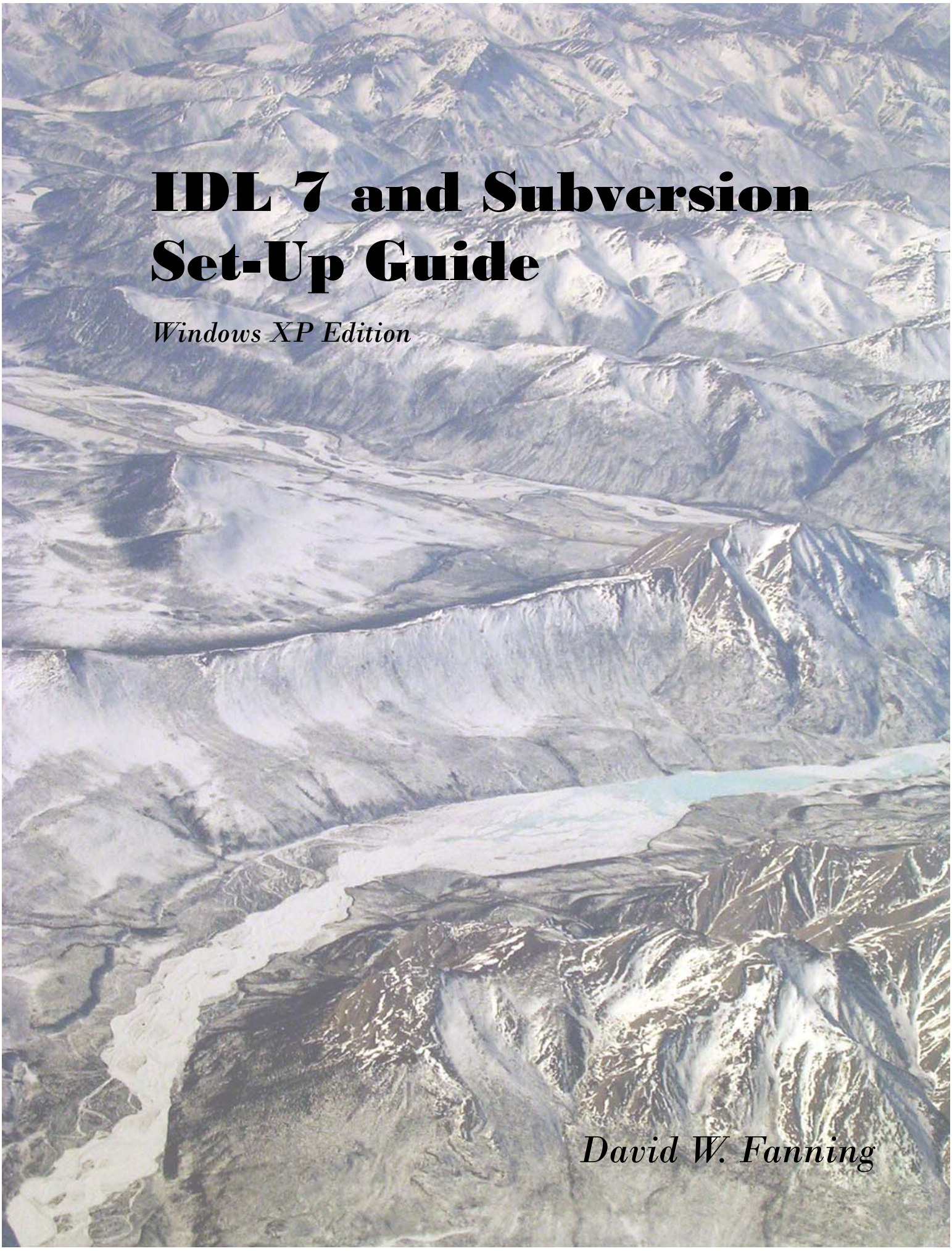


IDL 7 and Subversion Set-Up Guide

Windows XP Edition

David W. Fanning

IDL 7 and Subversion Set-Up Guide
Windows XP Edition

David W. Fanning, Ph.D.

Printed in the United States of America.

Published by Fanning Software Consulting, Inc.
1645 Sheely Drive
Fort Collins, CO 80526
Phone: 970-221-0438
Fax: 970-221-0438
E-Mail: david@dfanning.com
Internet Address: http://www.dfanning.com/

Printing History

December 2007: First Printing

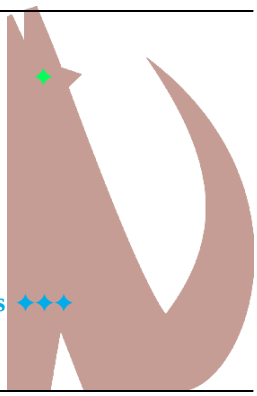
Cover photograph of Brooks Range, Alaska by David Fanning.

ISBN 978-0-9662383-3-4

December 2007

Set-Up Guide

✦ Discovering the Possibilities ✦✦✦



Setting up IDL 7 with Subversion

Getting Started

One of the purported advantages of the new Eclipse-based IDL Workbench, introduced in IDL 7, is that it can work together with plug-ins for other software to enhance your working experience with IDL. One obvious synergetic match-up would be to have IDL working together with a software version control system. Subversion is one such system that has received a great many favorable comments from programmers. It is an Open Source project and freely available.

The purpose of this Guide is to take you though the steps necessary to set-up and install a Subversion version control system on a Windows XP computer and get it working with the IDL 7 Workbench for a single-user of IDL. The steps described here are not the *only* way this can be done. Rather, it describes *one* way it can be done. In truth, it describes the way I would do it now, having benefited from nearly a week's worth of experience doing it in every possible wrong way. The goal here is to save you some time and frustration, and perhaps get you set up in a way that makes it easier to work with IDL and not harder.

Step 1: Install IDL 7

You may have already installed IDL 7 and spent some anxious moments trying to figure out how to organize your previous collection of IDL programs as Workbench projects. My own goal in doing this was to keep intact as much of my previous IDL software organization as possible. I am going to presume this is your goal, too.

My organization consists of a directory, *david*, which contains an assortment of IDL programs. This is my "home" directory and the place where I work in IDL if nothing special is going on. Inside of this directory, I have a number of other directories that contain IDL code organized for a particular purpose. For example, I have three directories that contain "library" code of the IDL libraries I use frequently: *coyote*, *jhuapl*, and *nasa*. Normally, these three libraries, at least, are on my IDL path at all times. In addition, I have directories that pertain to software development projects of my own: *catalyst* and *activecontour*. And I have other directories that pertain to software development projects for clients. Normally, these are stored in a directory named *clients*. I have perhaps 10-12 directories or so at any one time. I wish to maintain most of the IDL software I write (but not all the IDL software I use) under version control.

The IDL directory structure I was using prior to installing IDL 7, looked similar to the directory structure you see in Figure 1, below, although it was stored in another place. I copied it from there to the C drive and put it into a folder named *david*..

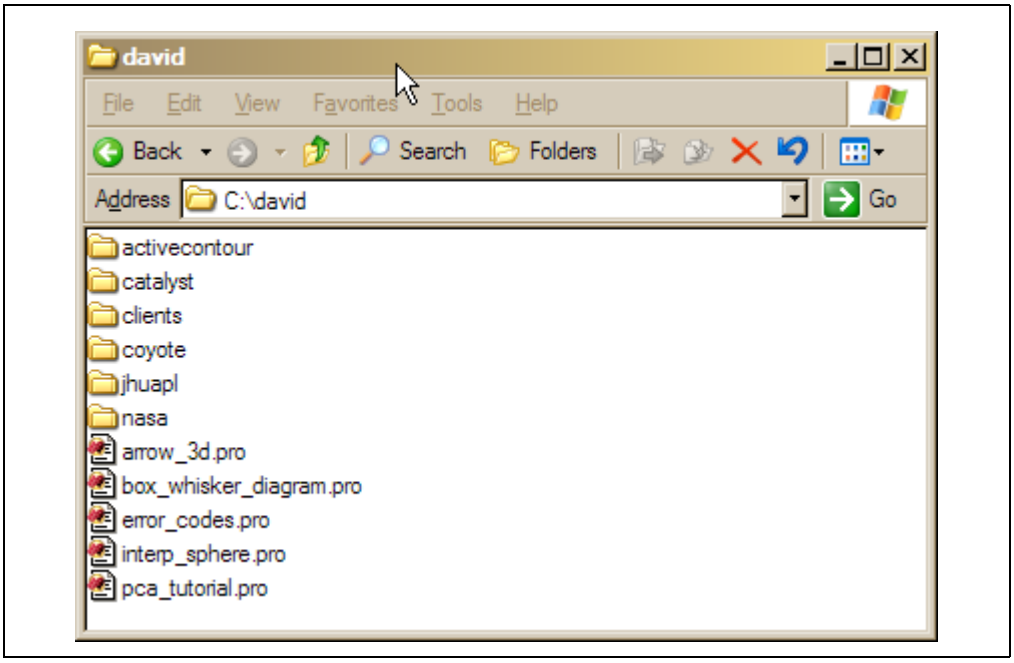


Figure 1: The directory structure as it looked prior to installing IDL 7.

I encourage you to work with a copy of your IDL program structure because the possibility that everything will work perfectly here, even with these instructions, is—in my experience—not that high. Having a backup to fall back on will save several years worth of gray hairs, at a minimum. Plus, you may be like me and find that your ad-hoc IDL directory structure can use a little pruning here and there anyway. This gives you the perfect opportunity to make your directory structure better than it was before.

If we incorporate this duplicate directory structure successfully into IDL 7, we will be able to make it the default IDL Workspace, or we will be able to move our new directory structure over to the old Workspace. Either way, there is no risk in working on the backup.

If you are just now installing IDL 7, then when it asks you where you want to install your Workspace think carefully. This is going to be the place where you can store your IDL project files. A “project” in Eclipse-speak is a group of files, normally IDL files (but you can include data and other resource files as well), that are all related in a particular file hierarchy under the project folder. This is not always the same as where you keep all your IDL-related files, although it could be. The IDL project hierarchy is shallow. It is not possible to create IDL projects in other IDL project directories. (Although individual projects can have complicated and deep file hierarchies, they cannot have other projects inside them.) So keep this in mind as you organize your new directory. I show you a fairly simple hierarchy in this example. Yours might be more complicated. Projects do not have to reside inside the IDL workspace folder, although they often do, and I have found that life is often easier if they do.

I will show you later how to create IDL project folders. These are often just created with your Windows Explorer, or they can be created from within IDL itself. The latter is often the best way if you are working with the Subversion version control system. I will show you how to create folders both ways in this Guide.

When I installed IDL, I found the default Workspace was *C:\Documents and Settings\David\IDLWorkspace.* But I'm old school and don't work much in the *Documents and Settings* folder, so I preferred to locate the Workspace somewhere else. You can choose to locate it wherever you like. If you make a bad choice, don't worry about it. You can change the location later, or even create another Workspace. You can have as many Workspaces as you like. I could have chosen, as I did in this example, to have *C:\david* be my IDL Workspace.

OK, go ahead and install IDL 7 now. Choose your Workspace to be the directory you just created for working with this Guide (your copy, not the original). Be sure to license IDL 7, too. If you have already installed IDL 7 and your Workspace is already configured, just read on.

