IBM ILOG CPLEX Optimization Studio Getting Started with the IDE

Version 12 Release 5

Trademarks

IBM, the IBM logo, ibm.com, WebSphere, and ILOG are trademarks or registered trademarks of International Business Machines Corp., in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Contents

Figures

Tables

Part 1. Introduction to the CPLEX Studio IDE

Provides an overview of important concepts and features of the CPLEX Studio IDE that you should be familiar with before starting to work with it.

Chapter 1. Launching the CPLEX Studio IDE

There are various ways of starting the CPLEX Studio IDE and displaying the Welcome window and the main window.

Starting the IDE from Windows

You can start the IDE from the Windows Start menu, from Windows Explorer, or from the command line.

To launch the IDE from the Start Menu:

1. Click the Windows **Start** menu.
2. Select **All Programs** > **IBM ILOG** > **CPLEX Optimization Studio [version_number]** > **CPLEX Studio IDE**

To launch the IDE from Windows Explorer:

1. Go to `<Install_dir>\opl\oplide`, where `<Install_dir>` is your installation directory.
2. Double-click the IDE executable `oplide.exe`

To launch the IDE from the command line:

1. Open a command prompt window.
2. Enter `oplide`

   The Welcome window is displayed.

Starting the IDE from Linux

You can start the IDE from a Linux terminal.

To launch the IDE from a Linux terminal:

1. Open a terminal window.
2. Change directory to `<installdir>/opl/oplide`
3. Enter the command `./oplide`

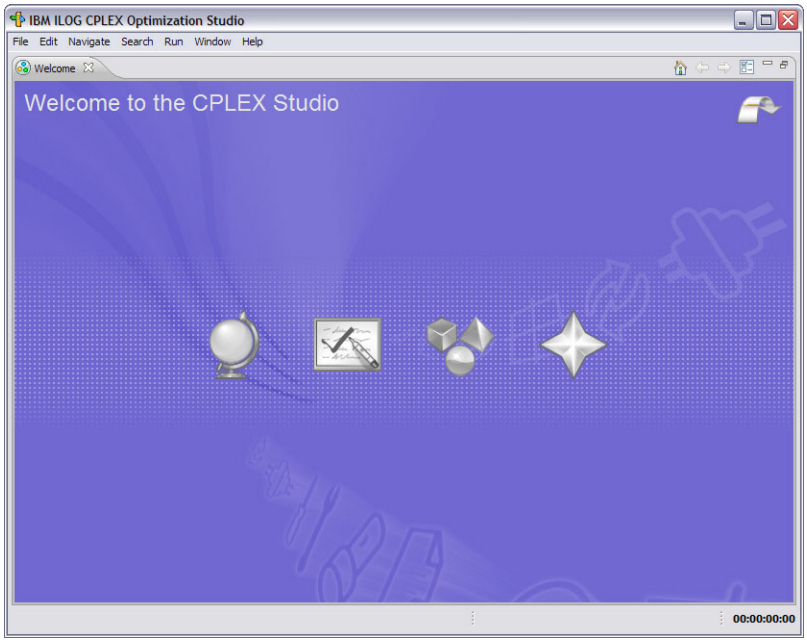
Or, from any terminal location, type the absolute path:

```
<installdir>/opl/oplide/oplide
```

The Welcome window

Describes the CPLEX Studio Welcome screen and how to close it and begin working with the IDE.

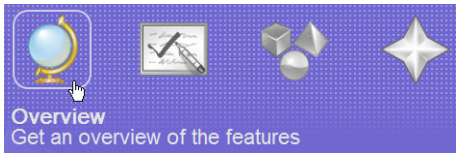
When you first launch the CPLEX Studio IDE, a Welcome window displays:



The buttons in the Welcome window provide access to user guides, sample manuals, release notes, migration information and user forums.

To access the information on the Welcome window:

Move the mouse over a button to see the tooltip.



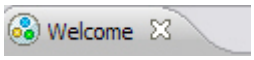
The buttons lead to information on the release and parts of the CPLEX Studio documentation you might frequently refer to.

- **Overview** — displays links to:
  - A quick start to CPLEX Studio
  - Migrating from previous versions of OPL
- **Tutorials** — displays a set of links to different sections of the IDE Tutorials manual
- **Samples** — displays a set of links to different sections of the Language and Interfaces Examples manual
- **What's New** — displays a set of links to:
  - The CPLEX Studio Release Notes, which provide an overview of new and changed features in the CPLEX Studio IDE and the OPL language.
  - This *Introduction to the CPLEX Studio IDE*

– A set of links to various OPL and ODM Enterprise user forums. These links are driven by an RSS feed, so they are constantly updated to reflect the latest information on those forums.

To close the Welcome window and use the CPLEX Studio IDE:

- Click the **X** in the Welcome window tab to close it.



- Or, click the **Workbench** icon at the top right of the Welcome window.

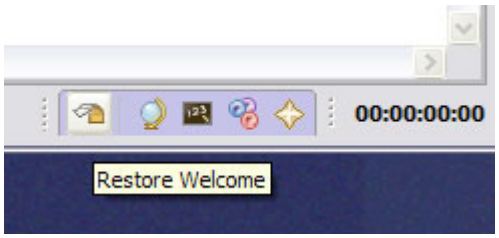
Closing the Welcome window using the second method displays a toolbar at the bottom of the IDE, in the Status Bar, as shown below.

Closing the Welcome window by clicking the **X** in the tab does not display this toolbar.

- When you close the Welcome window using either method, the main window appears. It is described in the next section.

To return to the Welcome window from the IDE:

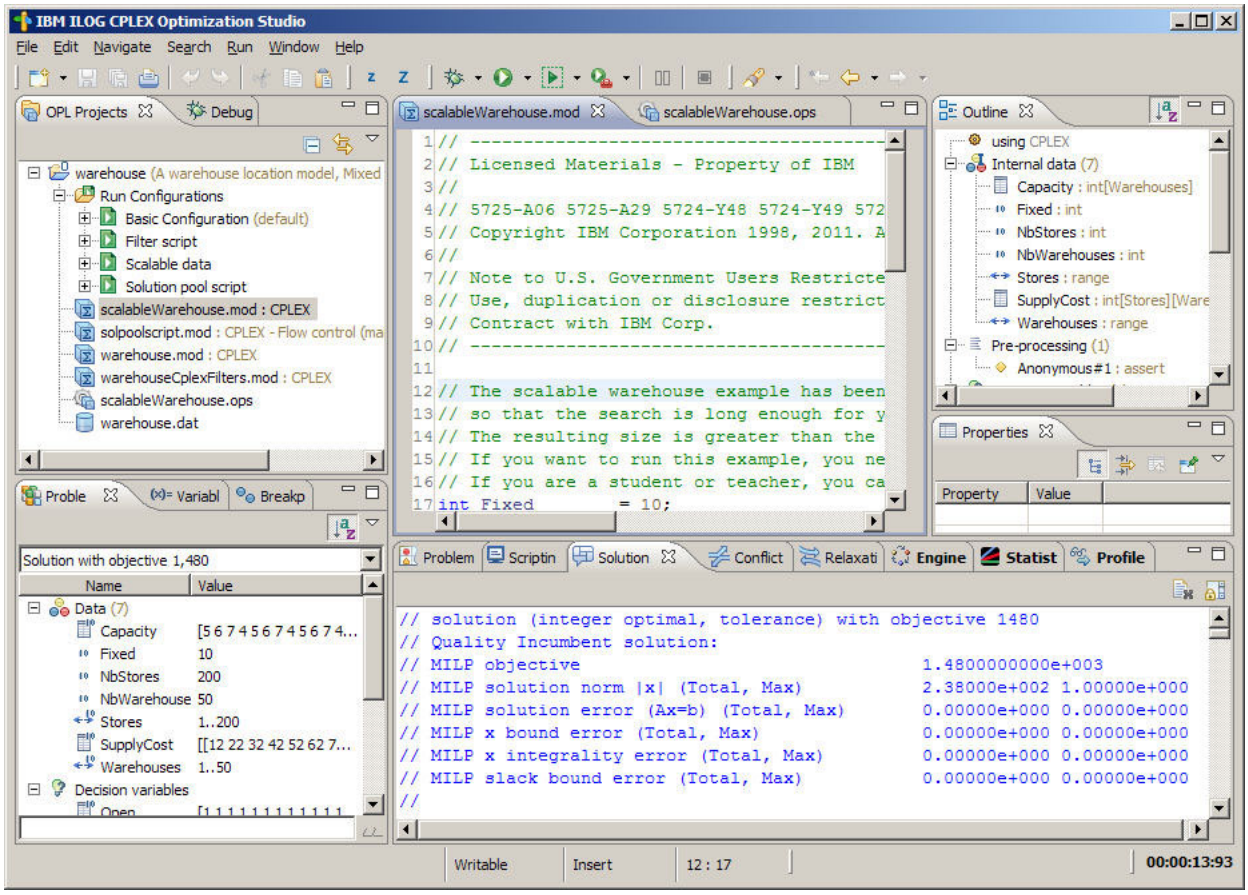
- Choose **Help** > **Welcome** from the main menu.
- Or, click the **Restore Welcome** icon at the bottom right of the IDE. (It appears when you click the Workbench arrow.)



The main window

Presents the main window of the CPLEX Studio IDE (Integrated Development Environment), with its primary areas and controls.

Tooltips appear when you move the pointer over most elements of the main window.



Chapter 2. Opening distributed examples in the CPLEX Studio IDE

Examples of OPL models and projects are distributed with CPLEX® Studio.

Why you should use the New Example wizard

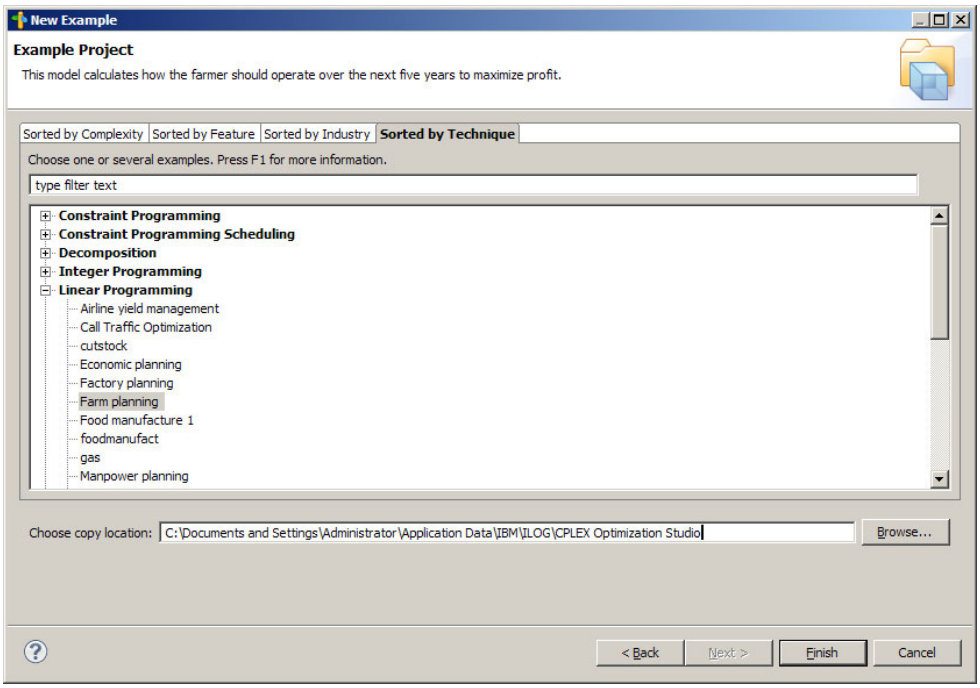
Why using the New Example wizard is the recommended method of opening distributed examples.

Examples of OPL models are available in the installation directory and can be opened in the CPLEX Studio IDE using **File** > **Import** > **Existing OPL projects** and the Import wizard that opens.

However, the recommended method of opening distributed examples is to use the **File** > **New** > **Example** menu command to launch the New Example wizard.

