Compiling PRO/5 Code Using Eclipse and BDT
A Tutorial

by Jerry Karasz and Sebastian Adams

If you are interested in learning how to use Eclipse and the Business BASIC Development Toolkit (BDT) to edit text versions of your PRO/5 code, this is the tutorial for you. Follow the steps below to learn what Eclipse and BDT can do for your PRO/5 development.

The PRO/5 Compiler Plug-in

The PRO/5 Compiler Plug-in helps you compile PRO/5 code within the Eclipse environment. Compiling, though, is only one small part of converting PRO/5 code to BBj. For a complete description of the steps to follow to move PRO/5 code forward to BBj, see the Converting to BBj from Earlier Versions of BASIS Products documentation. As preparation for possibly wanting to test and debug your PRO/5 code in BBj, as will be explained later, you likely will want to complete at least steps 1 and 2 of this process.

The BBj CodeEditor (installed as part of the BDT Plug-in) is as close as any editor gets to being a “PRO/5 editor”. Because of that, this article uses the BBj CodeEditor to edit PRO/5 programs. That editor is only available when you use Eclipse’s BDT Perspective with the BDT Explorer navigation control to manage your program files, and when you organize your PRO/5 code into one or more of Eclipse’s BBj Projects. Fitting your PRO/5 code into this Eclipse development model is covered in subsequent chapters of this article. For now, though, let’s look at what the plug-in is, and what it is not.

What It Is

The PRO/5 Compiler plug-in allows you to invoke the pro5cpl compiler from inside of Eclipse. As such, it offers the following functionality to identify errors in your code, and then to locate the source of those errors so that you can fix them:

- You can compile a PRO/5 program with a single click or automatically with every save
- You can compile multiple PRO/5 programs at one time
- You can access the tokenized PRO/5 programs placed in an output folder you define
- You can view any problems (errors and warnings) reported by the pro5cpl compiler
- You can double-click a PRO/5 error or warning in Eclipse to open the source file involved and scroll to highlight the source of that problem
- You can edit PRO/5 source code in any text editor supported in Eclipse, including the generic Text Editor or the BBj CodeEditor

What It Is Not

The PRO/5 Compiler plug-in is not a fully-functional Interactive Development Environment (IDE) for PRO/5 programs. It does not support the following functionality because it is not integrated with a PRO/5 interpreter and does not include a dedicated PRO/5 editor:

- You can use an editor other than the BBj CodeEditor, but then you will not have BBj code syntax coloring (all of the code will be in black text). And even the BBj CodeEditor is not a perfect fit, because it provides syntax coloring in accordance with BBj language parsing rules, not PRO/5 rules (although they are very, very close…).
- You can only see BBj parser errors as you edit in the BBj CodeEditor; most of these are also PRO/5 errors, but there are a few parser differences between the two languages.
- In the BBj CodeEditor, you will not see PRO/5 errors and warnings (unless they are also BBj errors and warnings) until you save (with *auto-compile* on) or manually PRO/5-compile your edited program.
- In the BBj CodeEditor, you will only see code completion assistance according to BBj language rules. Since most of the BBj code completion offerings relate to attributes and methods on BBj classes, the BBj CodeEditor does not offer any significant PRO/5 code completion assistance.
- Since the PRO/5 Compiler plug-in is not integrated with the PRO/5 interpreter, you cannot run or debug your PRO/5 programs in a PRO/5 interpreter. In many cases you could run your PRO/5 programs as BBj programs using BDT's connection to a running BBjServices. Of course, while BBj is highly compatible with PRO/5, it's not necessarily always identical in every detail, so final testing before deployment should always be done in the closest possible equivalent to the production environment, including the same BBx interpreter (either PRO/5, Visual PRO/5 or BBj.) For example, some applications have been known to define windows and controls to be sized as small as possible in the environment in which the application is developed. Since there are differences from one look and feel to another (some small, some large), a carefully designed window can look less good in a different look and feel.

Eclipse Requirements

In order to use the PRO/5 Compiler plug-in in Eclipse, you must follow Eclipse's model. The following statements should help you to understand a little more about working in Eclipse:

1. **Eclipse runs within a workspace.** All interactions with Eclipse occur within a workspace. You can create multiple workspaces, where each one is independent from the others.
2. **Eclipse relies upon a Perspective to organize its display.** BDT has defined a BDT Perspective so that Eclipse will know when you are working on BDT tasks using BDT features. BDT and its BBj CodeEditor are only fully enabled when the BDT Perspective is active.

3. **Eclipse only supports files that are part of a project.** Eclipse has a model that it “owns” every file that is in its projects. It manages them, monitors them, and offers them to users to view and edit. Eclipse only works well with source code files that it “owns”. Using any external editor (any editor such as vi or Emacs that is not invoked from inside of Eclipse) on those same files can confuse Eclipse.
4. **To use the BBj CodeEditor, source code files must be in a specific type of project: a BBj Project.** BDT’s BBj CodeEditor is only able to parse and work with source code files that it recognizes and owns in a BBj Project. You can create multiple BBj Projects in a workspace, and each helps you to organize your program files in folders. Each also provides some isolation from other BBj Projects, holding administrative properties defining how and when BDT performs certain actions such as building and parsing for BBj.

For a more extensive discussion of the role of perspectives, views, and projects in Eclipse and BDT, see the upcoming Advantage article titled *Perspectives and Projects and Views, Oh My!*, which is due out by fall of 2017.

Getting Started with BBj, Eclipse, and the Plug-ins

In order to install and use the PRO/5 Compiler Plug-in, you must first install BBj, and then install Eclipse and BDT. The PRO/5 Compiler plug-in relies on BDT, which in turn relies on Eclipse and BBj. The current BDT at the time this article was written was 17. BDT 17 requires an installation of BBj 17.xx and of Eclipse Neon (Eclipse version 4.6).

Installing

The first step is to install BBj 17.xx on your development computer. Once you have installed BBj, see the following online articles for detailed guidance on setting up Eclipse Neon and installing the BDT and BBjUtils plug-ins (please follow these instruction in the listed order):

1. Preparing Eclipse for BASIS-Provided Plug-ins - in the section “BASIS Eclipse Plug-ins”, go ahead and install the full BDT plug-in set as well as the BBjUtils plug-in that includes the PRO/5 Compiler plug-in).
2. Creating Your First BBj Project - this is useful as a test to ensure that the installations are all correct and complete. If you encounter problems executing this tutorial, you should resolve them before proceeding.