If you have already installed IDL 7, and you chose your Workspace as something else, no problem. We will just create a new IDL Workspace that points to the duplicate directory structure you just set up. Go into the *File* menu on the IDL Workbench and select the *Switch Workspace -> Other* option and browse to the top of the new directory structure (in my case *C:\david*). Select that as your new IDL Workspace. The IDL Workbench should restart with this as the new Workspace. You should see something that looks similar to Figure 2, below.

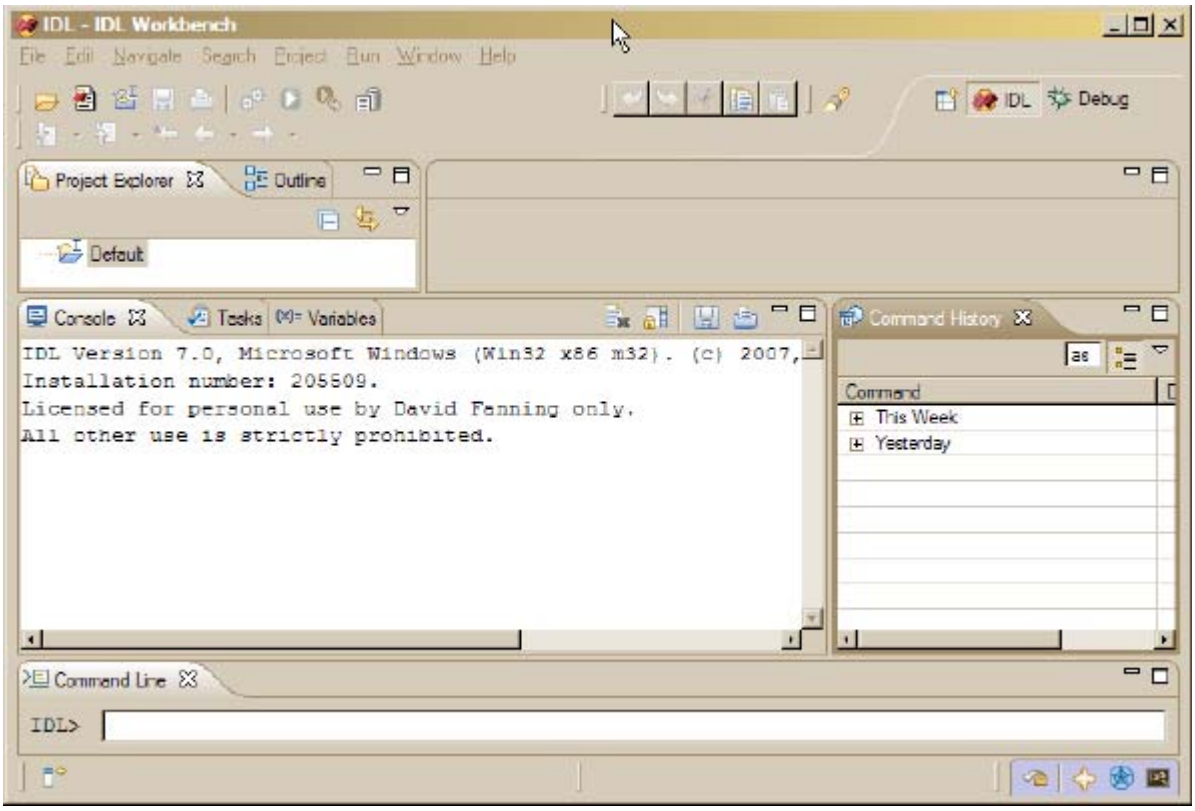


Figure 2: The IDL Workbench, now pointing to the new IDL Workspace.

In the Project Explorer window, you will see a "Default" project. I am going to give that a different name. My sense of the IDL files currently in the *david* directory is that these are the files I play around with in IDL. I hardly ever want these files under a version control system. This is where I explore ideas, test answers to IDL newsgroup questions, work out problems, etc. Once I have something I like, these finished files go into some other folder (such as *coyote)*, where I do want them under version

control. I will want to move all these files into my default project in my IDL Workbench. Thus, I am going to name my *Default* project, *The Incubator.* You can name yours whatever makes sense to you. You see what my folder looks like after I have done this in Figure 3. Notice that IDL has inserted an additional folder, named

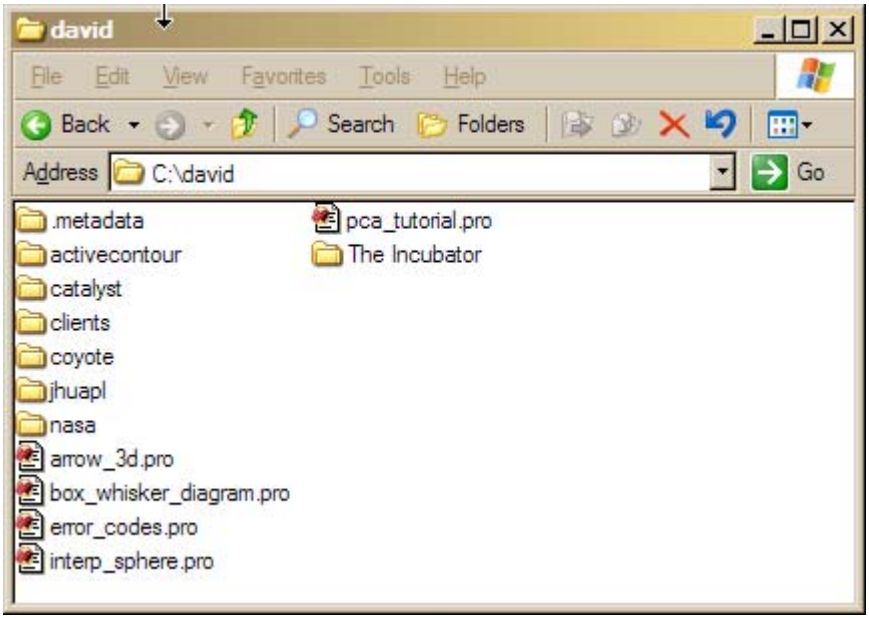


Figure 3: The directory structure after making the IDL Workspace and renaming the Default project to The Incubator. Notice the .metadata folder that was added by IDL.

.metadata, into the *david* directory. This contains information IDL 7 needs to manage the Workspace.

The next thing I do is simply select and drag all the IDL files currently in *david* into *The Incubator* folder. This is the equivalent to adding them to *The Incubator* project. My Project Explorer window now looks like Figure 4.

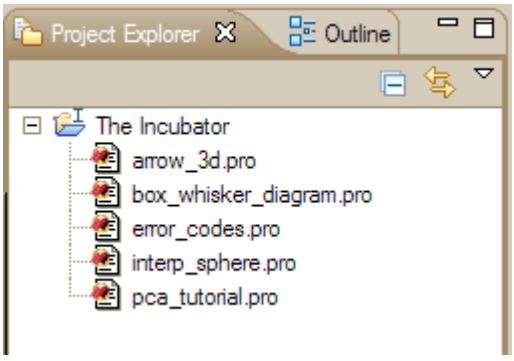


Figure 4: The Project Explorer window after the IDL files have been dragged into The Incubator folder.

Notice now that if you switched from a previous Workspace that under *File -> Switch Workspace* you can easily go back to your previous, untouched Workspace if and when you prefer to work there.

If you are the sort of IDL programmer who chooses to ignore good advice, and you have decided to follow my directions *without* making a duplicate copy of your directory structure, then I state again that I am taking no responsibility for lost files. I hope you at least had the good sense to backup your files to a safe place.

Step 2: Install the Subversion Software

Subversion is Open Source software and can be freely downloaded. I went to this page to download the software:

```
http://subversion.tigris.org/servlets/ProjectDocu-
   mentList?folderID=91
```

I selected the *svn-1.4.5-setup.exe* option. It contained a Windows installer with the basic *win32* binaries for running Subversion. I simply downloaded the file, ran it, and installed the Subversion package in the usual default location on my C drive

I am not going to describe how to use Subversion with an Apache server in this Guide. Rather, I am going to describe how to install Subversion as a file-system repository on your own computer. If you want to set Subversion up to run on the Apache server, I found these instructions to be excellent:

```
http://a-simian-mind.blogspot.com/2007/07/subversion-on-win-
   dows-with-remote.html
```

Subversion itself is normally run from a DOS command line. Personally, I find the syntax of the commands, and understanding the commands, rather cryptic. I am sure this is because I wasn't totally familiar with either Subversion or its predecessor, CVS, and because I am not fond of reading cryptic computer software documentation. In any case, I wanted some more intuitive way to work with Subversion, so I also downloaded and installed a Subversion front end, described in the next section.

Step 3: Install the TortoiseSVN Client

To make Subversion easier to set up and work with (I hate the DOS command line!) I also downloaded and installed TortoiseSVN from this location:

```
http://tortoisesvn.tigris.org/
```

TortoiseSVN is a subversion client, which is implemented as a Windows shell extension. It is also Open Source software and free for the downloading. What it does is add Subversion controls to the Windows Explorer. Since this is how I work with files, it feels completely natural to me to use this way of accessing Subversion. The file I downloaded was *TortoiseSVN-1.4.5.10425-win32-svn-1.4.5.msi.* I simply downloaded the file, opened it, and installed the software in the usual location on my C drive.

Once I have installed TortoiseSVN, if I right-click on a file, I will get additional TortoiseSVN functionality in the pop-up menu. You can see an example of what I mean in Figure 5.

TortoiseSVN is not strictly required, but I am going to use it in this Guide to create the Subversion repository. If you choose not to use TortoiseSVN, you will have to use the Subversion command line arguments to create the repository, and you will be on your own for that part of the set-up. I am *extremely* pleased with TortoiseSVN, however, and think you will be, too. I highly recommend you download it. I am certain you will find it useful.

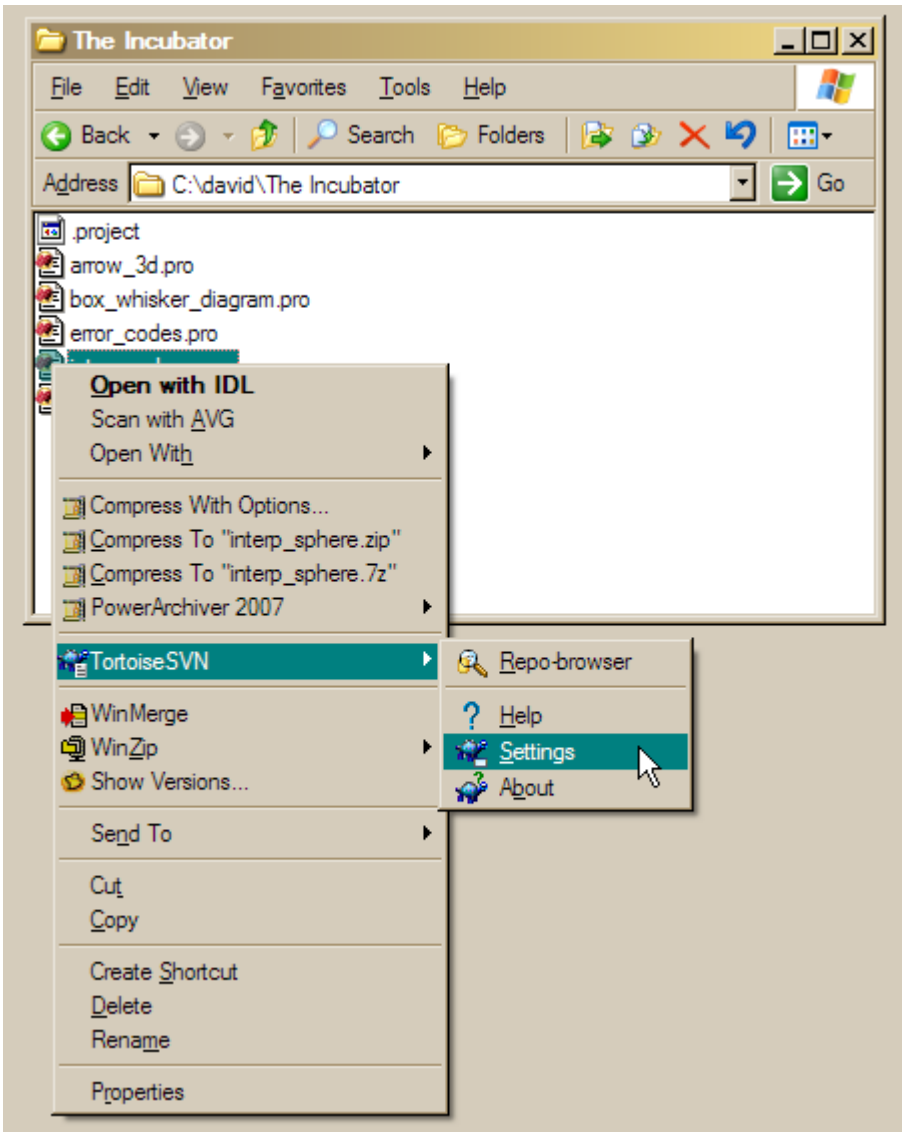


Figure 5: ***After TortoiseSVN installation, right-clicking any file gives access to TortoiseSVN functionality in the pop-up menu dialog.***