Benefits of using the New Example wizard:

- **Always working with a copy, not the original** — Opening examples using the New Example wizard ensures that you are always working with a copy of the example. Therefore, if you make changes and save them, the original example is always available to you in its original form.
- **Flexibility of project destination** — Using the New Example wizard, you can specify any folder in your file system as the destination for the project. Using the Import wizard, you have only the choice of opening the project in your default Workspace or "in place" , by opening the example in its original distribution directory. The latter risks the possibility of overwriting the distributed example by making changes and saving them, or of accidentally deleting the distributed example.
- **Enhanced project descriptions** — Using the New Example wizard, you see longer, more explicit descriptions of each project. Using the Import wizard, you see only the name of each example.
- **Enhanced sorting and display options** — The New Example wizard contains multiple tabs that allow you to view the examples sorted by complexity, feature being demonstrated, industry represented, and mathematical programming technique being used.
- **Built-in Help** — By selecting an example and pressing **F1** or clicking the Help icon  you can display the documentation for that example in the New Example wizard itself, and look it over before deciding to open the example.



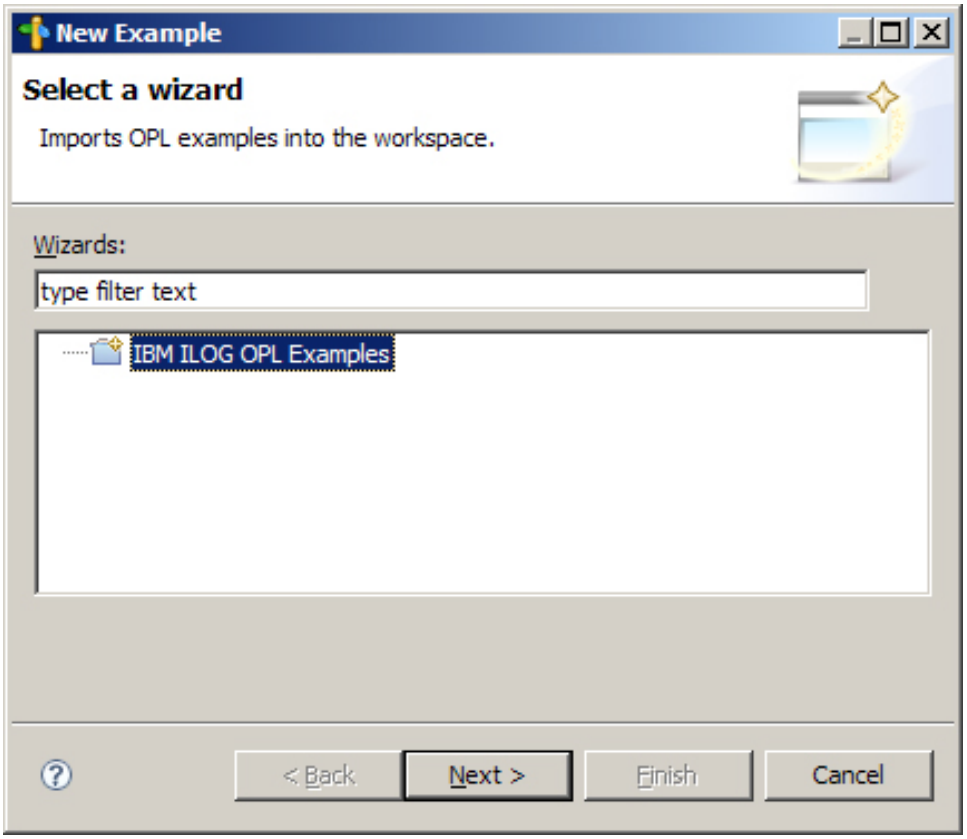
Working with the New Example wizard

Instructions for using the New Example wizard to open the distributed examples.

Procedure

To open examples in the CPLEX Studio IDE using the New Example wizard:

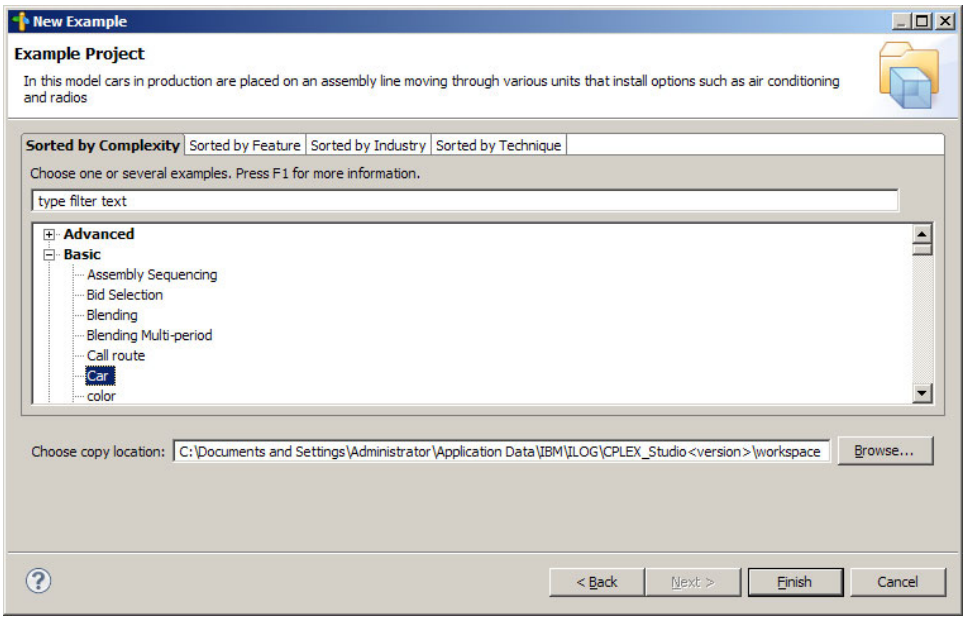
1. Choose **File** > **New** > **Example** to display the first screen of the New Example wizard.



Note:

You can also access the New Example wizard by choosing **File** > **Import** > **Example**. This command launches the same wizard as **File** > **New** > **Example**.

2. Select **IBM ILOG OPL Examples** and click **Next** to display the list of examples.



3. When you select an example project, a description of the project is shown at the top of the wizard.

4. To sort the list of examples, click one of the tabs:
   - **Sorted by Complexity** — to display examples grouped by complexity (Basic, Intermediate, Advanced, Demo, etc.)
   - **Sorted by Feature** — to display examples grouped by feature (OPL Model, OPL Project, OPL Script, etc.)
   - **Sorted by Industry** — to display examples grouped by industry (Finance, Manufacturing, Transportation, etc.)
   - **Sorted by Technique** — to display examples grouped by technique (Constraint Programming, Decomposition, Search, etc.)
5. To filter the display, type the text you are looking for in the field at the top of the wizard. Only examples that contain the text you enter will be displayed.
6. To obtain help for an example, select the example and press **F1** or click the **Help** icon  .
7. To select a destination for the opened project, type a path into the **Choose copy location** field or use the **Browse** button to search for one. The default location is your workspace.
8. Click **Finish**.

Results

The example appears in the OPL Projects Navigator. Expand the project and double-click the model name to display the model contents.

Tip:

Once you begin working with the New Example wizard, if you already know the name of the example you are searching for and you always open examples in your default IDE workspace, you will find that the fastest way to use it is:

1. On the second screen of the New Example wizard (step #2 above), type the name of the example you are searching for in the filter field.

   The example will be displayed in the selection window.
2. If the example displayed in the selection window is the one you want, simply press **Enter** twice to open it in the IDE.

Chapter 3. Working with projects in the OPL Projects Navigator

How to open OPL projects and work with them using the OPL Projects Navigator.

Importing existing projects into the workspace

Explains how to load OPL projects into the CPLEX Studio IDE.

About this task

Use the following procedure to import an existing OPL project into the OPL Projects Navigator.

Note:

This procedure is only for OPL projects. For importing projects from previous releases of OPL, see Migrating from previous versions of OPL.

In addition to the procedures detailed below, you can double-click on the files contained in an OPL project folder to open that project in the IDE:

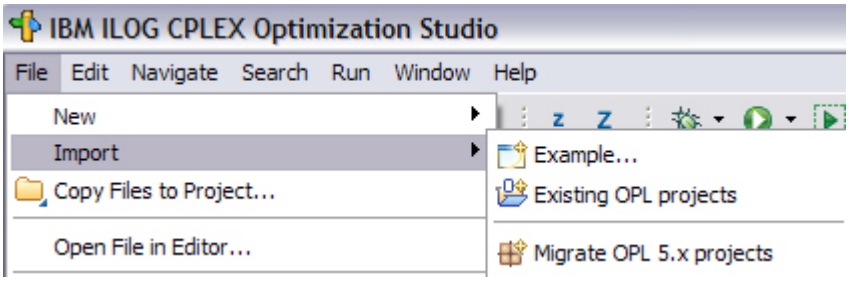
- If the project has already been added to the OPL Projects Navigator, double-clicking any model (`.mod`), data (`.dat`), or settings (`.ops`) file will open that project in the IDE.
- If the project has **not** already been added to the OPL Projects Navigator, double-clicking the file `oplproject` in

  `<Install_dir>\opl\examples\opl\<example_name>`

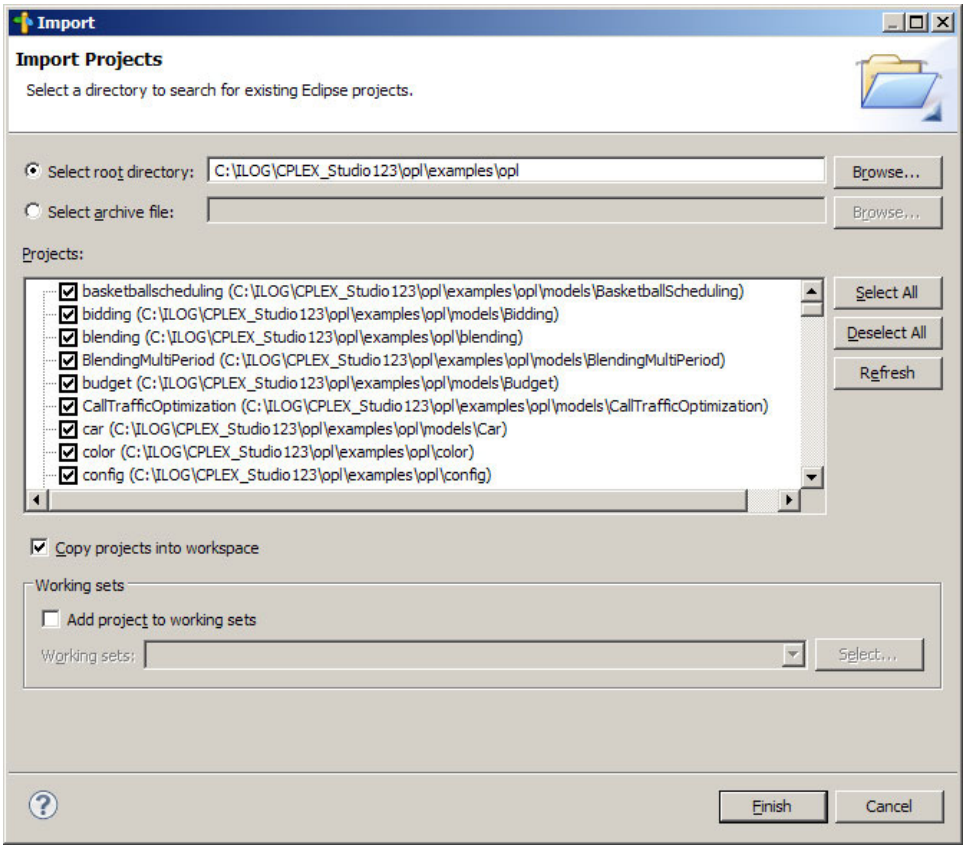
  will launch the Import Wizard discussed in the following procedures.

Procedure

1. To open an existing project, choose **File** > **Import** > **Existing OPL projects**, or right-click in the OPL Projects Navigator and choose **Import** > **Existing OPL projects**.



2. The first screen of the Import Wizard is displayed:



This screen can be used to load one or more OPL projects into the OPL Projects Navigator. The general procedure for doing this is to:

- Select a root directory (this is the directory on your file system where your existing OPL project is located). Alternatively you can select an archive file (JAR, ZIP, TAR) where the project has been stored.
- Select a project (or projects, if the root directory contains more than one project) to be imported.
- Indicate whether you want to copy the projects to the workspace, and to the working set if you have created one. The default workspace is:

  ```
  C:\Documents and Settings\Administrator\Application
  Data\IBM\ILOG\CPLEX_Studio<version_number\workspace
  ```

  CAUTION:

  If you do not copy the project into a workspace and work with the original, there is a risk of modifying or deleting the original example.