Setting Up a BBj Project with PRO/5 Code

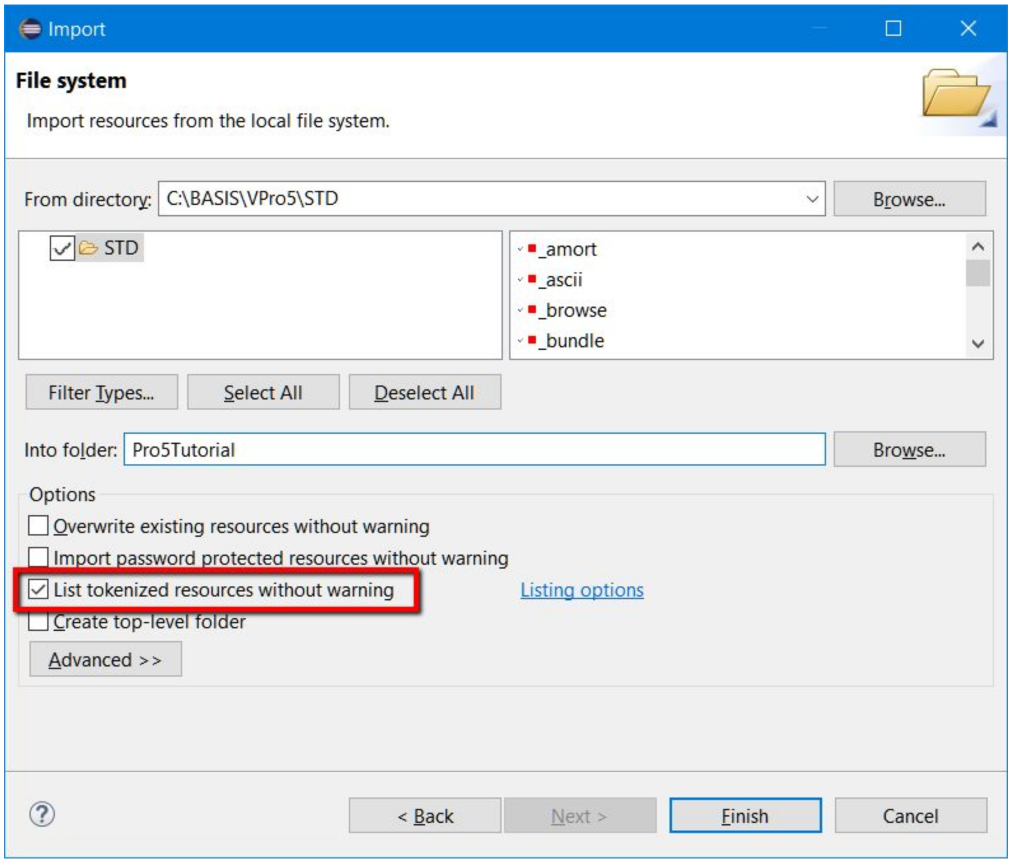
When you followed the *Creating Your First BBj Project* tutorial, you were led through the process of opening a workspace, selecting the BDT Perspective, viewing the BDT Explorer, and creating a BBj Project there. We will not cover those details again, but will proceed to create a new BBj Project with your PRO/5 code.

We will work under the assumption that you have some existing PRO/5 programs that you would like to edit in Eclipse, and that some number of these files are tokenized (the exact number is not important here). Since BDT and the BBj CodeEditor only edit ASCII files, the tokenized files need to be listed. All of the program files need to be imported into Eclipse using BDT and added to a BBj Project anyway, and the import process can handle listing the files for you. Let's import some PRO/5 sample code to demonstrate the process - since every PRO/5 installation includes some sample programs, we will treat the **<PRO/5 Installation Folder>/std** folder as containing the programs we need. Importing only required two steps:

Step 1: Create a new BBj Project in Eclipse following the model you learned in the *Creating Your First BBj Project* tutorial; for this tutorial, use the name **Pro5Tutorial** for this project. **NOTE:** if your PRO/5 files have no extension or have an extension other than .bbj or .bbx, be sure to check the **Add unrecognized file types to source path** radio button in the **New Project Source Options** group box - if you choose **Do not add unrecognized file types to source path**, BDT will not always know to treat your files as containing source code it can edit.

Step 2: Import the programs by right-clicking the **Pro5Tutorial** project and selecting **Import…** in the context menu. This launches the Import wizard that will guide you through importing:

- **Screen 1:** When you are asked to select an import wizard, expand the BDT entry in the tree and select **BDT > File System**. BDT's file system import will import an entire collection of files at one time, and will handle setting up those files correctly in the project. More importantly for us, it also will offer to automatically list any PRO/5 (or BBj) tokenized files it finds as it imports them. Click the **[Next >]** button to advance.
- **Screen 2:** Click the **[Browse…]** button at the top, next to the **From directory:** field. In the navigation display that appears, navigate to and select the **<PRO/5 Installation Folder>/STD** folder. Once the **STD** folder shows in the **From directory:** field, check the box next to the **STD** entry in the left-hand tree where it appears. This tells BDT to import all of the files it finds in that folder.
- Also in Screen 2, you may notice that Eclipse filled in the **Into folder:** field for us with the name of our new BBj Project (**Pro5Tutorial**). This determines which project will receive the imported files.
- Before leaving Screen 2, be sure to check the **List tokenized resources without warning** box. For more options you can set on the BBj lister, click the **Listing options** link just to the right of the checkbox:


Import

File system

Import resources from the local file system.

From directory: C:\BASIS\VPro5\STD

[✓] STD

_amort
_ascii
_browse
_bundle

Into folder: Pro5Tutorial

Options

[ ] Overwrite existing resources without warning
[ ] Import password protected resources without warning
[✓] List tokenized resources without warning
[ ] Create top-level folder

Figure 1. Importing using BDT's File System import

- When you click the **[Finish]** button, BDT will import all of the files in the **STD** folder and will list any of them that it needs to.

NOTE: The files in the **Pro5Tutorial** project are copies of the original files. You should maintain your original files as you deem necessary for backups and other archival purposes. The copies in the **Pro5Tutorial** project are now owned by Eclipse, and you should not in any way interact with those files except through Eclipse (do not open them in an external editor such as vi or Emacs or Notepad; do not overwrite them with different versions of the files; do not delete them; and so on). The copies will get the edits made inside of Eclipse, and they are the files that will be PRO/5 compiled by the plug-in.