Step 4: Set up the Subversion Repository

The next step is to set up the Subversion repository. This is where the files you are going to put under version control will reside, along with the detailed information that describes changes you make to those files over time. To work with the files you put into the repository, you “check them out”, work with them, then “check them back in” when you are finished with your changes. Subversion is different from CVS in that it can work in a natural way by checking entire directories of files in and out easily.

It pays to give some thought to how you want to organize your repository, since it can sometimes be difficult to change the structure later. I am going to use a “standard” structure here, since it fits nicely with the way I currently work with IDL, but other ways work just as well. For the next few minutes, in the initial stages of setting the repository up, if you make a mistake, just delete the repository file and start over. I did this four or five times before I got the repository set up like I wanted it to be. The

penalty only comes down the road when you have recorded lots of file changes. At that point change is more difficult. Unless, of course, you don't care about retaining a record of the changes to the files, in which case you can always delete the repository, or just make a new one.

Make a Subversion Repository Directory

First, create a directory where your Subversion repository is going to be located. I would try to choose a fairly obscure location where it is out of the way, but where it is likely to get backed up with your daily or weekly scheduled backups. (You are doing regularly scheduled backups, aren't you?) I just installed my *SVNRepository* folder at the top level of my C drive. You can put it wherever you like. Just write down the location, because you are going to need it later when you want to let IDL know where it is. I think you can name the repository directory whatever name you like, but I named mine *SVNRepository*, with no spaces in the name. I don't know if the space is important or not, but in my experience command line software is not usually fond of spaces in names, and I want to be ultra conservative here.

Configure the Repository Directory

Now, right-click on the file you just created (in my case *SVNRepository*), find the TortoiseSVN selection in the pop-up menu, pull that out, and find the *Create Repository Here...* menu selection. You will be asked to make a choice as to the type of repository you want to create. Almost certainly this is "native file system (FSFS)". The Berkeley Database file system is much older and not as powerful. Since we are just setting this up, let's choose the best and go with the default.

If you use your Windows Explorer to browse the repository directory now, you will see a number of folders, a *README.txt* file, etc. Feel free to browse around, but you will not be mucking around with anything in this directory directly from now on. Rather, you will be interacting with this repository either though the TortoiseSVN interface or, in a few minutes, through the Subclipse interface that we will install in IDL 7.

Set Up the Repository Structure

The next thing you need to do is set up the basic repository structure. This will consist of three folders at the top level of the repository, named *trunk*, *branches*, and *tags*. It is *not* possible to just create these folders in the repository directory directly from the Windows Explorer. Rather, it mush be done in a fairly odd way, it seems to me. Here is now I did it.

I used the Windows Explorer to go to the temporary directory in my C drive. My temporary directory is named *temp*. There, I created a new folder, which I named, inventively enough, *new*. Inside the *new* folder, I created three other new folders, named *trunk*, *branches*, and *tags*. My folders looked like those in Figure 6.

Once this is finished, right-click on the *new* folder, find the TortoiseSVN button, pull that out, and find and select the *Import...* selection. You will be asked to supply the URL of the repository. Since we are using a file system, not a server, the URL will point to a file. It will look something like this:

```
file:///C:/SVNRepository
```

There will be *three* slashes in front of the C. The rest of the name will be up to you. You can also click the little button beside the URL filename to browse for the right place. Once you have it, click the *OK* button. The three folders inside of the *new* folder will be added to the repository.

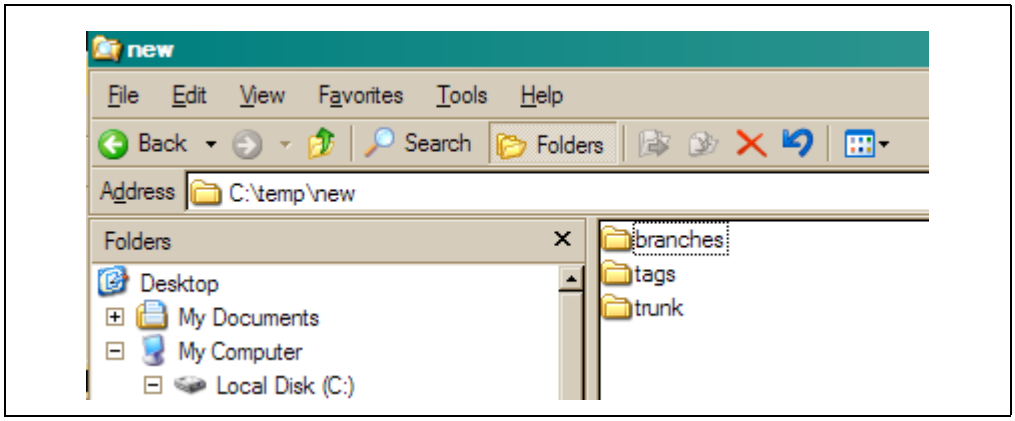


Figure 6: A new folder containing branches, tags, and trunk folders is created in the temp directory.

To see that this has been done correctly, navigate back to your repository folder (*SVNRepository*, in my case) and right-click to find the TortoiseSVN selections. This time, choose the *Repo-browser* selection. You should see something similar to Figure 7.

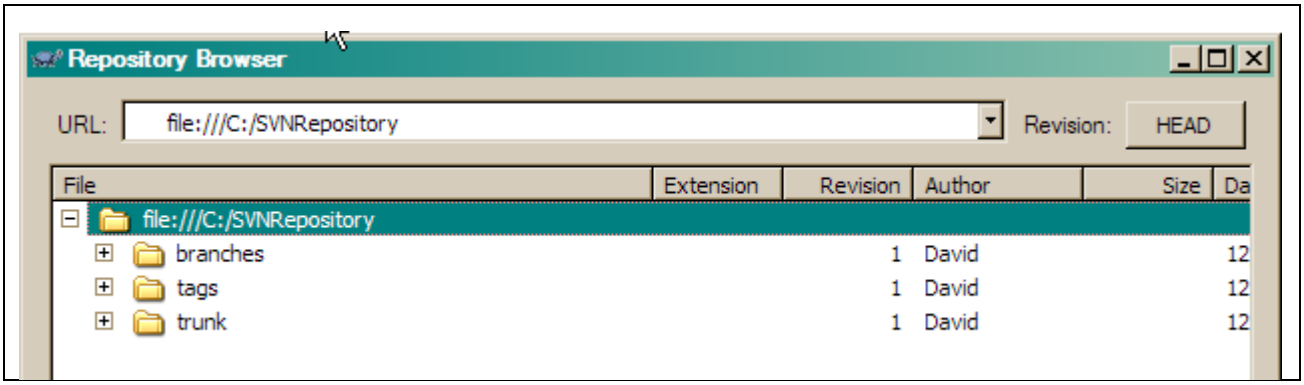


Figure 7: The basic repository structure imported into the repository from the new folder.

Copy Files into the Repository

The next step is to copy the files you want to have under version control into the repository. I am going to show you how to do this using TortoiseSVN now. Later, I will show you how to do the same thing using Subclipse from inside of IDL. There are various ways this can be done, and I want you to be aware of them.

This step, I am warning you, can go badly wrong. Not because of problems with Subversion or TortoiseSVN, but because of inadvertent human error. I *strongly* urge you to make copies of all the directories containing files that you want to add to the repository. (Which you will have already done if you are working with a duplicate directory structure.) You could copy them, for example, to the *temp* directory, where they can be deleted once we know things have gone right.

If you make a mistake placing these files, it is much easier to just delete the repository file and start over with this section, then to try to move files you have already placed into the repository. If you do make a mistake, rest assured, you are not the first programmer in the world to do so.

OK, so in my duplicate directory structure, I want to put all the IDL **.pro* files in the *coyote* directory under Subversion control. Choose a directory in your file structure you would like to add to Subversion control.

Before you import the files into the repository, however, this is a *very* good time to look through your folders for files you do *not* want to have under version control. These might include backup files, binary files, junk files, etc. Clean them up. Once they have been transferred to the repository, everything is harder to do than it is now. Seriously, do it now. It is *considerably* more trouble later.

Ok, I have cleaned up the files in *C:\coyote* so that only the files I want to include in the repository are there. I now navigate to that file with Windows Explorer and right-click to find the TortoiseSVN selections, and choose *Import* .... Be *very* careful where you put the files when you specify the URL. You want the files to go into the *trunk* folder, in a folder named *coyote*. You will be able to browse to your SVNRepository directory, but you will probably have to add the trunk and coyote directories to the end of the URL name, as in Figure 8. Be sure you specify the *exact* location where you want it to go. Please check your spelling before you hit the *OK* button. If you make a mistake, delete the repository directory and start over. *Much* easier!

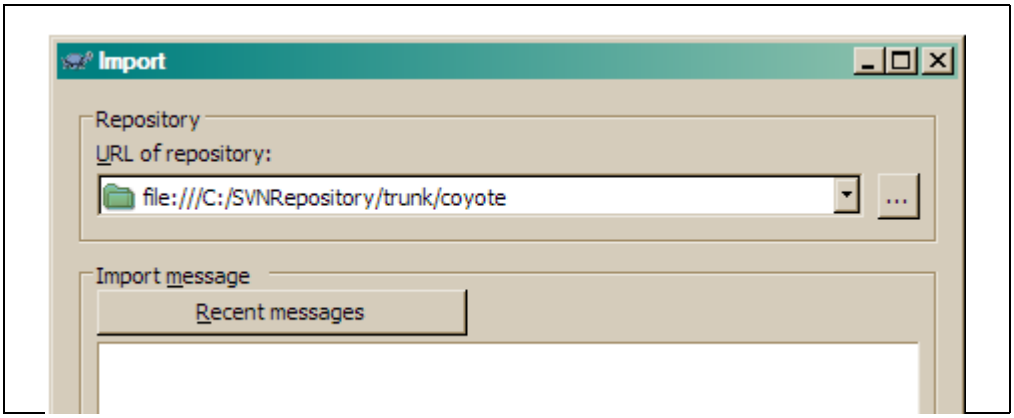


Figure 8: ***Be very careful to put the files in the right place. Start with the repository location, but make sure you put the files in the trunk/coyote directory. You may have to add the words "/trunk/coyote" to the end of the URL.***

Do this for as many directories as you care to add.