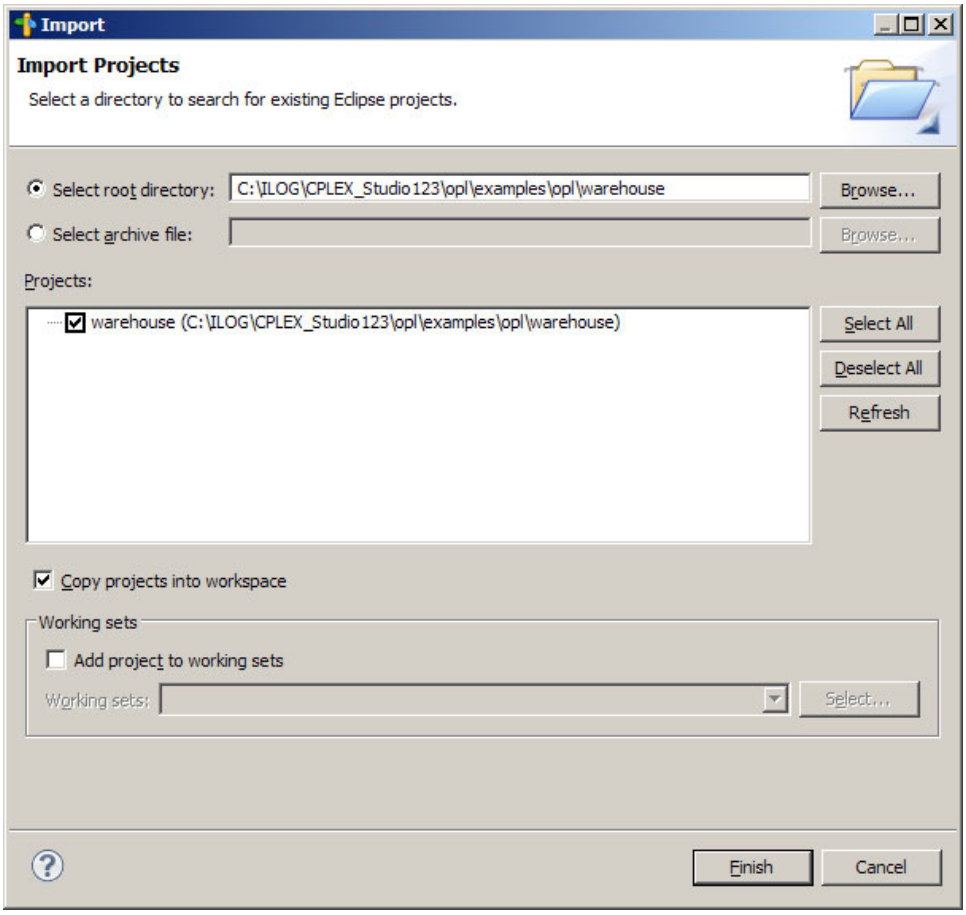
The next steps walk you through each of the general procedures above, using the `warehouse` example from the CPLEX Studio distribution.

3. In the **Select root directory** field, enter the path name of the directory that contains the project(s) you want to import. You can type in the path name or use the **Browse** button to search for it.

   Alternatively, you can use the **Select archive file** field and enter the path name of a JAR, ZIP, TAR or other compressed file that contains your project(s).

   After you have selected a root directory (or archive file), the OPL projects under that directory that do not currently exist in your workspace are listed in the **Projects** view.

4. Check the box of each of the projects you want to import. (In the example used in this procedure, there is only one, `warehouse`).



5. Leave the **Copy projects into workspace** box empty if you want to work with the project "in place" in its current location, or check the box to copy it to your workspace (recommended). If your projects are organized in working sets, you can add the project to an existing or new working set. Click **Finish** to import the project(s) into the OPL Projects Navigator.

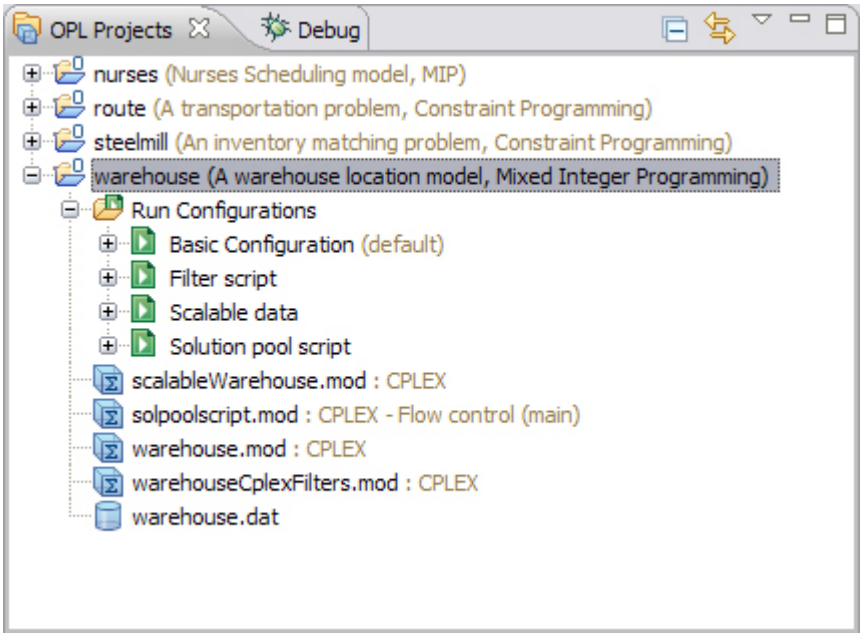
Managing projects in the OPL Projects Navigator

Explains how to work with your projects once you have imported them.

The OPL Projects Navigator is where you manage your projects by creating, adding, removing projects, models, data, settings, and run configurations. You can display more than one project at the same time (see Working with several projects)

The OPL Projects Navigator displays projects as tree structures with the files below them listed in alphabetical order (except for within run configurations, where the order of files is important). The `warehouse` project is expanded in the example below:

- the project name (and optional description) at the root
- an optional default run configuration (followed by 'default' in parentheses)
- one or more additional run configurations (optional)
- at least one model file
- data and settings files (optional)



OPL Projects Navigator

After you have migrated OPL 5.x projects or imported or created later versions of projects, you can leave them in your OPL Projects Navigator. If you exit from CPLEX Studio, when you next launch the IDE, they will be there, ready to use.

If you have loaded a number of projects and the OPL Projects Navigator starts to get full, there are two ways to save memory or space in the Navigator window — by closing (collapsing) the projects or by deleting them.

Closing/Opening projects

Projects are either open or closed. When a project is closed, it cannot be changed, but its resources still reside on the local file system. Because they are not examined during builds, closed projects require less memory. Therefore, closing projects you are not working with can improve build time.

- Right-click the project name and choose **Close project** from the menu to close the project. The plus sign next to the project name disappears, but it remains in the OPL Projects Navigator.
- To reopen the project, right-click the project name and choose **Open project** from the menu.

Deleting projects

If you are not currently working with a project, you can also safely delete it from the OPL Projects Navigator, without deleting it from the file system.

To remove a project from the OPL Projects Navigator, right-click the project name and choose **Delete** from the menu.

A popup message appears asking whether you want to delete the project only from the navigator, or from the hard disk as well.

- If you do not check the box, you remove the project from the OPL Projects Navigator but leave it on the file system.
- If you check the box **Delete project contents on disk (cannot be undone)**, the project will be completely deleted, and cannot later be recovered using **Undo** or the **Import** > **Existing OPL projects** menu command.

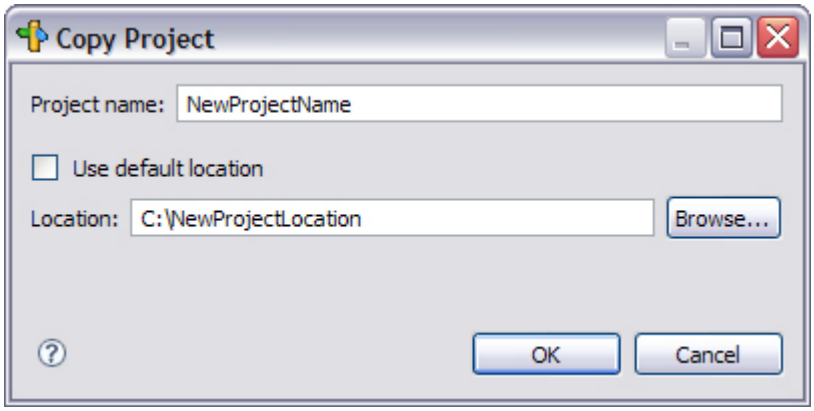
Copying projects in the OPL Projects Navigator

To make a copy of an existing OPL project, in order to modify a file and test new features, you need to copy the project and give it a new name.

To copy a project in the OPL Projects Navigator:

1. Open the project you want to copy. Right-click the project name and choose **Copy** from the menu, or press **Ctrl + C**.
2. Right-click again and choose **Paste** from the menu, or press **Ctrl + V**.

The following popup window is displayed with the default copy name. You may keep the default name, or choose a new name for the copied project.



File types

Lists the different types of files used by an OPL project, along with their extensions.

Each type of file in a project has a distinctive file name extension, as shown in the table below.

File Extension	Description
`.dat`	Files containing data instances
`.mod`	Files containing modeling and scripting statements
`.opl`	Compiled model files
`.ops`	Settings files, containing user-defined values for OPL language options, CPLEX parameters, and CP Optimizer parameters

For more information, see:

- Chapter 10, “Creating a project,” on page 47

- Understanding OPL projects in the *Quick Start* manual
- Generating output files in *IDE Reference*
- The Problem browser toolbar in *IDE Reference*
- Right-click menu commands in *IDE Reference*

Chapter 4. Important concepts and terms

Describes some of the terms and concepts that are important to understand about the CPLEX Studio IDE.

Resources

Describes what resources are in CPLEX Studio.

Resources refers to the projects, folders, and files that exist in the CPLEX Studio IDE. The OPL Projects Navigator provides a hierarchical view of these resources and allows you to open them for editing.

There are three basic types of resources:

- **Files** — Similar to files in the file system.
- **Folders** — Similar to folders in the file system.
- **Projects** — Used for builds, version management, sharing, and resource organization. Projects are comprised of folders and files, and map to directories in the file system.

Workspace

Describes the CPLEX Studio workspace.

The *workspace* is the working directory in which you store and work with your resources.

The workspace can be located anywhere on the file system, but its default location is `C:\Documents and Settings\Administrator\Application Data\IBM\ILOG\CPLEX_Studio<version_number>`.

Views

Explains the different types of views in the CPLEX Studio IDE.

The various panes or windows within the CPLEX Studio IDE are referred to as *Views*.

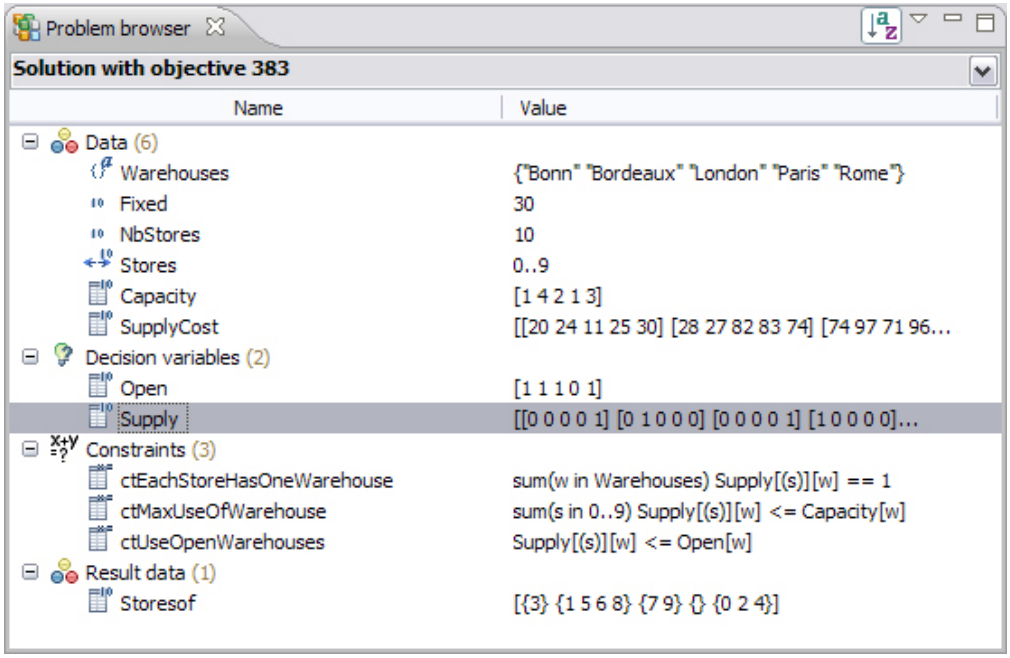
Views can be editors or navigators or provide alternative ways to visualize and work with your projects. For example, the OPL Projects Navigator view displays the resources in your OPL projects, and allows you to open them in editor views.