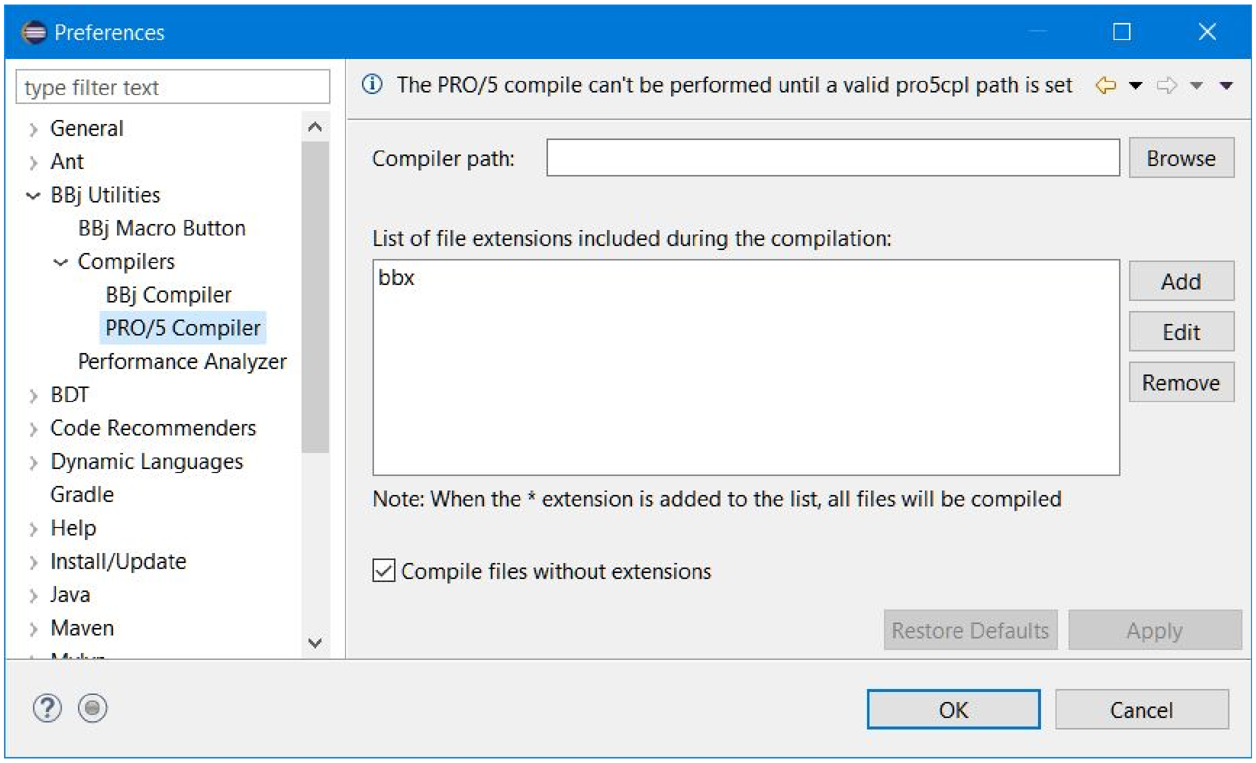
Configuring the PRO/5 Compiler Plug-in for Your BBj Project

Now that you have a BBj Project with code in it, you are ready to configure the PRO/5 Compiler plug-in. Until you configure it, the plug-in will not be active and you will not be able to compile PRO/5 code. There are two aspects to configuring the plug-in: Preferences, and Project Properties.

Preferences

Eclipse stores preference settings in the workspace, so any value you set here is only set in the workspace that you have open. If you create or open a second workspace, you will have to set these values there, too. These PRO/5 Compiler preferences apply to all of the projects in the workspace.

From inside of Eclipse, open the Preferences display. In Windows, this is done via the **Window > Preferences** main menu item; other operating systems have an analogous way to access preferences. Select the **BBj Utilities > Compilers > PRO/5 Compiler** entry in the navigation tree (as shown in **Figure 2**) to see the PRO/5 Compiler preferences:



The PRO/5 compile can't be performed until a valid pro5cpl path is set

Compiler path:

List of file extensions included during the compilation:

bbx

Note: When the * extension is added to the list, all files will be compiled

☑ Compile files without extensions

Figure 2. The PRO/5 Compiler preferences

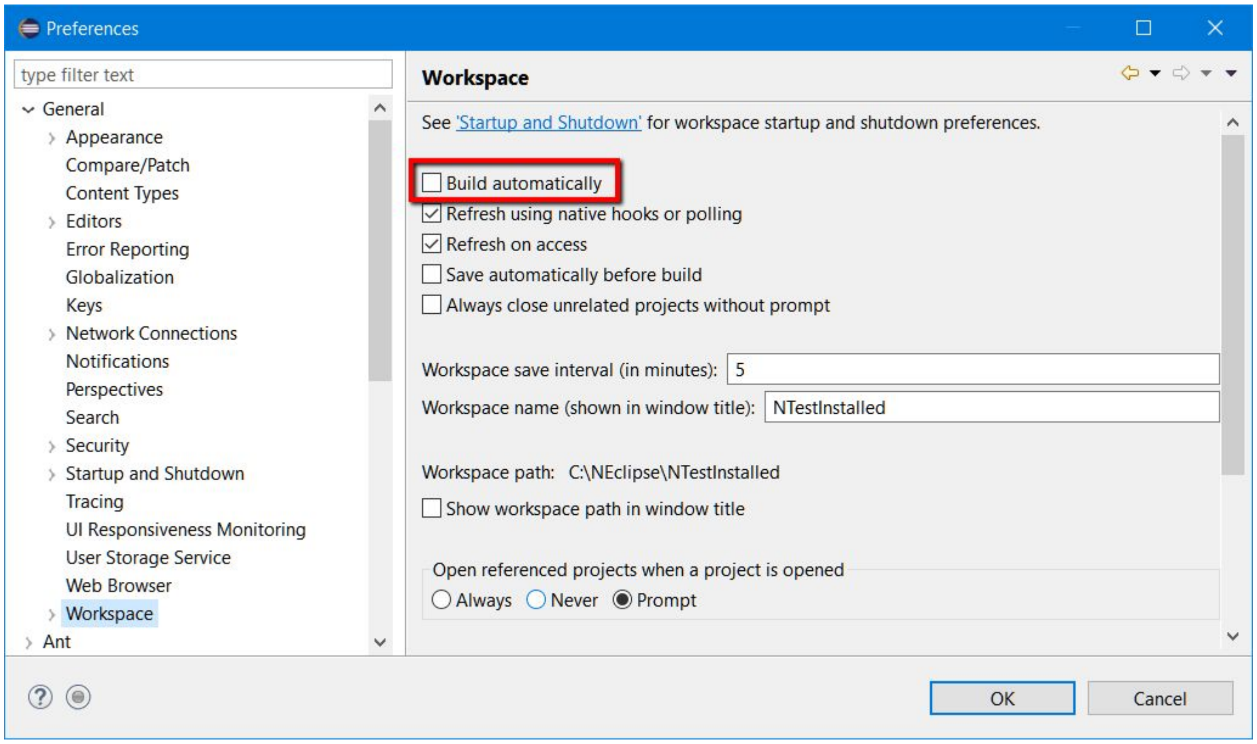
The top setting, the compiler path, is critical. Until you tell the plug-in where your pro5cpl executable can be found, it cannot compile files for you. To set the path, click the **[Browse]** button, navigate to and select your PRO/5 compiler executable, and click the **[Open]** button.