You see what my repository looks after adding the *coyote* and *activecontour* directories to the repository in Figure 9.

Step 5: Install the Subclipse Plug-in for IDL

The next step is to install a Subversion plug-in into the Eclipse IDLDE that the IDL Workbench is based on. There are several of these, but the one I am going to use is named *Subclipse*. Subclipse is similar to TortoiseSVN in that it is a front-end for Subversion functionality. In this case, I think it is not quite as easy to use as TortoiseSVN, at least with respect to organizing the repository directory structure, so this is why I have used TortoiseSVN to add the files to the repository first.

In addition to the instructions for installing Subclipse in this Guide, you can also find good instructions for installing the Subclipse plug-in here:

`http://subclipse.tigris.org/install.html`

Here is how to do it. In the IDL Workbench, find the *Help -> Software Updates -> Find and Install...* menu item. In the ensuing dialog, choose the *Search for new feature*

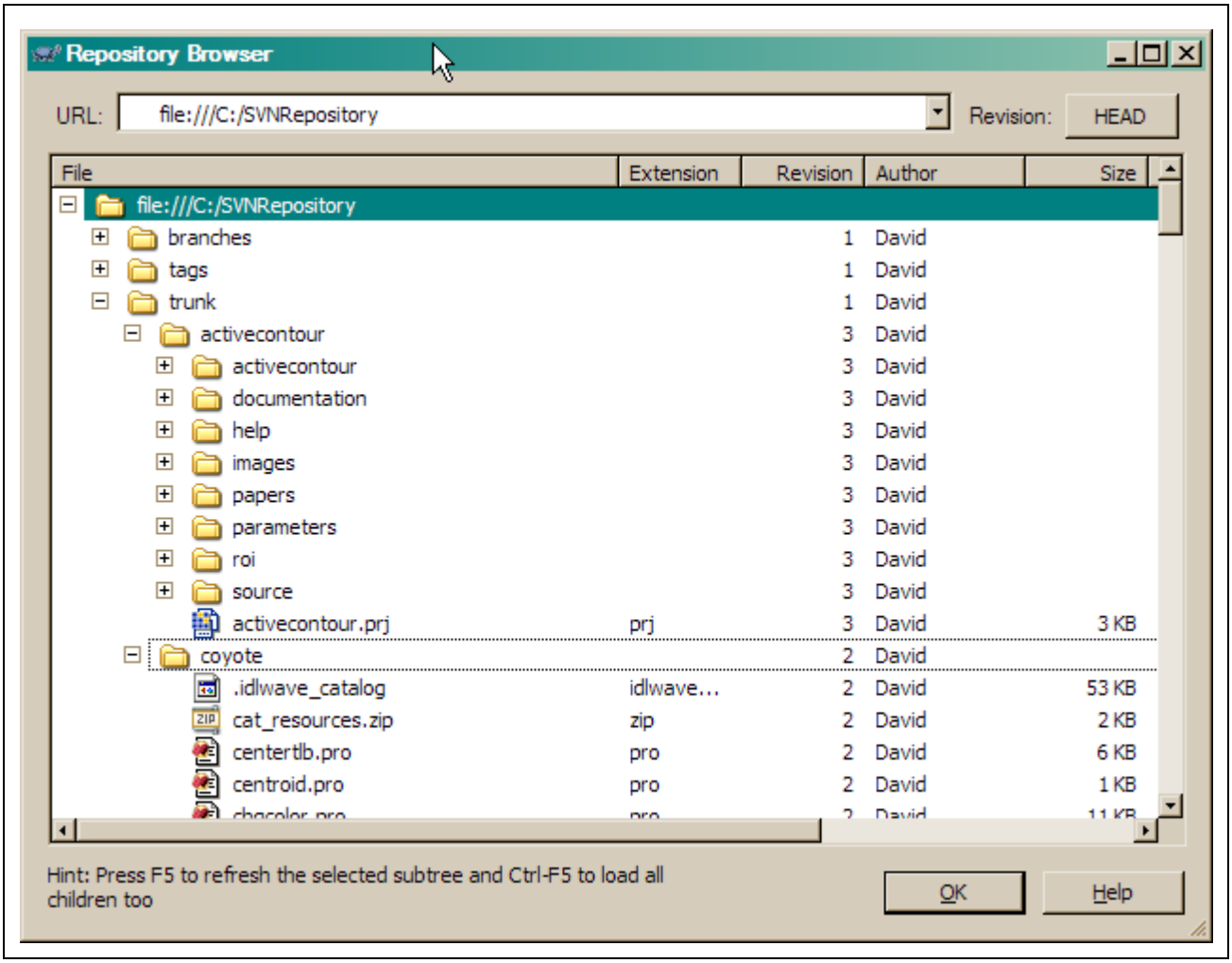


Figure 9: This is what my repository looks like after adding files from the coyote and activecontour directories to the repository.

to install selection and hit the *Next* button. On the subsequent dialog choose the *New Remote Site...* button. Enter *Subclipse* for the Name, and *http://subclipse/tigris.org/update_1.0.x* for the URL, as shown in Figure 10, and click *OK*.

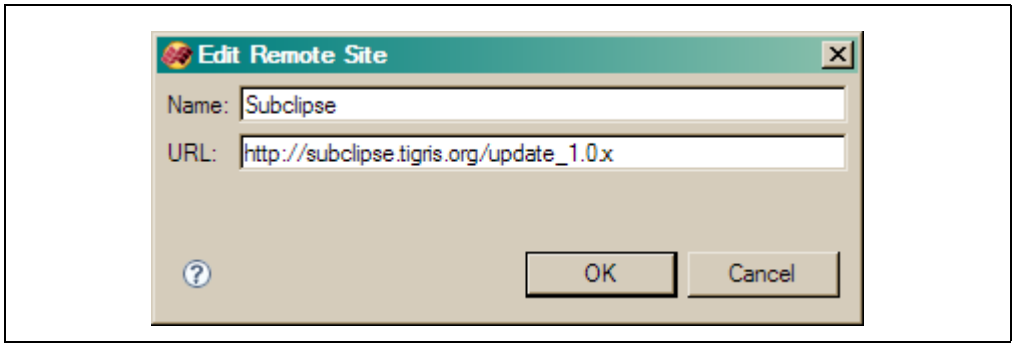


Figure 10: The New Update Site dialog.

When you finish, select the Subclipse site, by putting a checkmark in the box by its name, and click the *Finish* button. IDL will download the feature to install.

Next, you will see an Updates dialog, as in Figure 11.

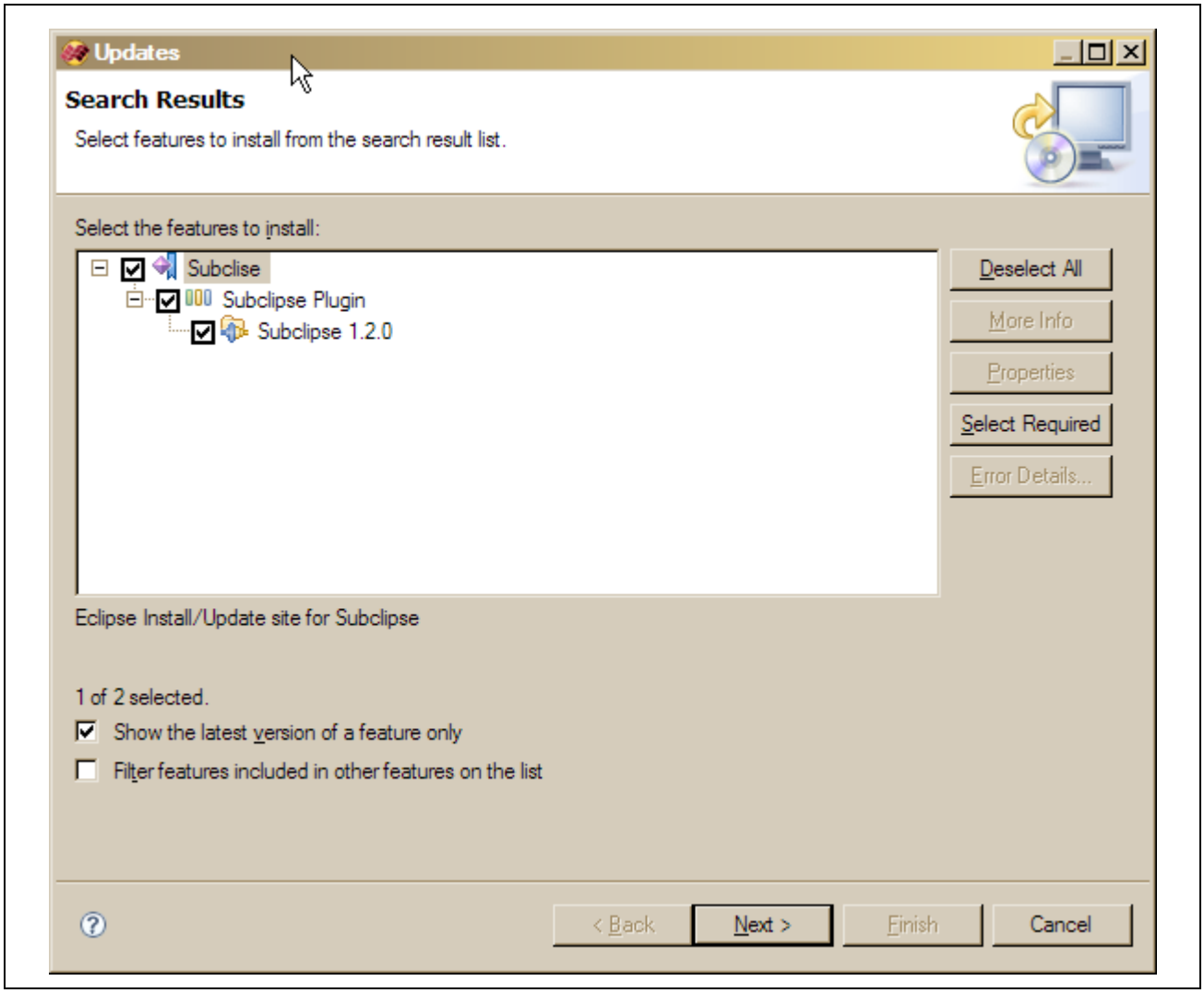


Figure 11: Select the features you want to install in this portion of the Wizard.

Simply click the *Next* button, accept the license agreement on the next display, hit the *Finish* button, and IDL will download and install the plug-in software. This will take several minutes. Once the features are installed, you will have to restart the IDL Workbench for the features to appear in the Workbench.

Don't worry if you don't *see* any features yet, they are not obvious. You will see how to access them in the next few steps.

Step 6: Connect IDL to the Repository

The next step is to connect IDL to the Subversion repository. Under the *File* menu, find the *New -> Other* selection. You should see a dialog similar to the one in Figure 12.

Select the *Checkout Projects from SVN* selection and click the *Next* button. Select the *Create a New Repository* button, and click *Next*. You will be asked to supply the URL to the repository.