Views may also have their own menus and some views have their own toolbars.

Chapter 5. The Problem Browser

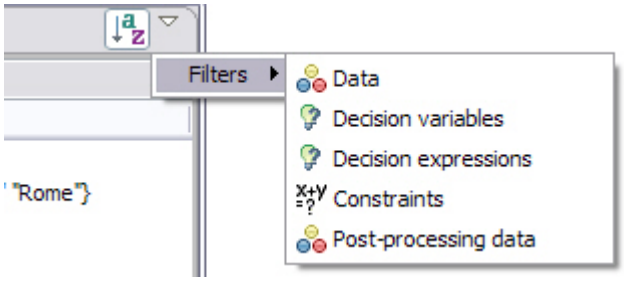
Describes the features of the OPL Problem Browser.

Describes the OPL Problem Browser.

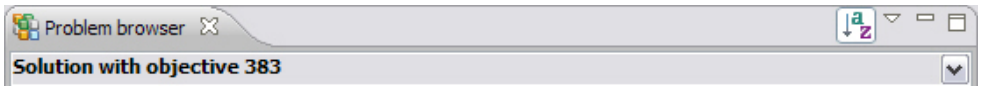


Some of the features of the Problem Browser are described briefly below:

- Sort the items of each displayed element.
- Filter the displayed element types.



- See the solution status in the drop-down list at the top of the view.



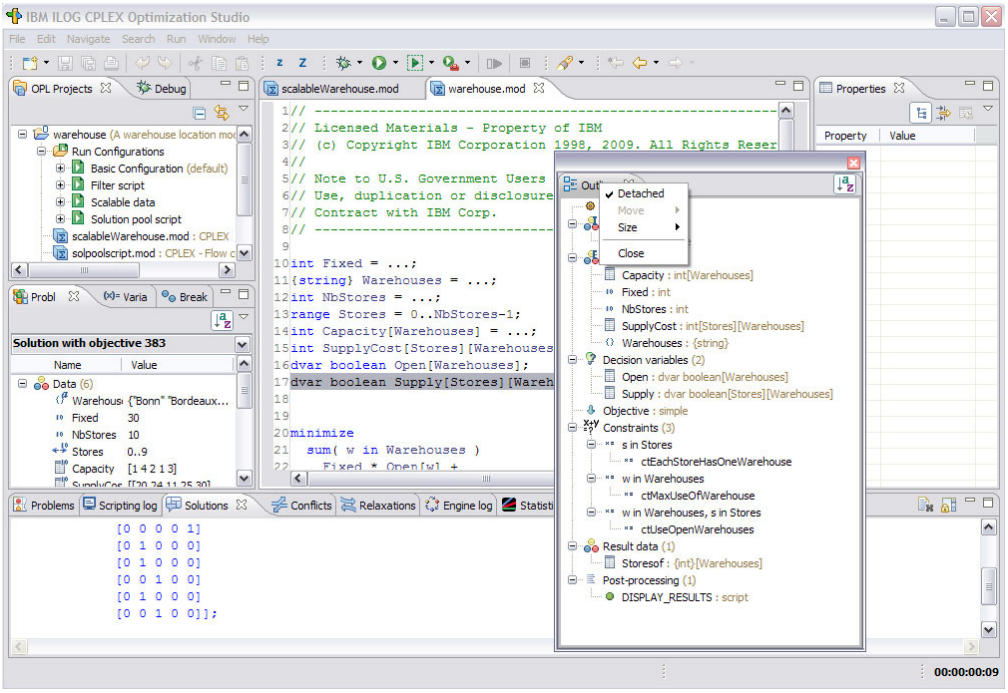
  When solving MIP problems several solutions may be displayed in this drop-down list. Choosing one of them displays data for that solution in the lower part of the Problem Browser.
- Tooltips show data that is too wide to display.
- Double-clicking an item opens an editor for that item.

Chapter 6. Resizing, moving, hiding, and restoring IDE views

You can customize the appearance of the CPLEX Studio IDE, or restore the original appearance.

All views are resizable, movable, and detachable. *Movable* means that you can drag a view from one location and drop it in another, even within another view. For example, you could drag the Outline view into the Output Area, and it would become another tab there.

Detachable means that you can drag a view outside the IDE frame and it becomes its own standalone window. To put the view back into the frame you need to right-click in the tab area of the detached view and deselect the **Detached** item in the menu.



When you do this, the detached view will return to the bottom frame. To restore the detached view to its default position, select **Window** > **Reset Perspective**.

Resizing views

To resize views, click one of its borders or corners and drag it to the size you want.

Or, to temporarily expand a view to the full size of the IDE frame, double-click its tab. Double-click again to shrink it back to original size.

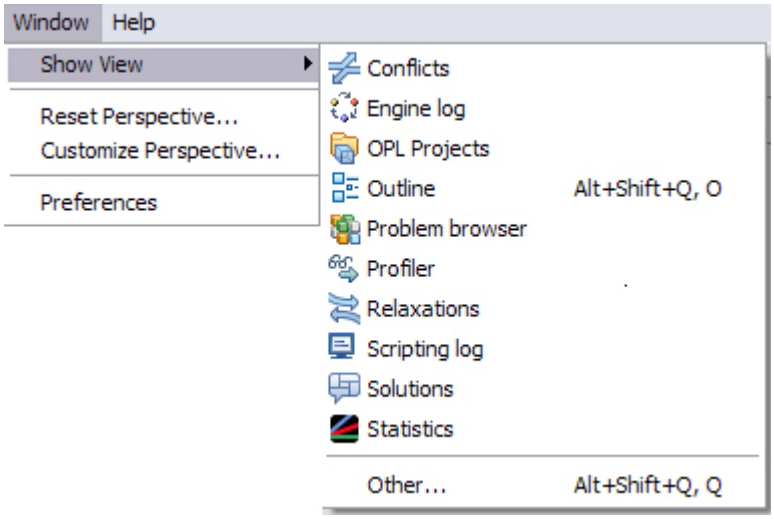
Moving views

To move a view, right-click its top border and drag it to the new location. A black border appears as you drag the view around the frame, to tell you where the view will be placed if you release the mouse button in that location.

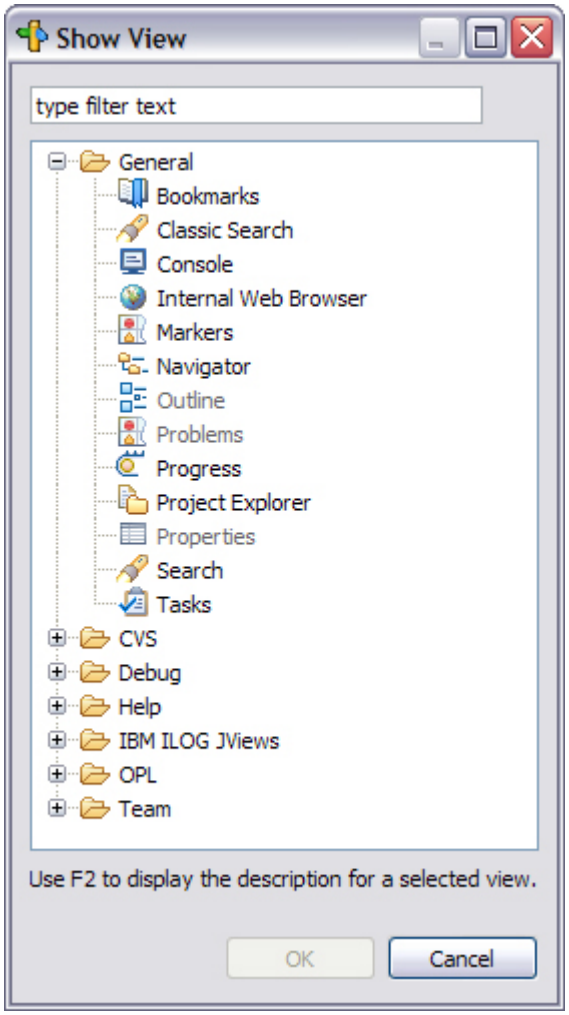
Hiding views

To hide a view, click its Close box .

To display a view that has been closed, in the main menu choose **Window>Show View** and select the name of the view you want to display:



If the view you want to display is not shown in that menu, click **Other** to display more views:



Restoring views

The menu command **Window** > **Reset Perspective** resets the current perspective to its original views and layout (the positions of the various windows, what information is visible or not).

Chapter 7. Working with files in CPLEX Studio

Shows how to open, edit, and work with files in the CPLEX Studio IDE.

Adding existing files to a project

How to add files to your OPL project.

You can use the **File** > **Copy Files to Project** command to open a dialog box that allows you to open files and import them into selected projects.

You can also drag existing files from a Windows Explorer window and drop them onto the project folder in your OPL Projects Navigator. This is always a *copy* operation, not a move.

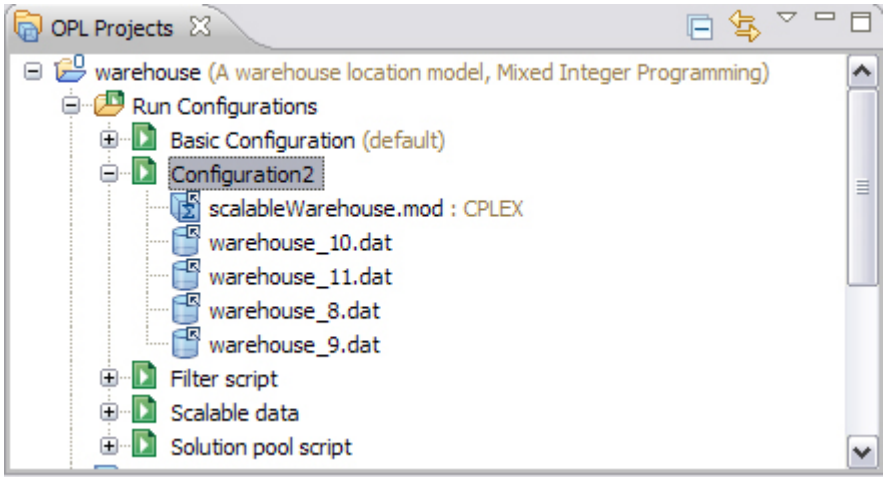
Ordering files within a run configuration

How to specify the order of data or settings files in a run configuration.

About this task

When you execute a run configuration, the order of the data or settings files relative to each other is important. Since some data in a `.dat` file may depend on other data in a different `.dat` file, if the data files are in the wrong order it may cause an error at execution time.