The list of file extensions defaults to only containing “bbx”. This means that you will only be able to invoke the PRO/5 Compiler on files that end with that extension (*.bbx). You can use the **[Add]**, **[Edit]**, and **[Remove]** buttons to tailor this list of file extensions to fit your source program files. To be able to compile files that have no extension, simply check the box below the list.

To save your changes, simply click the **[Apply]** button (which should now be enabled).

Other Preferences

Besides the PRO/5 Compiler preferences, there is one Eclipse preference setting that you may want to change. If you will be doing exclusively PRO/5 editing and compiling, you probably do not want to have Eclipse doing “automatic builds” for you with other compilers (such as the BBj and Java compilers). Disabling Eclipse’s “Build automatically” setting (see **Figure 3**) lets you stop all plug-ins from participating in automatic Eclipse Build events; this will not affect the PRO/5 Compiler plug-in because it does not participate in those build events, but it will keep you from losing time waiting for other compilations to complete, and minimize the number of problems markers that may end up being placed in your source code editor:



Workspace

See 'Startup and Shutdown' for workspace startup and shutdown preferences.

- [ ] Build automatically
- [✓] Refresh using native hooks or polling
- [✓] Refresh on access
- [ ] Save automatically before build
- [ ] Always close unrelated projects without prompt

Workspace save interval (in minutes): 5

Workspace name (shown in window title): NTestInstalled

Workspace path: C:\NEclipse\NTestInstalled

- [ ] Show workspace path in window title

Open referenced projects when a project is opened

Always Never [✓ Prompt]

Figure 3. Eclipse’s Build Automatically preference

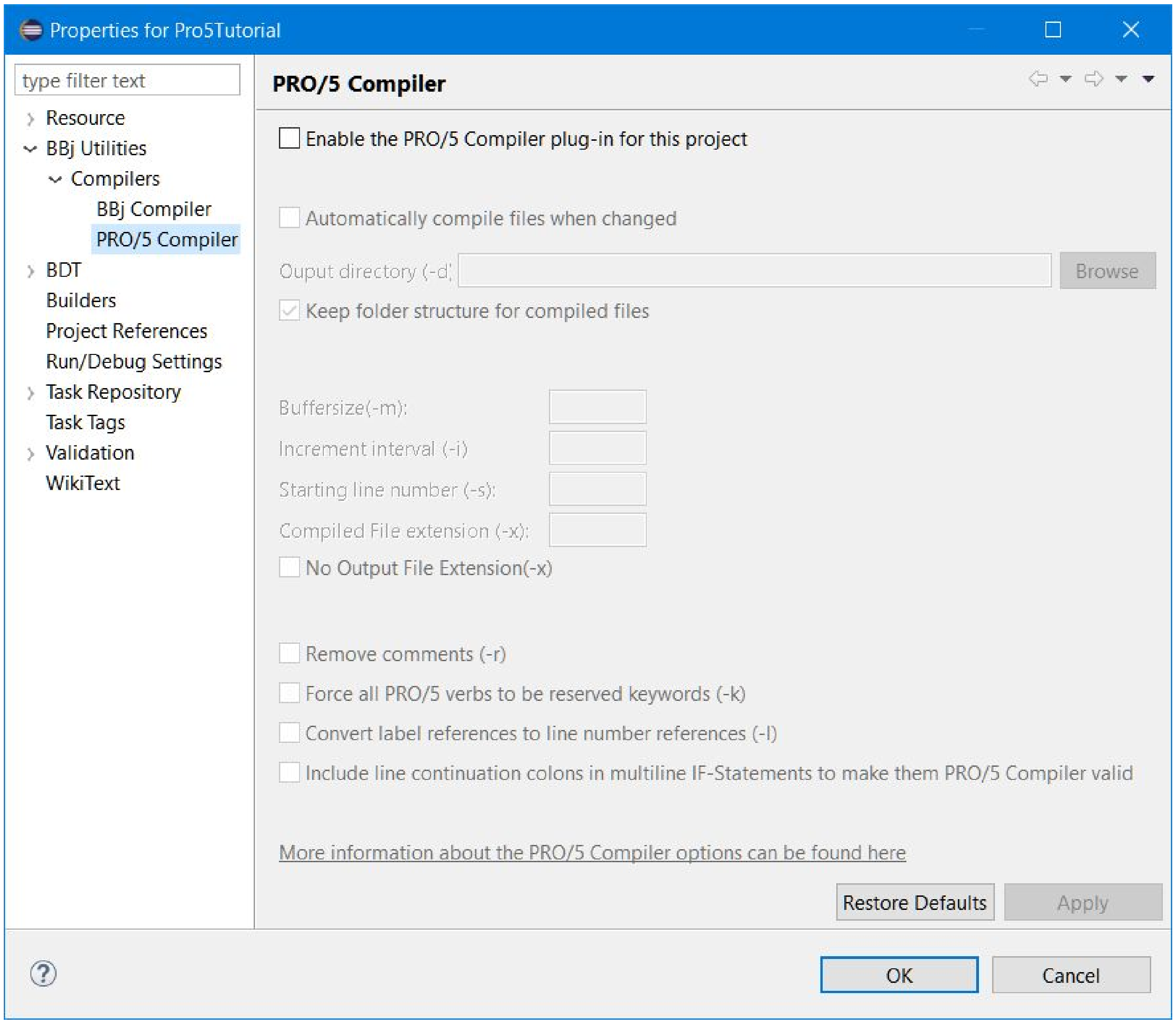
Simply uncheck the “Build automatically” box and click **[OK]** to save your change.