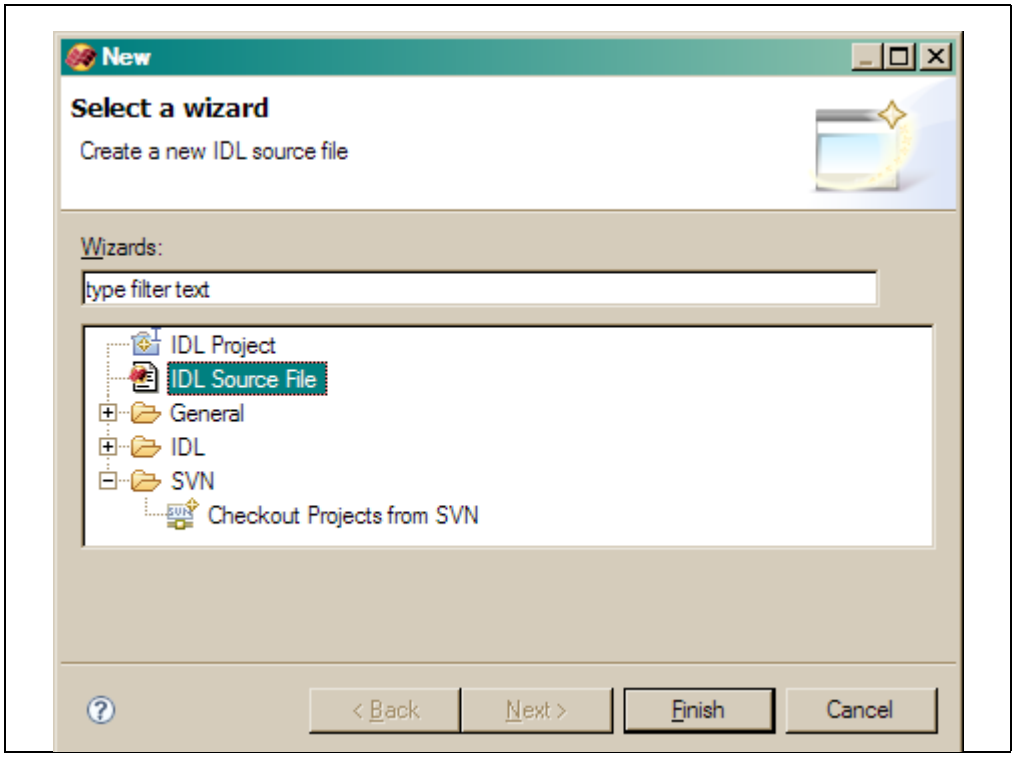


Figure 12: To connect IDL to the Subversion repository, find the File->New->Other dialog, and open the SVN folder. You are looking for the button that says Checkout Projects from SVN.

This is where you wish you had followed my advice and written it down. But it is something like this:

```
file:///C:/SVNRepository
```

If you have problems with this, and you get errors, you probably spelled something incorrectly. Check your work carefully. You need to get this correct. Cancel if you get confused and start over.

If you do this correctly, you will find yourself at the dialog in Figure 13. My advice to you now is to simply hit the *Cancel* button. Don't download files yet!

Step 7: Check Out Files into an IDL Project

This step may be the scariest of all. *Be absolutely sure* you have copies of all your files before you proceed. We need to copy the files from the repository back into our normal IDL file structure so we can work with the copies of the files, and not the originals. But, of course, what is in our normal IDL file structure at the moment *are* the original files. So, you may feel some concern when I tell you to *delete* all those files in your original IDL file structure that are now copied into the repository.

Yikes! Really!?

Yes, really. In my case, since everything in both the *coyote* and *activecontour* directories was copied into the repository, I can just delete both of those directories. But it is possible the directories you added to the repository contained files that you did not add to the repository, too. *These files will be deleted in the next step in the next step of this process*, so if you want to add them back to your file structure after we have finished checking files out of the repository, put them somewhere safe. (I could

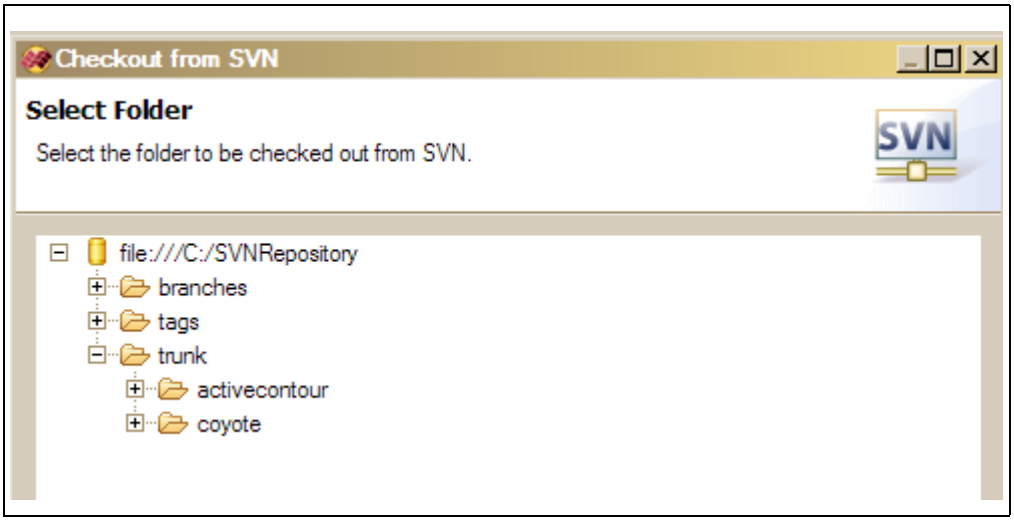


Figure 13: Ready to check out files into IDL. At this point, just hit Cancel. You will download files in a minute.

have moved the *coyote* and *activecontour* directories from the *C:\david* directory to the *C:\temp* directory, for example.)

The idea here is that we are going to set up directories *Coyote* and *ActiveContour* as IDL projects in the IDL Workspace. And we are going to check out the files we have already added to the repository into these projects. (We will do this again, in a different way, in just a minute.)

So, I have deleted the *contour* and *activecontour* directories in the *C:\david* directory. Do the same with your directories.

Now access the *File* menu, and find the *New -> Other* selection shown in Figure 12. This time when you select the *Checkout Projects from SVN* selection you can choose to use the *Existing Repository Location*. Navigate to the part of the repository you want to check out. In Figure 14, I am navigating to the coyote directory.

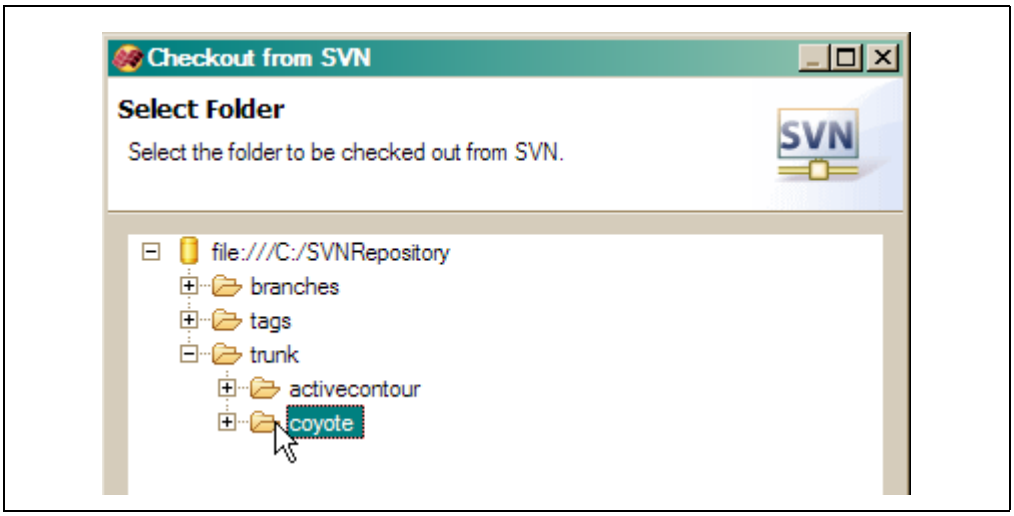


Figure 14: Checking out the coyote directory into an IDL project.

In the next dialog, choose the *Check out as a project in the workspace* selection, and name the project appropriately. In Figure 15, I have chosen the name *Coyote*.

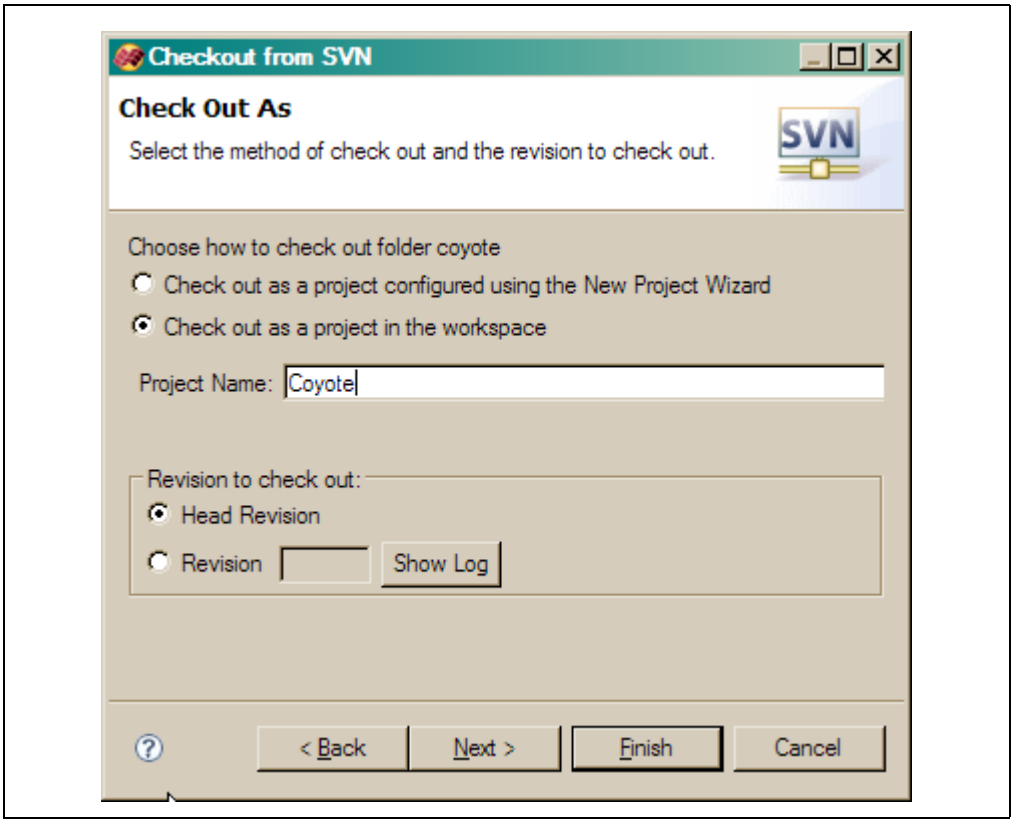


Figure 15: Chose to create the project in the workspace and give the project an appropriate name. Here I am naming the project Coyote.

When you hit the *Finish* button, a new project, named *Coyote* will be added to your Workspace (that is, inside the *C:\david* folder), and the files in the repository will be copied into that new directory. If this is done correctly, there should now be a Coyote project in the Project Explorer window of the IDL Workbench and it (and the icons for the files inside it) should be "decorated" in a way that you can tell at a glance that they are under Subversion control. You can see what my Project Explorer window looks like after adding the *Coyote* files in Figure 16.