As an example of this, look at the following screen capture of a run configuration, **Configuration2**:

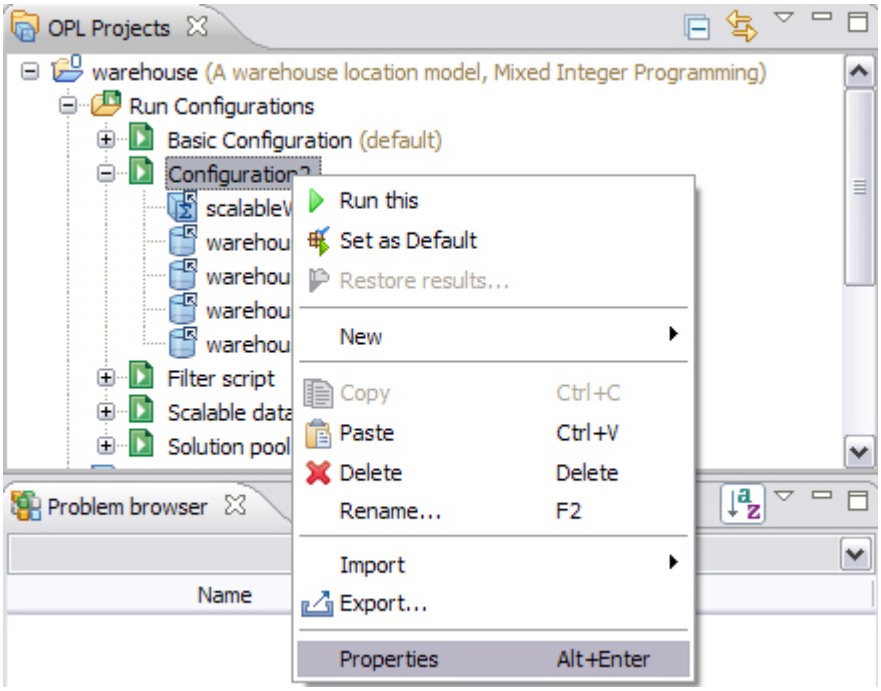


In **Configuration2**, the four data files are intended to be executed in numerical order. However, as you can see, they are sorted in ASCII order. This would cause them to be executed in the wrong sequence.

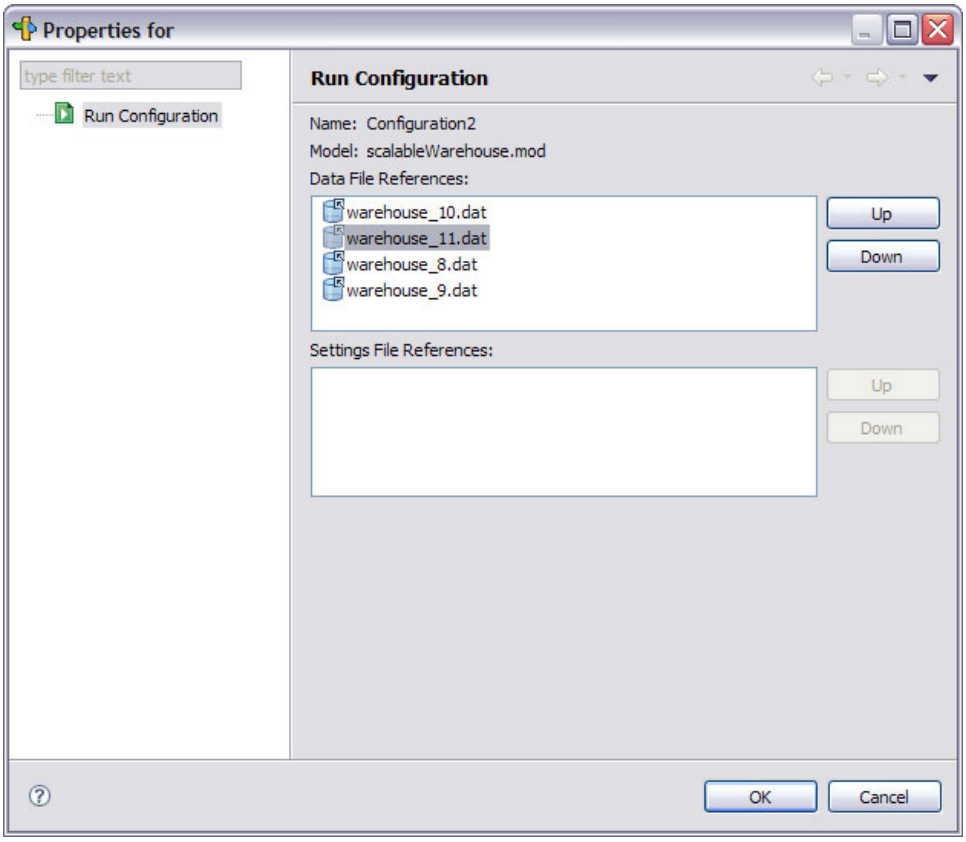
You can set the order of multiple data or settings files in a run configuration using the following procedure.

Procedure

1. Right-click the configuration name and choose **Properties** from the menu:



2. A properties window appears for the run configuration:



You can use the **Up** and **Down** buttons to rearrange the order of your data files and settings files.

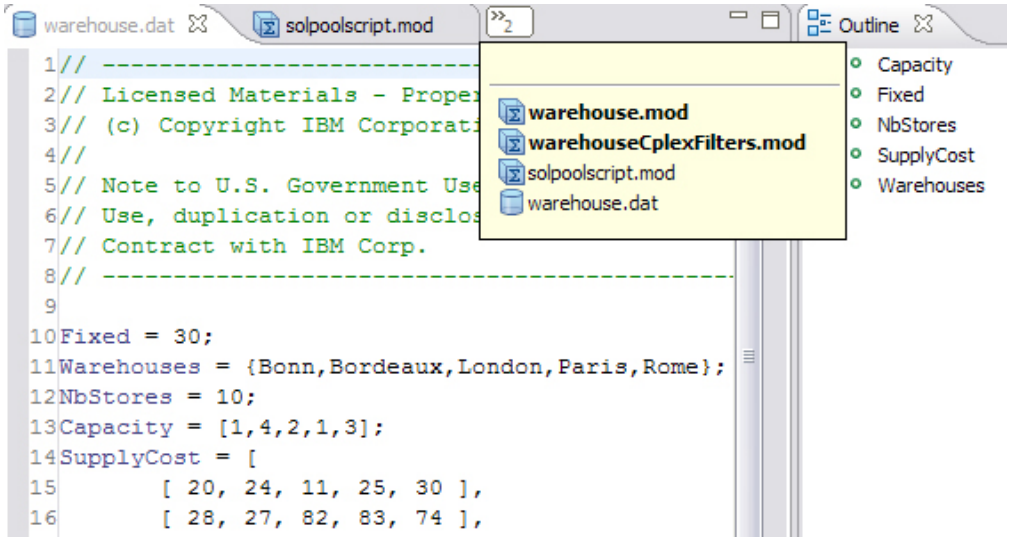
Opening files for editing

Shows how to open and edit your files in the OPL IDE.

In general, you open your model files, data files and settings files for editing by double-clicking the file in OPL Projects Navigator.

Several editors can be open in the Editing Area at once. You can switch back and forth between the open views by clicking the tabs of the views that are visible in the Editing Area.

If so many edit views are open that all of their tabs cannot be displayed, a "more views" icon  becomes visible. Click it and a list of the other views appears:

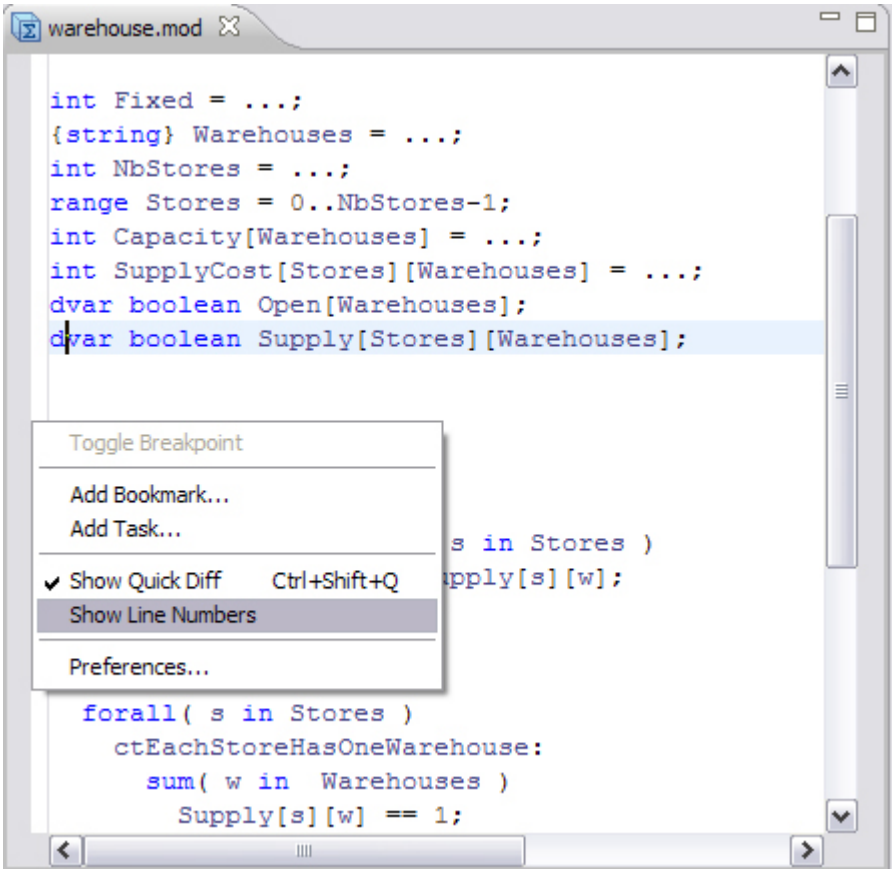


Click any of the views in the popup list to view them.

Line numbers in the Editor

By default, line numbers appear in the text editor. However, you can hide them in one of the following ways:

- From the menu bar select **Window** > **Preferences**, then **General** > **Editors** > **Text Editors**. Clear the box for **Show line numbers**.
- Inside the Editing Area, right-click and select **Preferences**, then **General** > **Editors** > **Text Editors**. Clear the box for **Show line numbers**.
- Inside the Editing Area, right-click in the left margin to display the popup that allows you to deselect the option **Show line numbers**.



Opening external files in the Editor

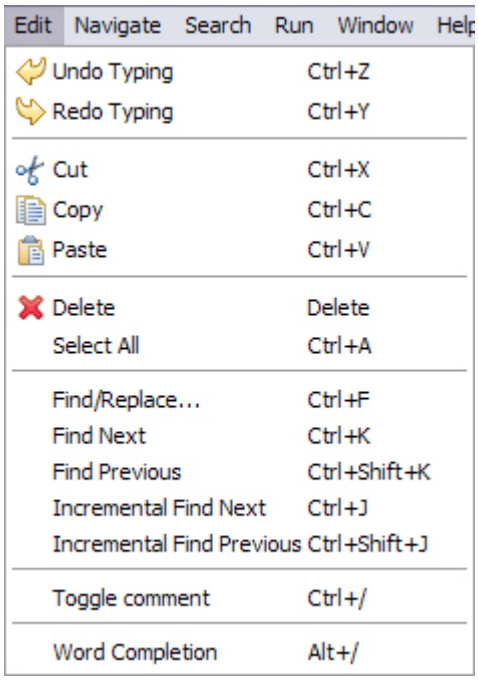
To open files that are not currently in any open project in OPL Projects Navigator, use the **File** > **Open File in Editor** menu command.

This does not add the files to any project, but allows you to view or edit them in the Editing Area. To add external files to a project, see the "Adding existing files to a project" on page 25 section.