Project Properties

Eclipse stores properties for each individual project in the open workspace. Any value you set in a Project Properties display are only set on that project. If you create a second project, you will have to set these values there, too.

From inside of Eclipse, in the BDT Explorer, open a Project Properties display on the project you just created with your PRO/5 source code. This is done by right-clicking the project and selecting the **Properties** context menu item. Select the **BBj Utilities > Compilers > PRO/5**

Compiler entry in the navigation tree to see the PRO/5 Compiler project properties (as shown in **Figure 4**):


Properties for Pro5Tutorial

type filter text
- Resource
- BBj Utilities
  - Compilers
    - BBj Compiler
    - PRO/5 Compiler
- BDT
- Builders
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

PRO/5 Compiler

[ ] Enable the PRO/5 Compiler plug-in for this project

[ ] Automatically compile files when changed

Ouput directory (-d): Browse

[✓] Keep folder structure for compiled files

Buffersize(-m):

Increment interval (-i)

Starting line number (-s):

Compiled File extension (-x):

[ ] No Output File Extension(-x)

[ ] Remove comments (-r)

[ ] Force all PRO/5 verbs to be reserved keywords (-k)

[ ] Convert label references to line number references (-l)

[ ] Include line continuation colons in multiline IF-Statements to make them PRO/5 Compiler valid

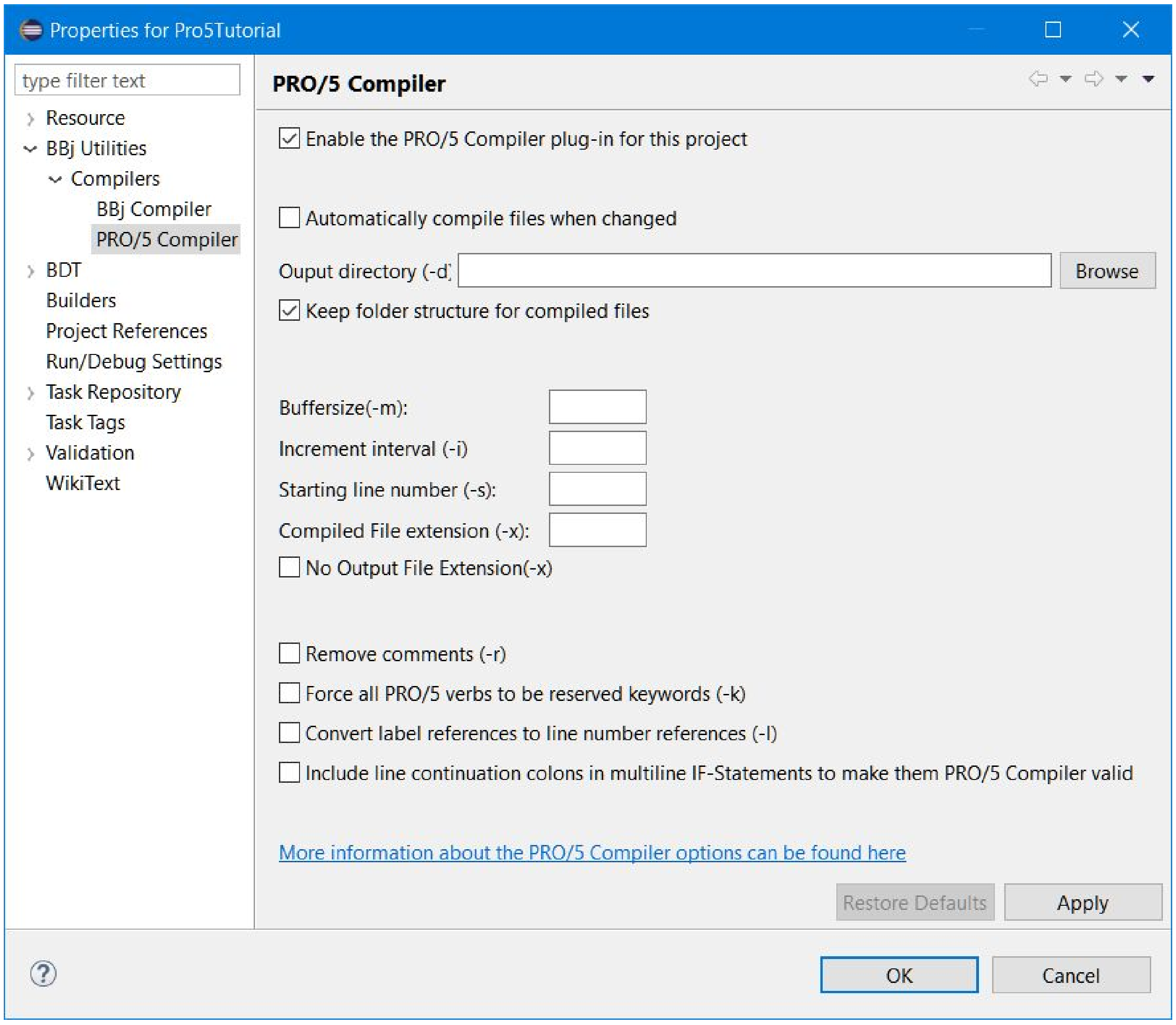
More information about the PRO/5 Compiler options can be found here

Restore Defaults | Apply

OK | Cancel

Figure 4. The PRO/5 Compiler BBj project properties by default

Notice that all of the settings are disabled until you check the first box to enable the compiler for source files in this project. Once you check this box, you will be able to edit the other properties (**Figure 5**):



Properties for Pro5Tutorial

PRO/5 Compiler

☑ Enable the PRO/5 Compiler plug-in for this project

☐ Automatically compile files when changed

Ouput directory (-d)

☑ Keep folder structure for compiled files

Buffersize(-m):

Increment interval (-i)

Starting line number (-s):

Compiled File extension (-x):

☐ No Output File Extension(-x)

☐ Remove comments (-r)

☐ Force all PRO/5 verbs to be reserved keywords (-k)

☐ Convert label references to line number references (-l)

☐ Include line continuation colons in multiline IF-Statements to make them PRO/5 Compiler valid

More information about the PRO/5 Compiler options can be found here

Figure 5. The PRO/5 Compiler BBj project properties enabled for editing

Now that the compiler plug-in is enabled for this project, the remaining settings values other than the **Output directory** are optional. You can leave them at their default values, or you can edit them.

To have the plug-in automatically trigger a PRO/5 compile when you save changes to a file in a project where it is enabled, check the **Automatically compile files when changed** box.