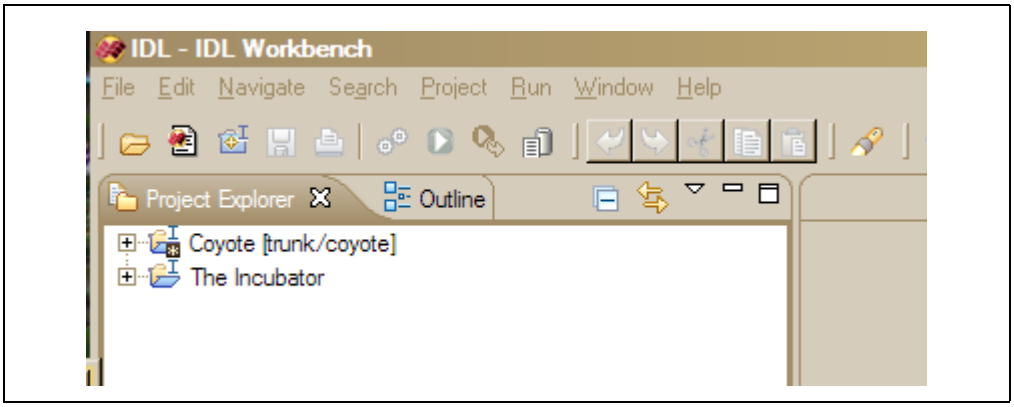


Figure 16: Files under Subversion control are "decorated" so they can be identified at a glance.

You work with the files under Subversion control by accessing the Subversion menu. You can find this by right-clicking on a project under Subversion control, and finding the Team pull-down menu, as in Figure 17.

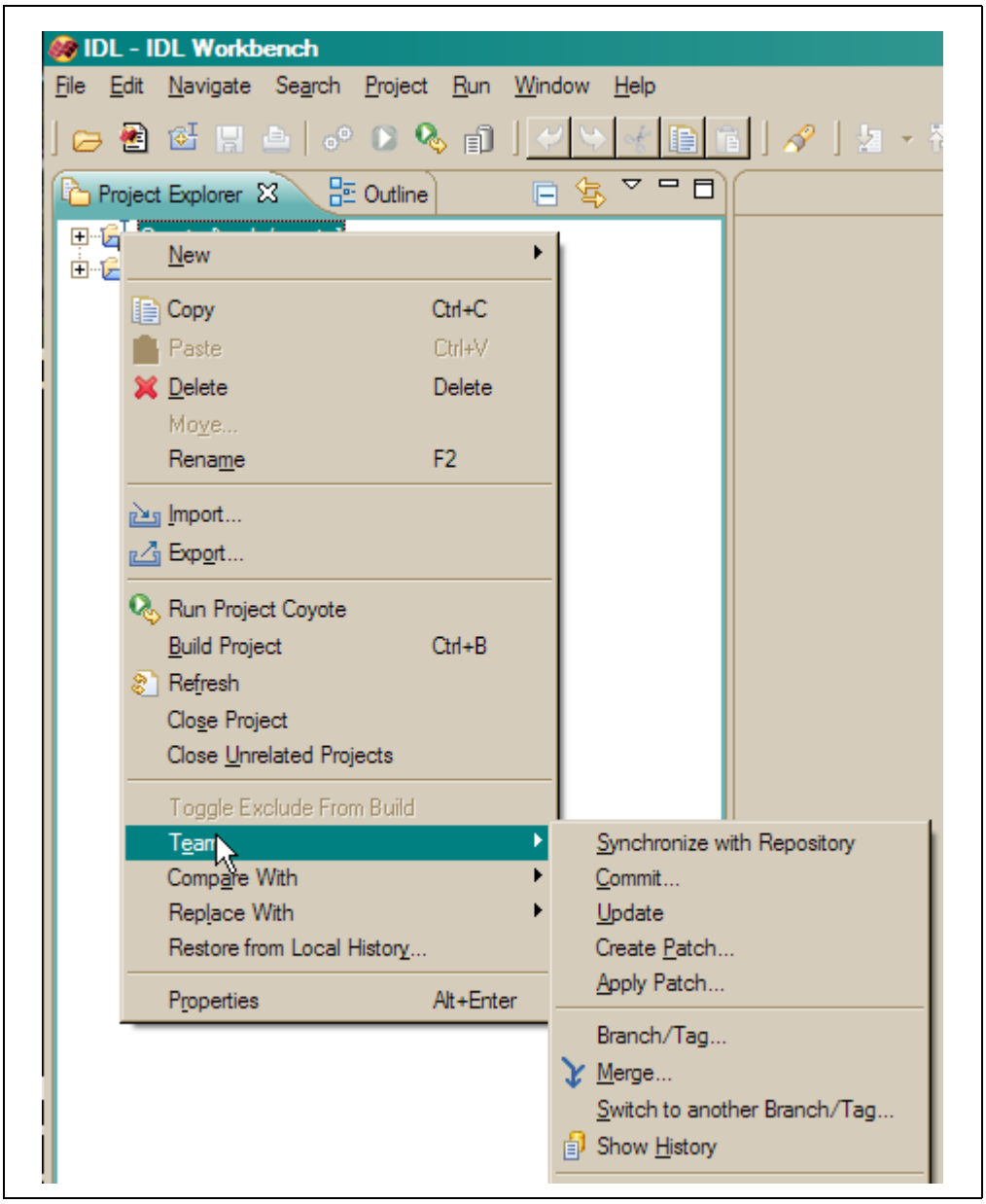


Figure 17: Access Subversion functionality by right-clicking on project under Subversion control and pulling down the Team menu. Only a partial menu listing is shown in this figure.

For example, if you had made changes to files in the project, you could choose the *Commit* button to check the files back into the repository and record a new version of the files.

Follow the same set of steps for all of the directories you have already added to the repository. Then I will show you how to add files that already exist in the IDL workspace to the repository from within IDL. When you are finished, you can add additional file to this project. (Perhaps files from the directories you just put over in the *temp* directory.) You will be able to add these files to Subversion or not, as you

choose. Any additional files you add will be seen in the IDL project and will be considered part of the IDL project. This is true even if the files are not managed by Subversion.

Step 8: Adding Files from the Workspace to the Repository

You may have files already in your IDL workspace that you would like to add to the repository. For example, in my *clients* folder, I have a folder named *BigBucks*, which relates to work I am doing for a deep-pocket client named Big Bucks Unlimited. You see the kinds of things I have in my file in Figure 18.

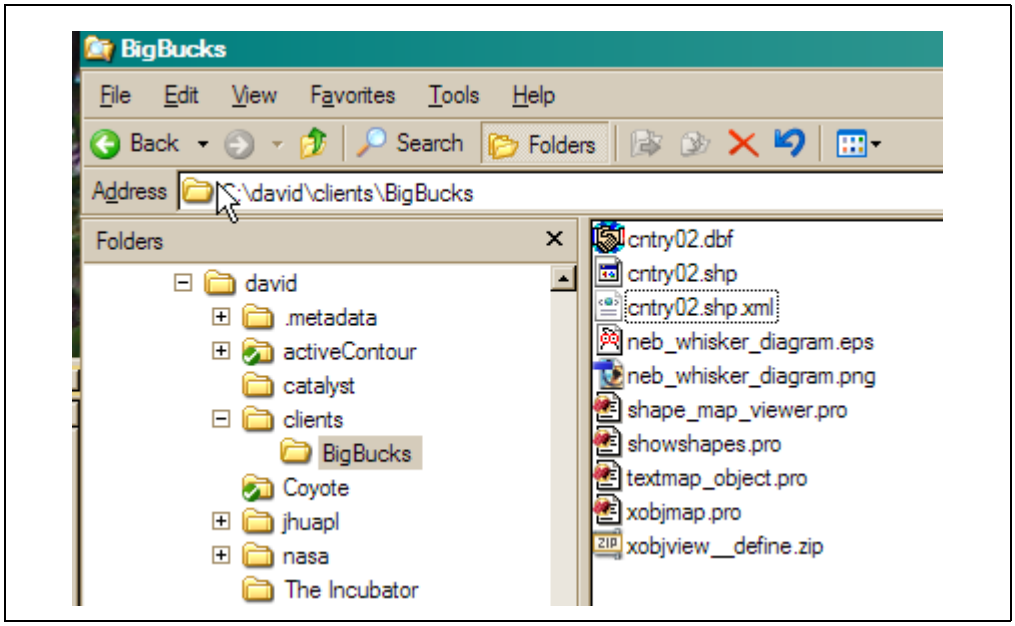


Figure 18: The files I have in clients/BigBucks from work I was doing in IDL 6.4. I want to work with these files in IDL 7, and I would like to put the files under Subversion control.

When I moved to IDL 7, I wanted to work with these files, and I wanted to add the IDL source files to Subversion control. The directory structure I have here is not as good as it could be to meet my goals. So while I was making the move, I decided to reorganize the file structure a little differently. I decided that I would move all the IDL files into a folder named *source* and all the other files into a folder named *resources*. This will make it easier for me to manage my IDL source files with Subversion. You see the revised file structure in Figure 19.

The first thing I need to do here is make the BigBucks files into an IDL project. From within the IDL Workbench, I choose the *File -> New -> IDL* Project selection. I gave the project the name *BigBucks* and selected the *Create the new project from an existing directory* option, as in Figure 20. Notice that I have the button selected to have IDL update the path preference when the file is opened or closed. I'll show you how to change that later in case you prefer to manage your paths directly and not have IDL do it for you when projects are opened or closed.

Click the *Finish* button to have IDL create the project from the existing directory. The project will show up in the Project Explorer window, as in Figure 21.

I might work with these files for some time like this. But, eventually I will get nervous about making changes to these files, or I will want to send a version of the files to the

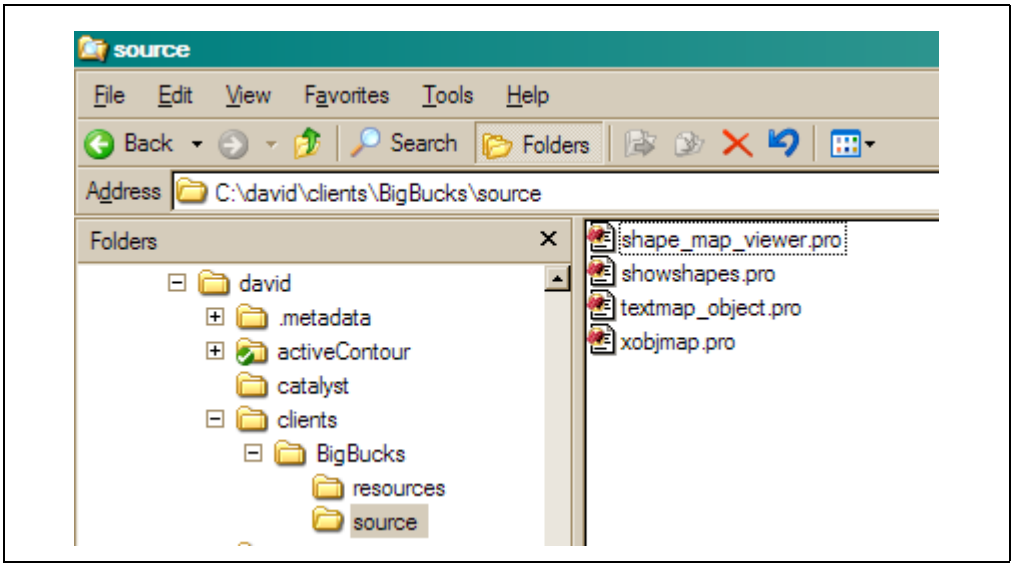


Figure 19: The revised BigBucks folder, with all my IDL source files in a single* source *folder and all the other files in a* resources *folder.