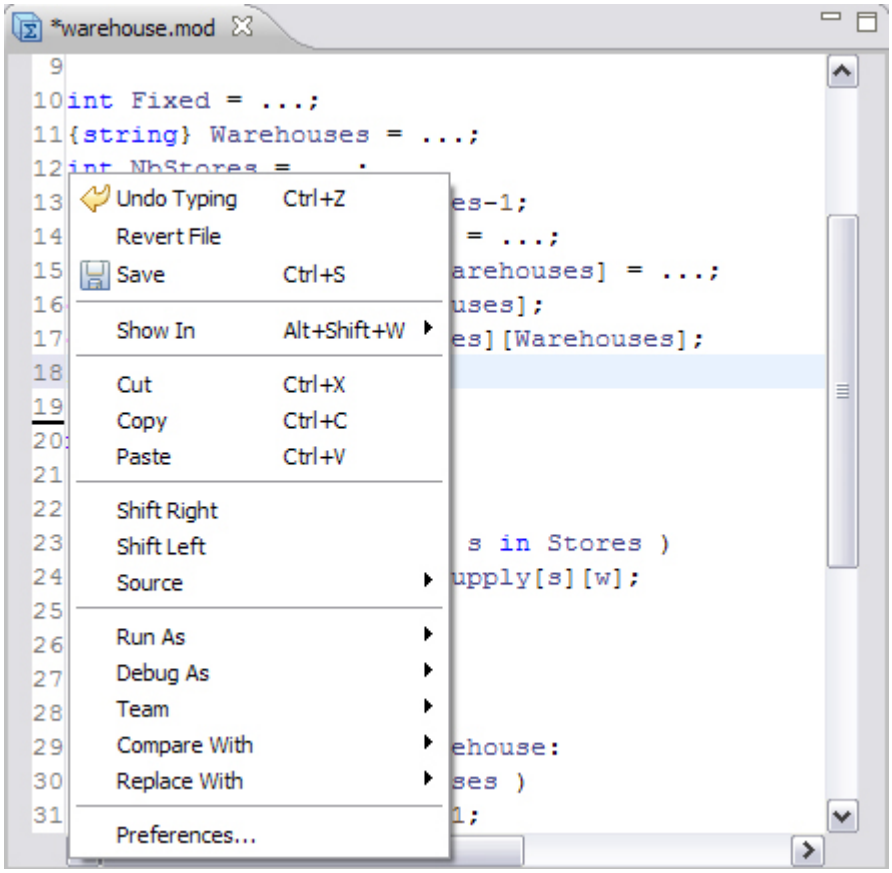
Editing commands and shortcuts

The common commands you need to use while editing can be found on two menus:

- The main menu **Edit** menu:



- The right-click menu within the editor itself:



Local History and its related features

How to track and compare different versions of your files as you edit them in the CPLEX Studio IDE.

A limited form of version control called **Local History** allows you to track and compare different versions of your files as you edit them.

For example, if you edit the same model file several times, all versions of the file are still available to you. You can use the **Compare With** and **Replace With** commands to compare different versions of a file or revert to previous versions of a file or its contents.

Each of these features is described in the following sections.

'Compare With' features

How to compare files with each other and with Local History.

Before you begin

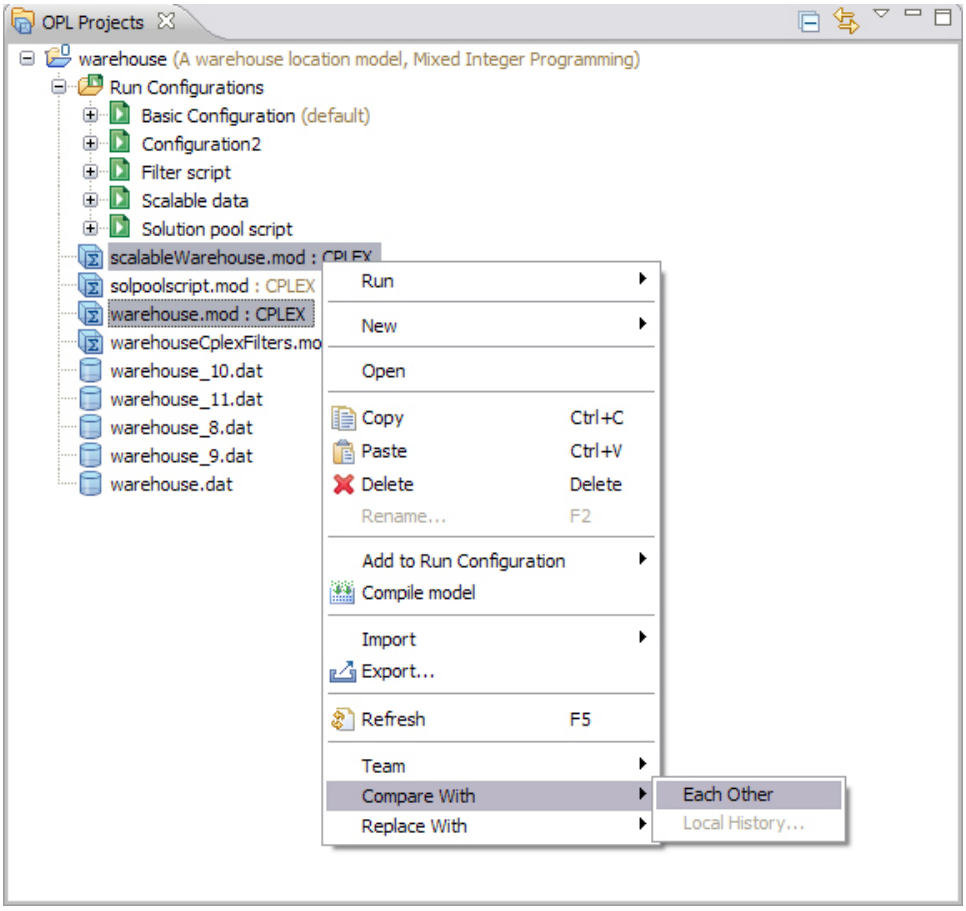
- "Compare With Each Other"
- "Compare With Local History" on page 31

Compare With Each Other

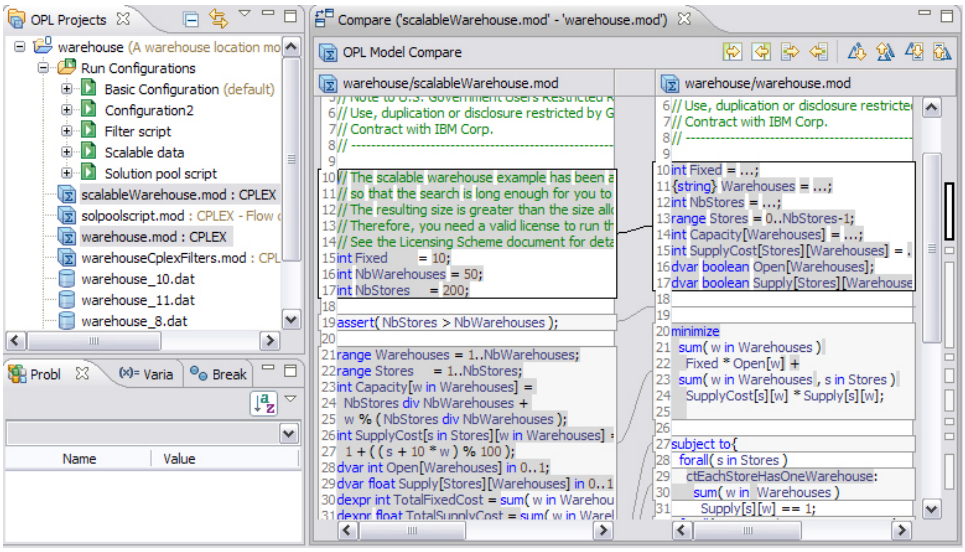
Procedure

To compare two files in the same project with each other:

1. In the OPL Projects Navigator, highlight the two files you want to compare, right-click and choose **Compare With** > **Each Other**.



2. The files are opened in the Editing Area in a special view that allows you to see them side by side, with the differences between the two files highlighted:

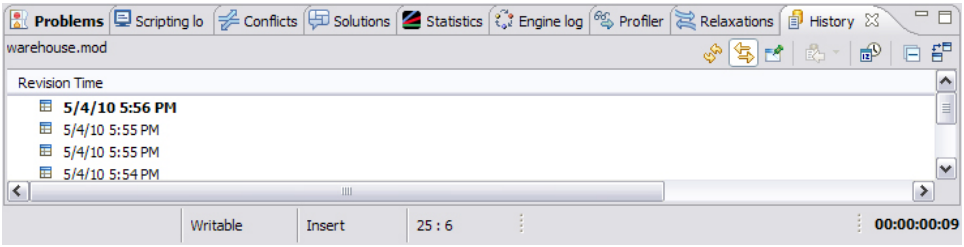


Compare With Local History

Procedure

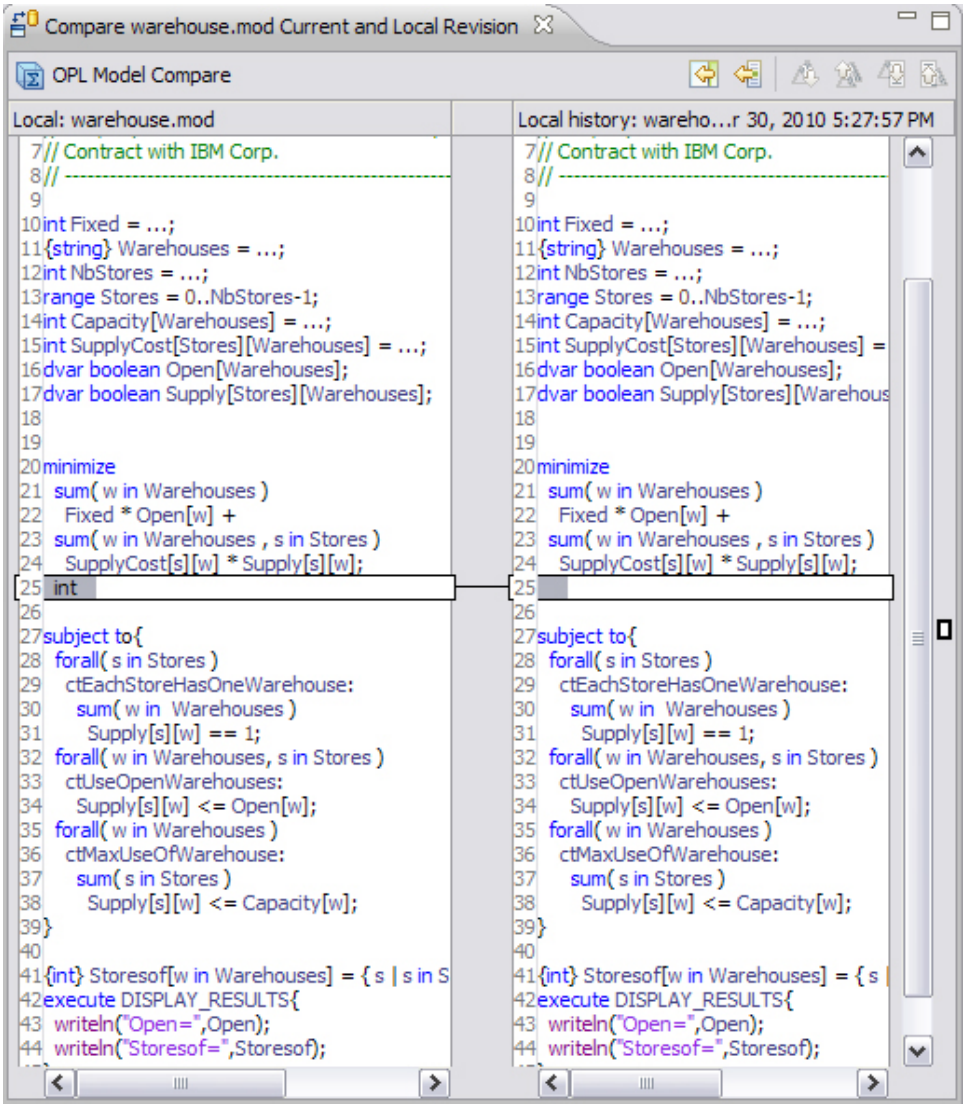
To compare a file with another version of itself in Local History:

1. In the OPL Projects Navigator, highlight the file you want to compare with its own Local History versions, right-click and choose **Compare With** > **Local History**.
2. A list of the different versions of the file is displayed in the **History** tab.



3. Double-click the version of the file that you want to compare with the current version.

   The files are opened in the Editing Area in a special view that allows you to see them side by side, with the differences between the two versions highlighted.



'Replace With' features

How to compare files with each other and replace the contents from other versions of the file in Local History.