You must specify an **Output directory:** (a folder that is accessible on this development computer) where the plug-in can put the output files each time it compiles a file in this project. This folder must be a location where you have permission to write, and should NOT be anywhere in Eclipse's folder structure (not in the workspace nor under the Pro5Tutorial project root folder, both of which Eclipse manages as source code locations, not output locations). You probably also do not want to have the compiled output files placed back into the original folder from which we just imported. Unless, of course, you want to overwrite your original files. One purpose of having a dedicated output folder for each project is so that you can know where to go to find the compiled programs when you are ready to transfer them to a test platform or to a production machine. To specify an Output Directory, either enter the full path or use the **[Browse]** button to select a folder. Remember, this output folder is for this BBj Project; to avoid file collisions, use a different output folder for each project.

To have the plug-in organize this project's compiled output files into a folder structure that mirrors the folder structure of your source code files, check the **Keep folder structure for compiled files** box.

For information on the command line options (those with a "**-**" and a letter in parentheses), click the hyperlink at the bottom. This will open a browser window on the help page for the pro5cpl command line arguments. For the purpose of this tutorial, leave those values at their defaults.

To save your changes, simply click the **[OK]** button. Repeat these steps for each project where you would like to use the PRO/5 Compiler plug-in.

Compiling with the PRO/5 Compiler Plug-in

Now that you have told the plug-in where to find your pro5cpl executable, and enabled the plug-in for the BBj Project with your PRO/5 source code, you are ready to compile something.

From inside of Eclipse, in the BDT Explorer, expand the BBj Project you created with your PRO/5 source code. There are several options for compiling your files: select file(s) in the BDT Explorer and compile them, compile the source code that is open in the active editor, or compile all of the files with the correct extension in one project. Let's look at each of these.

Compile Based on the BDT Explorer Selection

If you select one or more files in the BDT Explorer that have a file extension identified in the PRO/5 Compiler Preferences page as "valid for a PRO/5 compile", then you can right-click the selected file(s) and click **Compile PRO/5** in the context menu.

Compile Based on the Active Editor

If you open a file in an editor (usually done by simply double-clicking the file in the BDT Explorer), and that file has an extension identified in the PRO/5 Compiler Preferences page as "valid for a PRO/5 compile" and is in a project that has PRO/5 compiling enabled, then you can click the PRO/5 "Compile" toolbar button in Eclipse's main menu as shown in **Figure 6**:

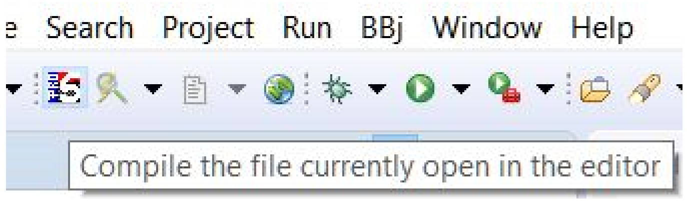

e Search Project Run BBj Window Help
Compile the file currently open in the editor

Figure 6. The PRO/5 Compiler toolbar button

Compile All Valid Files in a Single BBj Project

If you right-click a BBj project in the BDT Explorer that has PRO/5 compiling enabled, then you can click **Compile Project PRO/5** in the context menu as shown in **Figure 7**:

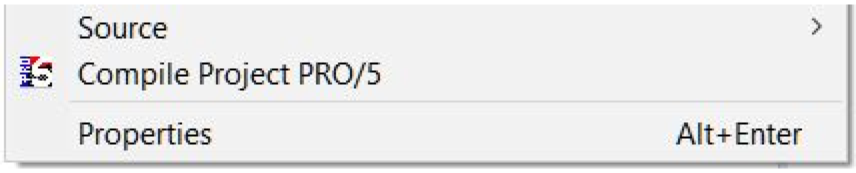


Figure 7. The PRO/5 Compiler context menu for a BBj project

This will compile every file in that project that has an extension identified in the PRO/5 Compiler Preferences page as “valid for a PRO/5 compile”.

Viewing the pro5cpl Command Line

Regardless of which method you use to compile PRO/5 source code files, Eclipse’s Console view will display the command line that was actually used to execute pro5cpl. To see this, click on the “Console” tab (usually in the pane at the bottom of Eclipse, but not always). If the Console view is not open yet, open it from the main menu using the **Window > Show View > Console** menu item seen in **Figure 8** below:

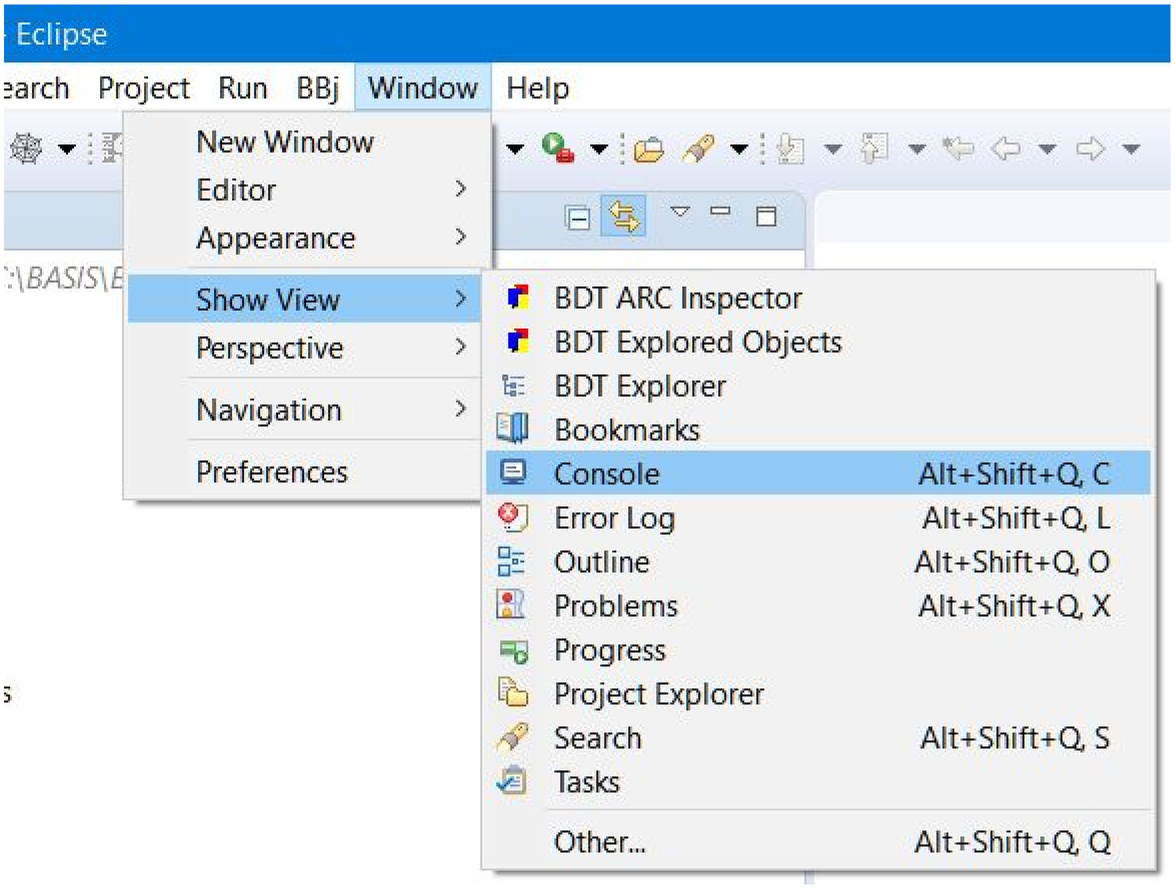
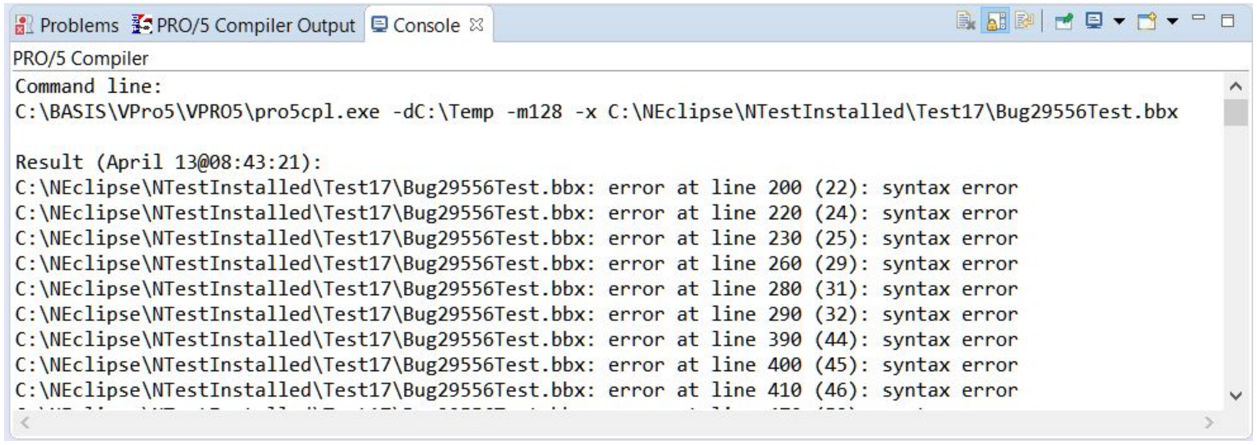


Figure 8. Opening the Console view

Alternatively, you can use the **Window > Show View > Other...** menu item and select the **General > Console** view in the tree. If the Console view is open when you compile, or is opened shortly thereafter, it will display Eclipse's command line interaction with pro5cpl. For example, in **Figure 9** we just compiled Bug29556Test.bbx. This is a new file created for this tutorial so that we can demonstrate how you will see and manage problems (errors and warnings) when you encounter them in your programs:



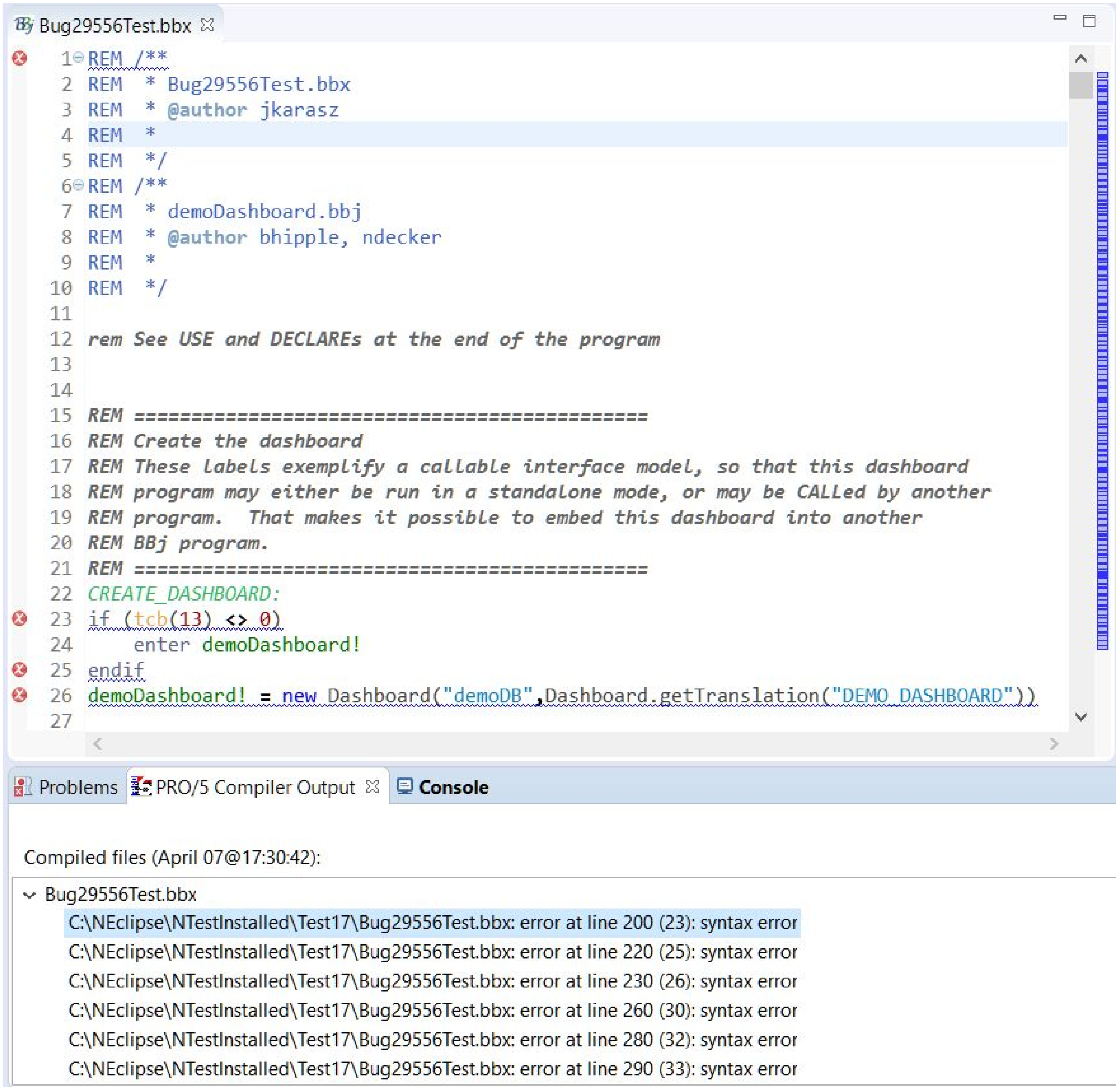
```
PRO/5 Compiler
Command line:
C:\BASIS\VPro5\VPRO5\pro5cpl.exe -dC:\Temp -m128 -x C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx

Result (April 13@08:43:21):
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 200 (22): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 220 (24): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 230 (25): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 260 (29): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 280 (31): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 290 (32): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 390 (44): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 400 (45): syntax error
C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 410 (46): syntax error
```