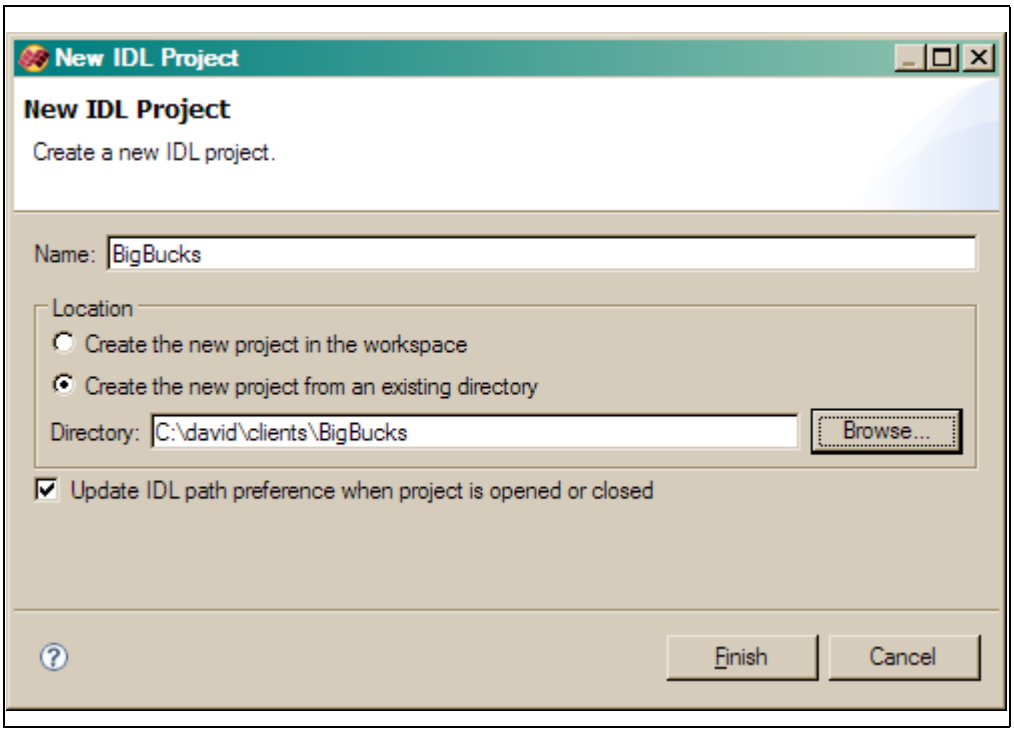


Figure 20: Creating the new BigBucks project from files that already exist somewhere on disk. I am allowing IDL to manage the path when this project is opened or closed.

client, and I will want to mark which ones I send, etc. In short, I will want to have them under version control.

This is how I would do that from within IDL, using the Subclipse front end for Subversion that I previously installed. This is pretty much the opposite of what I did in the previous step, where my files were already in the Subversion repository.

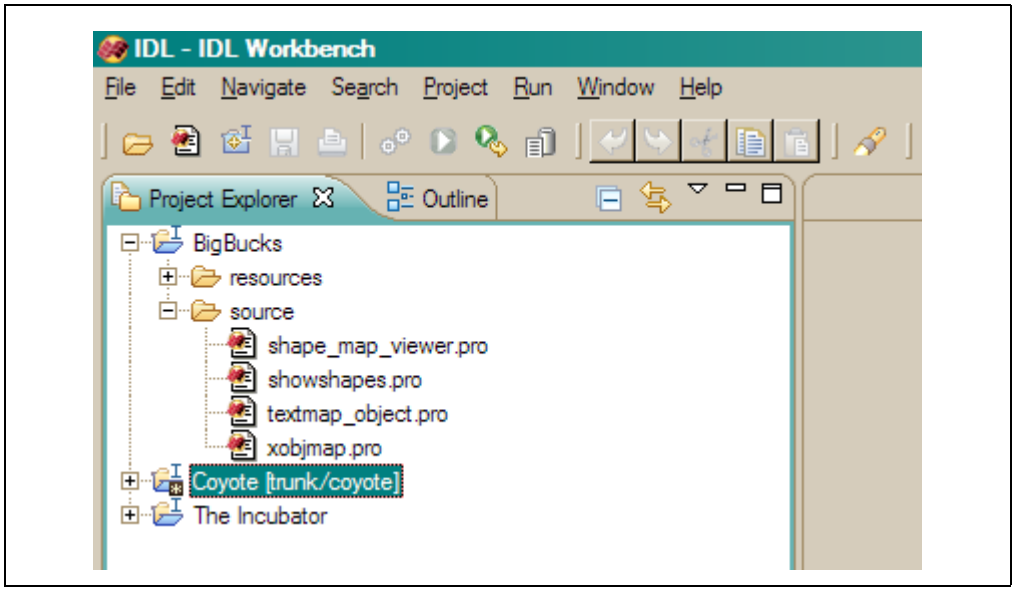


Figure 21: The BigBucks project is now in the Project Explorer window.

First, I need to create a directory in the repository where I want my files to live. I would like my repository files to more or less reflect my directory structure. In this case, I would like to create a folder in the repository named *.../truck/clients/bigbucks*.

To do so, I have to switch to the SVN Repository perspective. Under the *Window* menu in the IDL Workbench, chose the *Open Perspective-> Other* selection and choose *SNV Repository Exploring* choice, as in Figure 22.

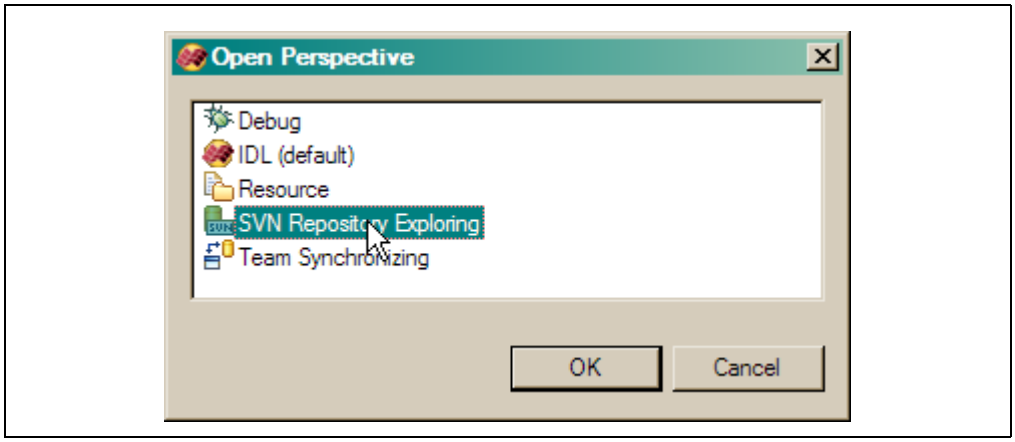


Figure 22: Open the SVN Repository perspective by choosing the Window->Open Perspective->Other selection and choosing the SVN Repository Exploring selection in this dialog.

Next, right-click anywhere in the SVN Repository window, and choose the *New -> New Remote Folder* option. This is how you make a new folder in the repository. Browse in this window to the *trunk* folder. You want to make a *clients* folder here, as in Figure 23. (It is not possible to make a *clients/bigbucks* folder directly. It must be done in two steps.) Click the *Finish* button to add the folder to the repository.

Now, do this again to add the *bigbucks* folder. Be sure you add the bigbucks folder to the clients folder you just created. You repository structure should look similar to that

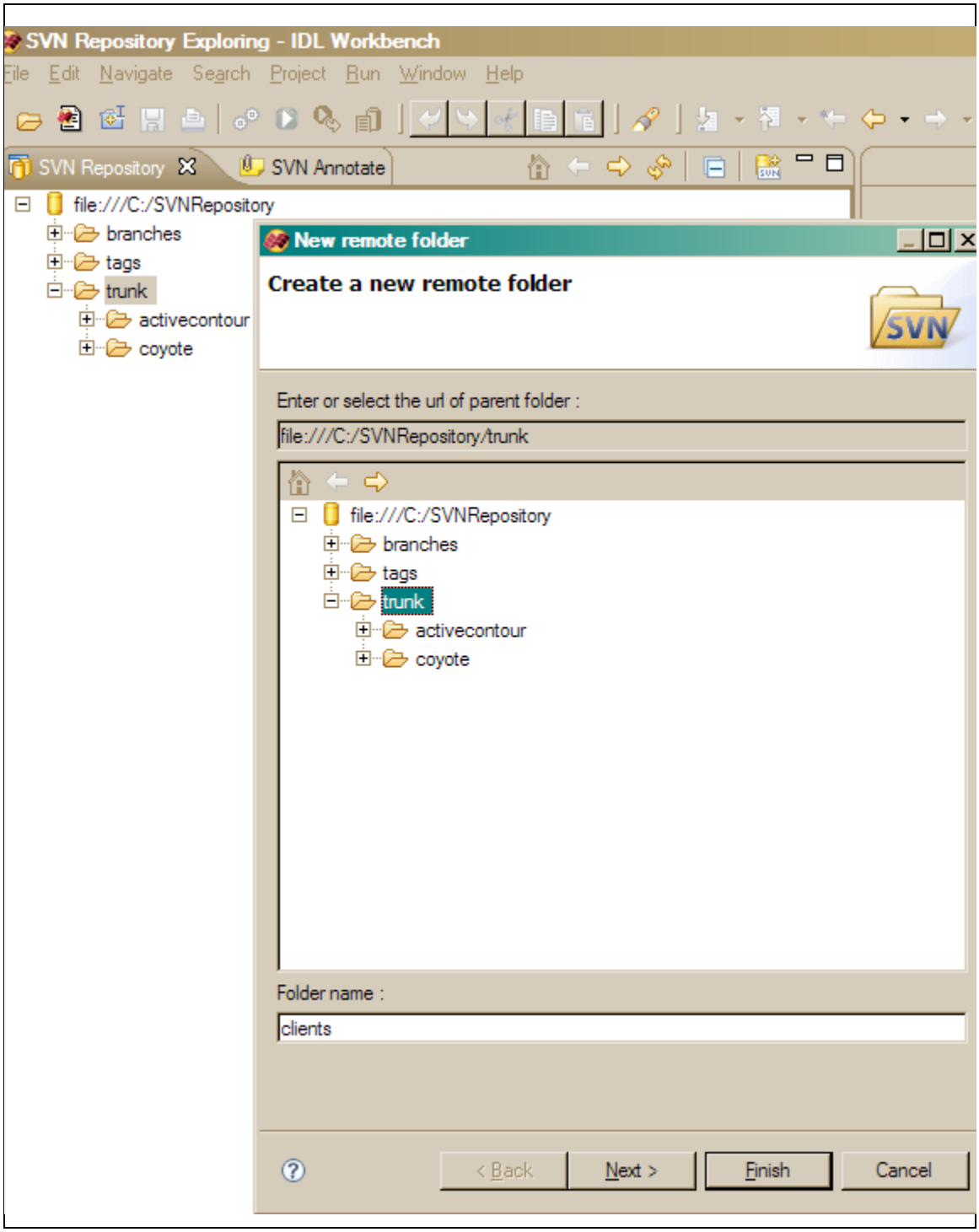


Figure 23: You add a new folder in the repository by choosing the New->New Remote Folder option after right-clicking in the SVN Repository window. Here we add a* clients *folder to the* trunk *folder.

in Figure 24. If you make a mistake, just right click on the part of the repository you would like to delete, choose the *Delete* option, and try it again.