Before you begin

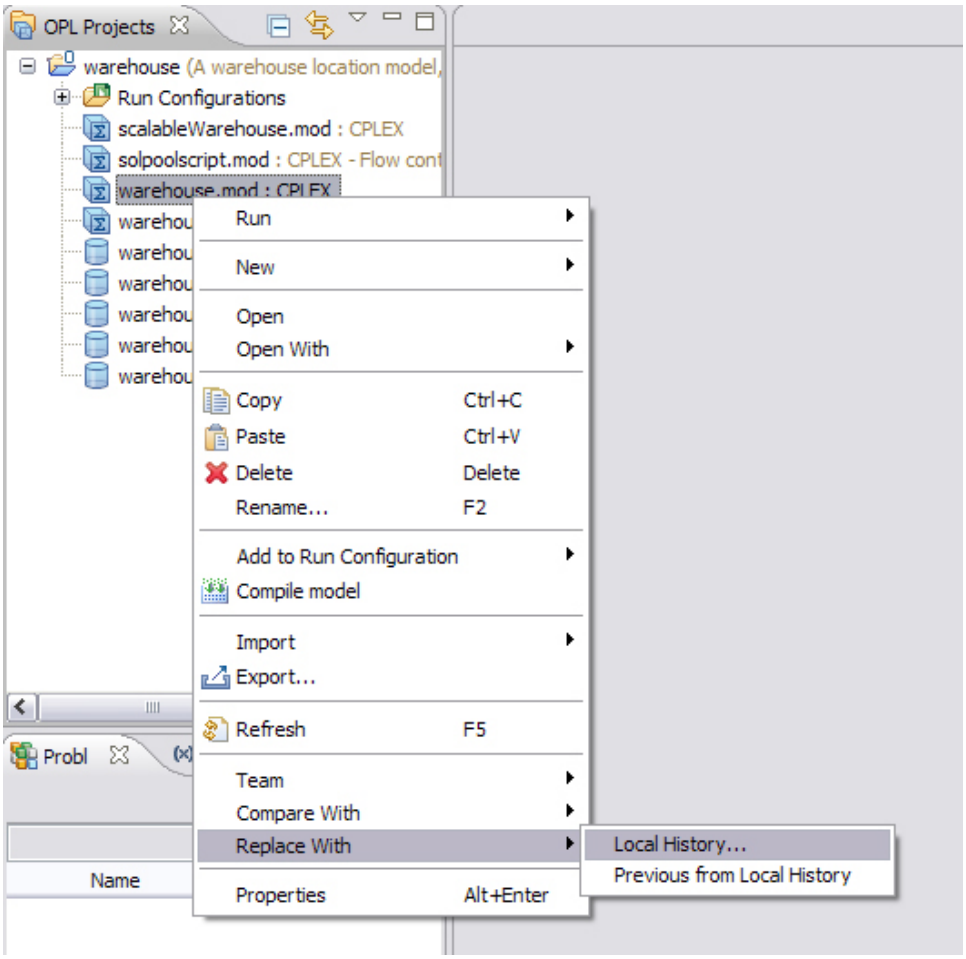
- "Replace With Local History"
- "Replace With Previous Version" on page 34

Replace With Local History

Procedure

To replace a file with a selected version of itself from Local History:

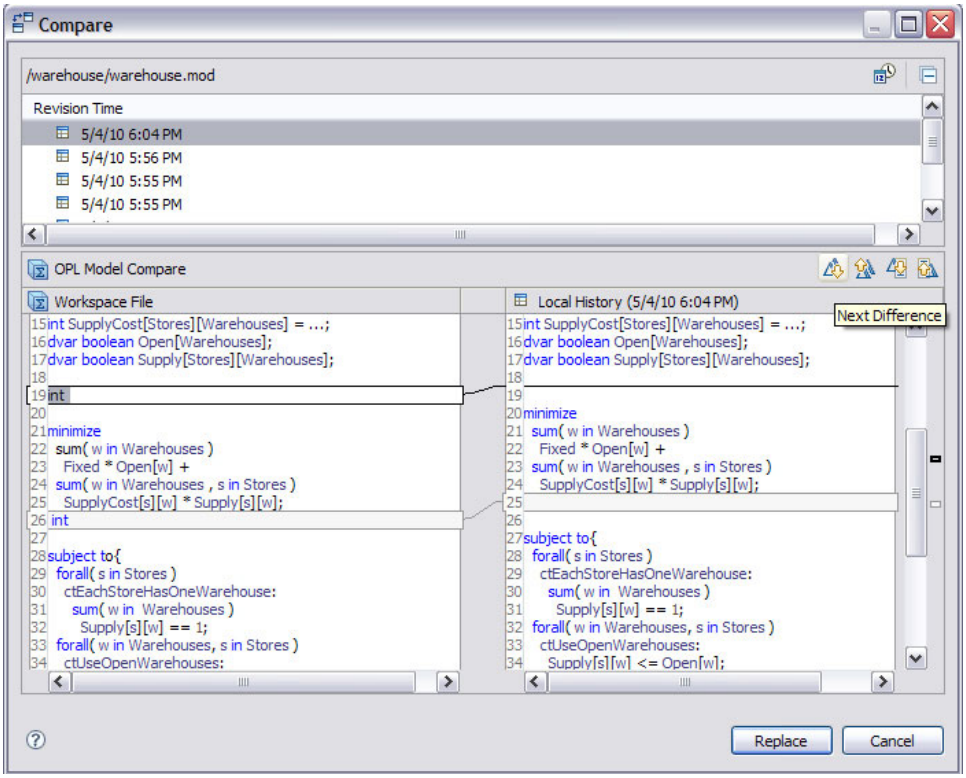
1. In the OPL Projects Navigator, highlight the file you want to revert to one of its Local History versions, right-click and choose **Replace With** > **Local History**.



2. A popup window appears in which you can view the different versions of this file in Local History.

   Double-click the version of the file that you want to compare with the current version and use as a candidate for replacement.

   The files are opened in a special view that allows you to see them side by side, with the differences between the two versions highlighted.



You can use the icons that appear in the **Text Compare** area to move to the next and previous differences in the files, and thus determine whether you want to replace the contents of the current file with the contents of the version you are comparing it to.

Replace With Previous Version

Procedure

To replace a file with its previous version in Local History:

1. In the OPL Projects Navigator, highlight the file you want to revert to its previous version, right-click and choose **Replace With** > **Previous Version**.
2. The file is automatically replaced with the version of the file from Local History that immediately precedes it. This creates a new version in Local History.

Chapter 8. Executing OPL projects

Describes the different ways of running and browsing OPL projects.

The Run options

Explains how to use the menus to run projects in the CPLEX Studio IDE.

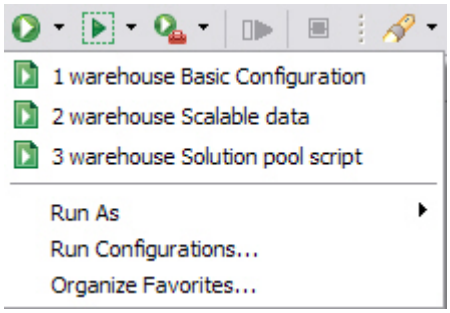
You can solve an OPL model by clicking the **Run** button in the execution toolbar.

You can also use right-click menus.

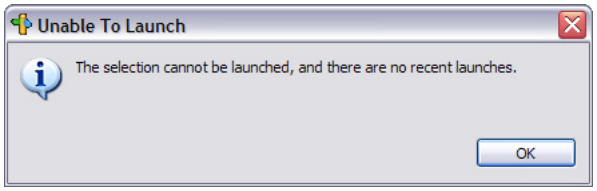
How the Run button works

The behavior of the **Run** button in the execution toolbar depends on your run history.

- As runs are executed, they are added to a numbered list that is visible by clicking the arrow button to the right of the **Run** button.



- If you have just launched CPLEX Studio and no OPL model has been run yet, clicking the **Run** button for the first time may produce the following message:



  In this case, you should try again by right-clicking the run configuration name (left panel) and selecting **Run this**.

- Once the list is populated, clicking the **Run** button launches *the most recently launched run configuration in the list*, no matter what project is selected in the OPL Projects Navigator.

  Note:

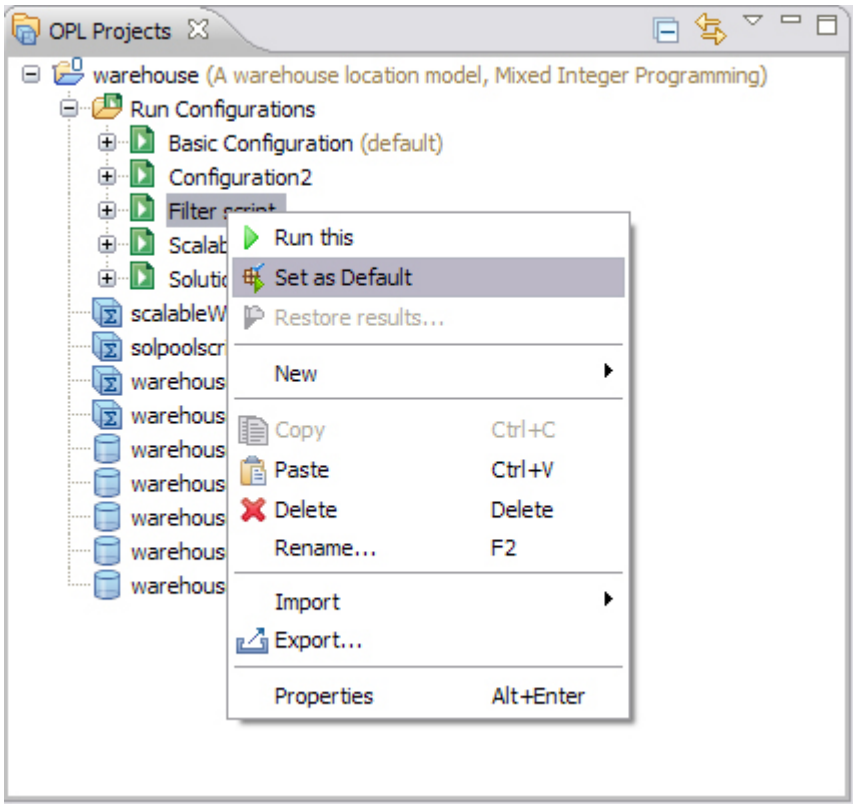
Obviously, this does not make it possible to just click a project in the Projects Navigator and launch its default run configuration by simply clicking the **Run** button. For this reason, many OPL users prefer the right-click menus to launch their models.

The default behavior of the **Run** button is configurable. See The execution toolbar for more information.

How the Run menus work

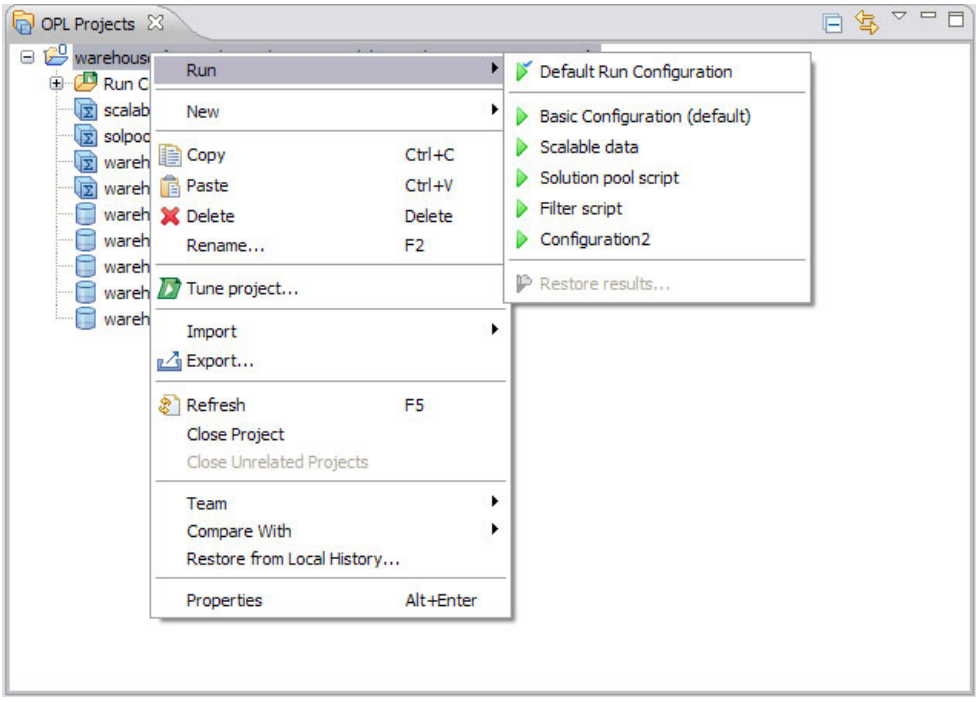
There are additional ways to run your projects in CPLEX Studio. This section describes the menu options that can be used to launch your projects directly from the OPL Projects Navigator.