Figure 9. The Console view after compiling

The “Result” information is also shown in the PRO/5 Compiler Output view that we discuss below. The Console view is a non-interactive display solely for the purpose of checking that the plug-in issued the desired pro5cpl command line.

Working with the PRO/5 Compiler Results

After you compile a PRO/5 source code file in the BDT Perspective using any of the methods outlined above, Eclipse will automatically open the PRO/5 Compiler Output view (usually at the bottom of Eclipse). In **Figure 10**, we have just compiled a file with an extension of “bbx” by clicking Eclipse’s toolbar button while the file was open in an editor (in this case, a BDT CodeEditor):



```
REM /**
REM  * Bug29556Test.bbx
REM  * @author jkarasz
REM  *
REM  */
REM /**
REM  * demoDashboard.bbj
REM  * @author bhipple, ndecker
REM  *
REM  */

rem See USE and DECLAREs at the end of the program


REM ============================================
REM Create the dashboard
REM These labels exemplify a callable interface model, so that this dashboard
REM program may either be run in a standalone mode, or may be CALLed by another
REM program.  That makes it possible to embed this dashboard into another
REM BBj program.
REM ============================================
CREATE_DASHBOARD:
if (tcb(13) <> 0)
    enter demoDashboard!
endif
demoDashboard! = new Dashboard("demoDB",Dashboard.getTranslation("DEMO_DASHBOARD"))

```

```
Compiled files (April 07@17:30:42):
Bug29556Test.bbx
    C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 200 (23): syntax error
    C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 220 (25): syntax error
    C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 230 (26): syntax error
    C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 260 (30): syntax error
    C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 280 (32): syntax error
    C:\NEclipse\NTestInstalled\Test17\Bug29556Test.bbx: error at line 290 (33): syntax error
```

Figure 10. The PRO/5 Compiler results after compiling

The PRO/5 Compiler Output View

Examine first the PRO/5 Compiler Output view. Initially, the list of problems shows as collapsed, so expand the (only) file entry by clicking the ">" arrow on the far left. Our file has a number of problems reported by pro5cpl. The output is shown in the order it was listed by pro5cpl, which is generally in increasing line number (when a line number is present). All errors that have no line number are consolidated and reported on line 1 of the editor (since there is no specific line involved). In the compilation above, hovering over the problem marker on line 1 would have displayed information about being "out of program buffer space", a problem not associated with any single line of code.

Still in the view, double-click any error or warning entry. Eclipse will open an editor on the appropriate file and scroll as necessary to display and highlight the problem line.

Problems in the Editor

If there was already an open editor on a PRO/5 file compiled, then that editor will display markers and underlining for each error or warning reported. In the preceding image, the small red circles on the far left are error problem markers, and the line of text is underlined in blue. Similarly, small yellow triangle are warning problem markers, and the line of text is underlined in yellow.

Hover the mouse cursor over either the problem marker or the underlined text to see the text shown in the PRO/5 Compiler Output view for that line.

Saving and Compiling

When you are finished making changes to the code to fix a problem, simply use Eclipse's **[Save]** or **[Save All]** toolbar buttons, or the **File > Save** or **File > Save All** menu items to save your work.

Each time you ask the PRO/5 Compiler plug-in to compile any file, it will clear any existing results from the PRO/5 Compiler Output view and then run the compiler again.

Using the PRO/5 Compiler Plug-in

You are now ready to apply these steps to your own workspace and your own PRO/5 program files. You have everything you need to be able to edit, save, and compile your code, including identifying and acting on any problems that may exist there.

As you have seen, the PRO/5 Compiler plug-in is simple to configure, easy to use to compile, and intuitive to use to navigate back to whatever code causes errors or warnings in your PRO/5 code. Although it is not a fully integrated PRO/5 interactive development environment, it provides you with the functionality you need to edit and compile your PRO/5 programs in Eclipse. And it starts you on the process of moving your PRO/5 code to BBj and Eclipse.

Other Useful Eclipse Plug-ins

As you have seen, Eclipse is all about expandability. You can install many other plug-ins depending on your needs. The eclipse marketplace offers thousands of plug-ins, some related to specific programming languages, others to the Eclipse IDE itself. When it comes to PRO/5 development, you might find the following plug-ins useful:

- BBjKeywordHelp - Access BASIS documentation from within Eclipse
- PerformanceAnalyzer - Visualize your application’s trace files in a table editor to reveal inefficiencies and improve its performance

And once you are writing BBj Code, these plug-ins might be handy:

- BBjCodeFormatter - Format and DENUM your source code
- WindowBuilder - WYSIWYG editor to create and maintain ARC-Resource files
- AppBuilder - An editor to help you write event handler code
- Dialog Wizard - Create and maintain BBj Programs by parsing ARC-Resource files

For information on installing any of these BASIS plug-ins, see the **BASIS PLUG-INS** section of this web page.

And don’t forget - when you are ready to upgrade your PRO/5 code to BBj code, the BBj Projects that you have created will work equally well for BBj development - even better, in fact, since BDT will help you run your BBj programs as well edit and compile them. And, it will offer a wide range of BBj code completion assistance (offering valid language constructs and BBj classes/methods to help you write code more efficiently and quickly). Not only that, but just moving to BBj will allow you to take advantage of a number of features and functions that have been added to BBj over the years. Leverage your PRO/5 investment today!