Now that we have the correct directory created in the repository, we are ready to add the IDL source files to it. Switch back to the IDL perspective by choosing *Window -> Open Perspective -> Other -> IDL (default)*.

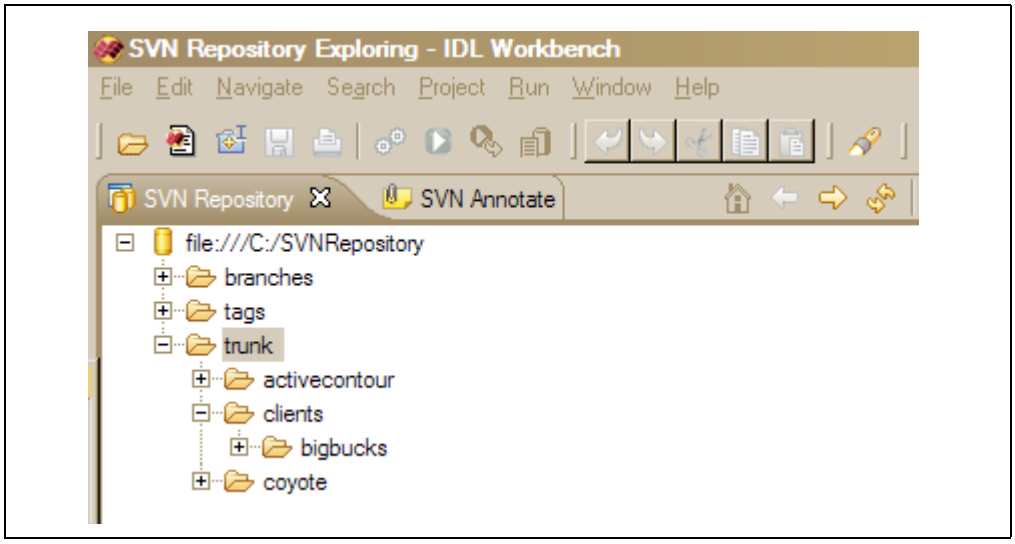


Figure 24: The SVN Repository view after adding* clients *and* bigbucks *folders in two steps.

Note that you do not have to change perspectives to see the SVN Repository Explorer window. In fact, it is often convenient to see this window in the IDL perspective. To do this, choose the *Window -> Show View -> Other* selection from the menu, and then find the *SVN* folder and the *SVN Repository* selection inside it. Selecting this and hitting the *OK* button will open the SVN Repository view in a window in the IDL perspective. (Probably in another tab down where the Console window is located.) This allows you to see both the repository and the Project Explorer window simultaneously.

To add the *BigBucks* project to the repository, navigate to the *bigbucks* folder in the SVN Repository view (probably just above the Command Line window at the bottom of your Workbench). Right-click and choose the *Checkout* option.

Unlike we did before, this time choose the *Check out as a project configured using the New Project Wizard* option, and choose *Finish*. The New Project Wizard window will open. Navigate to the *IDL* folder and chose the *IDL Project* selection, as in Figure 25. Click the *Next* button.

In this dialog, give the project the name *BigBucks*, and choose to create the project from an existing directory. Use the *Browse* button to select the *BigBucks* directory that exists in *C:\david\clients*. If you have done this correctly, you should get a Confirm Overwrite dialog similar to that shown in Figure 26.

This dialog asks you to confirm that you want to overwrite the contents of the folder with what is currently in the repository. Since there is nothing currently in the repository, this is not a dangerous operation, and you should select the *OK* button.

At this point, the *BigBucks* project is under Subversion control, as you can see by the icon decoration in the Project Explorer window. But the IDL source files still have not been added to the repository. You need to do so now.

In the Project Explorer window, navigate to the *source* directory inside the *BigBucks* project. Open it and select the **.pro* files. Right-click on one of the select files and choose *Team -> Add to Version Control*. The IDL files will now be added to the *bigbucks/source* folder in the repository.

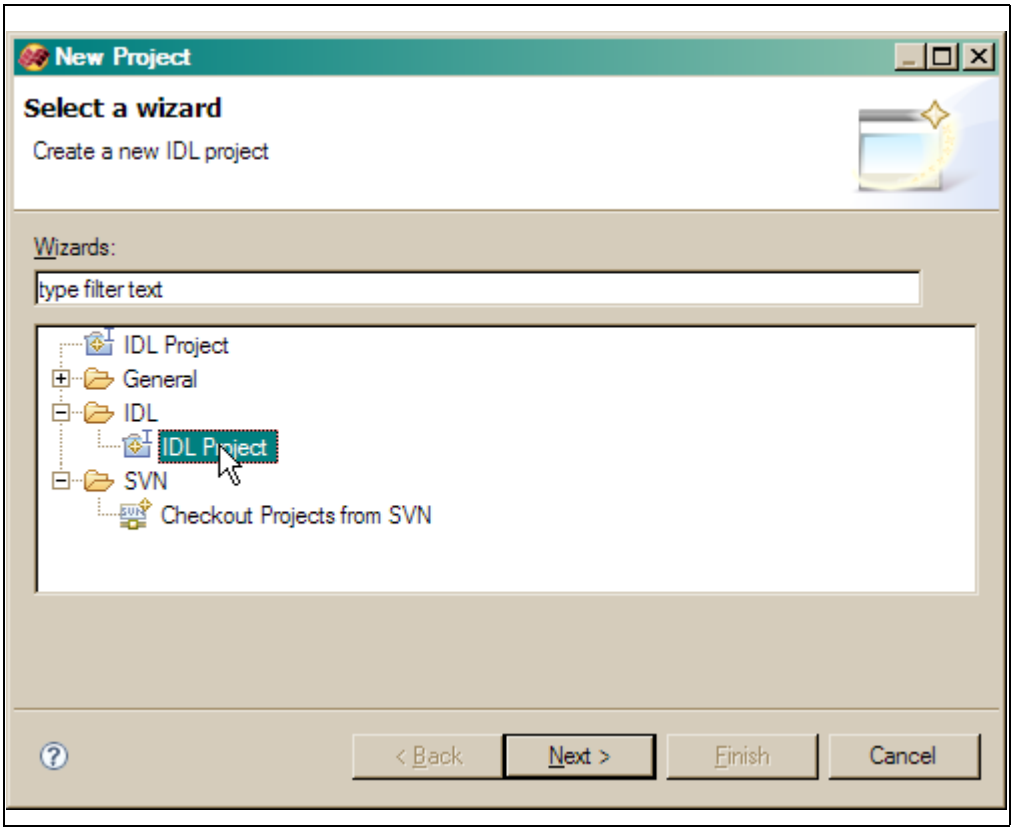


Figure 25: We are checking out the bigbucks folder in the repository to a new IDL project that already exists on disk.

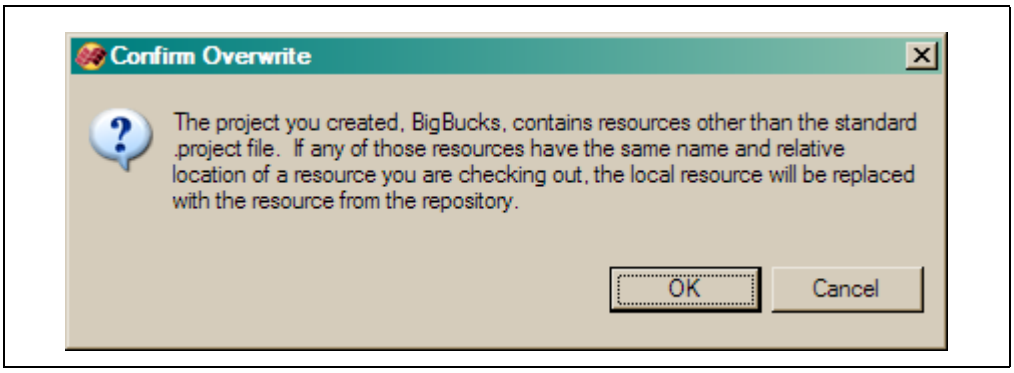


Figure 26: This dialog asks you to confirm that you want to overwrite the current contents of the directory with the contents of the repository. Since there is nothing in the repository currently, this is not a dangerous operation and you should hit the OK button.

Step 9: Enjoy Working in IDL 7

Everything else you want to do with IDL and Subversion are just variations on what you learned in this tutorial. If you want to learn more about how to use Subversion itself, I recommend the on-line Subversion book, or even better, the on-line TortoiseSVN help. Enjoy!

Appendix A: Hints and Other Odds and Ends

What I've collected here are a few of the odds and ends I've run into and had to figure out my way around. They are listed in no particular order, but they are things it might be handy to know every now and then.

Changing Project Properties

If you get a project set up incorrectly (perhaps you have misspelled its name, or located it in the wrong folder, or who knows what else), you can easily correct the problem. Simply right-click the project in the Project Explorer window and you will find selections that will allow you to change various properties of the project. Here, for example, is where you would rename a project, or delete it entirely. If you delete it, you have the choice of deleting just the project itself, or the entire directory structure attached to the project. If you do the latter, be careful that is *exactly* what you mean to do, as there is no way to undo that operation.

Path Properties

If you wish to change the way IDL works with paths for a project find the *Properties* menu selection at the bottom of the pop-up menu. find the *IDL Project Properties* selection in the left side of the Properties dialog. Here you can check or uncheck the *Update IDL path preference* option. If this option is selected, the path to the project is added to the end of the !PATH system variable when the project is opened. The project path is deleted from the !PATH system variable when the project is closed. The order in which files are ordered in the !PATH system variable, then, can vary from one IDL session to the next, depending upon the order projects have been opened.

If you want a more defined or reliable !PATH system variable, then turn this property off for your projects and manage your IDL Path in the style of previous versions of IDL. You will be able to construct your IDL Path from the *Window -> Preferences -> IDL -> Paths* dialog. Directories added to the !PATH system variable in this way are never added or deleted when projects are opened or closed.

Note that IDL will "analyze" the code in any directory found on your !PATH system variable, when IDL starts up. In IDL 7.0 this can take from 1 to 10 minutes or more. You should be able to work during this time, but some files may not be ready for color coding, etc. until the process is completed. I typically put only frequently used IDL library code (e.g., the *coyote* directory) directly on my path, and let other projects add themselves to the path when they are opened.

Adding Project to Projects

Projects can be added to other projects, but new projects cannot be created directly in a project folder. To add already created projects to another project, right-click the other project and chose the *Properties -> Project References* selection. Select which projects you wish to add to the project.

Problems Creating Projects

If you have a problem creating a project that you have been working with previously, the problem can be a left over *.project* ("dot project") file in the project folder. This file is added to the project folder automatically when the project is created. If you chose to delete a project, but not delete the contents of the directory, this project file will be left behind, and it can prevent you from making this directory a project later on. Just remove the *.project* file before you try to create a project from the folder.