To set a default run configuration, right-click a run configuration in the project folder and select **Set as Default**.



To run your projects from the OPL Projects Navigator:

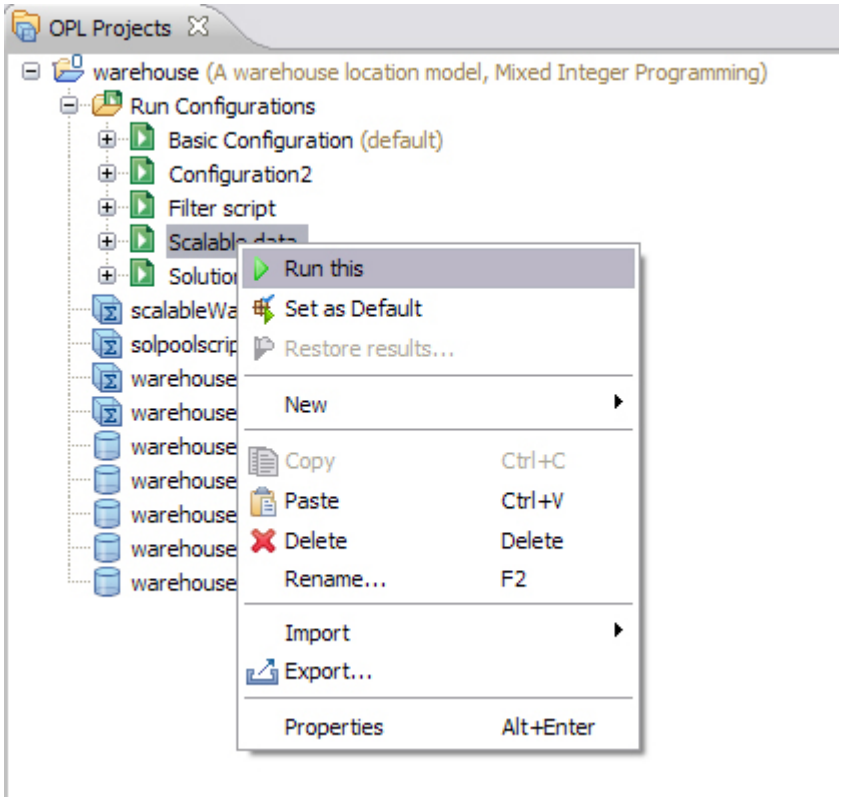
- If you right-click the project folder, you see the following menu:



  Two run options are listed:

  - **Run** > **Default Run Configuration** — this option executes the run configuration that is currently set as the default for this project.
  - **Run** > **<list_of_run_configurations** > — this option displays *all* run configurations for the project, so that you can choose which one you want to launch, whether it is currently the default run configuration or not.

- If you right-click an individual run configuration for a project, you see the option **Run this**, which enables you to run only that run configuration.



The Status Bar

Describes the area that displays messages about the current execution status of the IDE and information about files being edited.

As you solve, this area displays the execution status of projects being solved and the elapsed time of the solve.

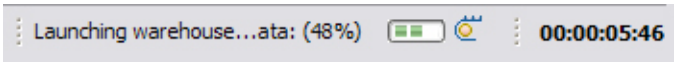


Figure 1. Status Bar (partial view during a solve)

The Status Bar also shows the status of documents being edited in the Editing Area, and the current line and column number of the cursor.

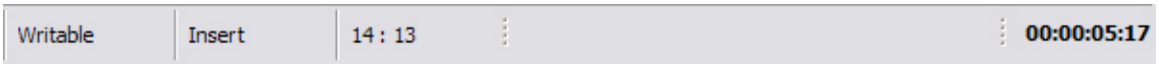


Figure 2. Status Bar (partial view while editing)

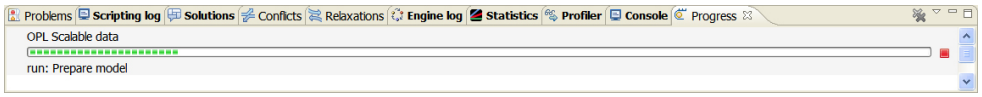
The middle part of the Status Bar indicates that the file currently being edited is **Writable** (as opposed to **Read-Only**) and that the editor is in **Insert** mode (as opposed to **Overwrite** mode). The numbers indicate the line number and column number of the current cursor location in the file.

The box at the right of the Status Bar displays an animated graphic while a project is running, and a message is displayed beside the graphic indicating progress.

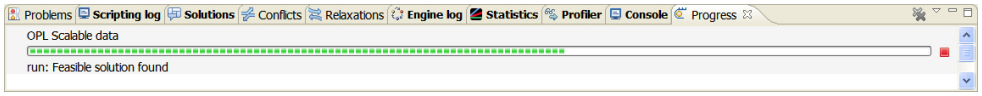
Run progress messages in the Status Bar

Whether the model is running in standard run mode or in debug, browse, or background mode, you can see messages about the progress of the run in the IDE Status Bar:

- When the solve begins, a **Launching <run configuration name** > message appears at the right of the Status Bar.
- As the solve progresses, the message changes to **<run configuration name** > **<percent** > , with a percentage displayed to indicate progress.
- When the solve is finished, the message changes to **<run configuration name** > **100%**, to indicate that the run has completed
- In addition, as the model is solving, the run indicator  becomes animated.
- If you click the **Shows background operations** icon  at the extreme right of the toolbar, a **Progress** tab appears in the Output Area. If the solve is still running, you see a display similar to this:



When the solve has finished, you see a display similar to this:



The execution toolbar options

Explains how to use the menus to run projects in the CPLEX Studio IDE.

In addition to the Run menu in the OPL Projects Navigator, there are buttons in the execution toolbar that can be used to run your projects in different ways. These are explained in the *IDE Reference Manual*; see The execution toolbar.

Executing projects in a separate process

You can launch OPL run time in a separate process and have up to 2 GB available for large models.

- **Memory management**

  The CPLEX Studio IDE and the execution of OPL projects are in different processes. This enables the IDE and the project execution to go up to the theoretical maximum heap size of the machine, depending on its memory resources (32 or 64-bit OS, or available RAM). Theoretically, on a Windows 32-bit system, processes can use up to 2 GB of memory.

  Note:

  If you use the PAE (Physical Address Extension) feature of Windows 32-bit systems (also known as the LARGEADDRESSAWARE switch or the /3GB parameter), neither of the 2 OPL processes will be able to use 3 GB of memory. The extra 1 GB of memory does not increase the theoretical maximum size of the

IBM® Java™ heap, but does allow the IBM Java heap to grow closer to its theoretical maximum size (2 GB - 1 byte), because the extra memory can be used for the native heap.

Information about the memory limitations of Windows can be found on the following pages:

http://www.microsoft.com/whdc/system/platform/server/PAE/PAEmem.mspx

http://www.microsoft.com/whdc/system/platform/server/PAE/pae_os.mspx

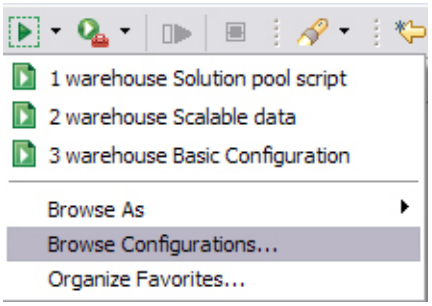
http://support.microsoft.com/kb/291988

- **Security aspects with respect to firewalls on Windows systems**

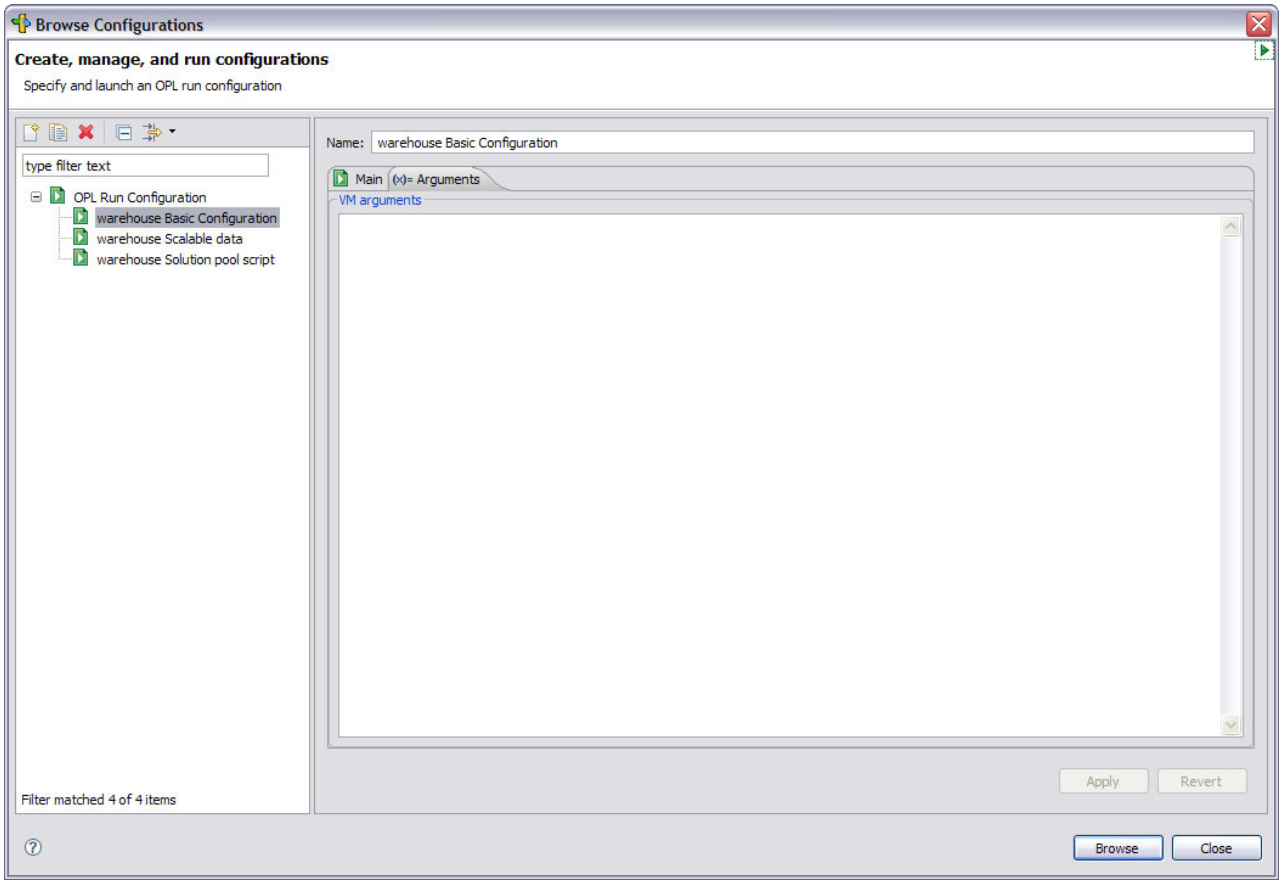
  Because the CPLEX Studio IDE and the model execution are in different processes, at the first launch of the IDE, the firewall can display a warning about security, which comes from the connection between the two processes. To get rid of this warning, you should choose to allow this connection forever. If you do not accept the connection, you will not be able to execute your configurations.

- **Configuration of the spawned solving process**

  The Java spawned process can be configured using Java options. To configure the spawned process, open the **Run Configurations** panel via the menu:



and select the Arguments tab, where you can add your options.


Browse Configurations
Create, manage, and run configurations
Specify and launch an OPL run configuration
type filter text
OPL Run Configuration
warehouse Basic Configuration
warehouse Scalable data
warehouse Solution pool script
Filter matched 4 of 4 items
Name:
warehouse Basic Configuration
Main
(x)= Arguments
VM arguments
Apply
Revert
Browse
Close

Part 2. Getting Started Tutorial

A tutorial in which you launch the IDE, create an empty project, enter an OPL model, add data, add a settings file, create run configurations and execute them. More tutorials elaborate on IDE features in *IDE Tutorials*.

Chapter 9. Prerequisites - before you start

Before you start the CPLEX Studio IDE.

At this stage, it is assumed that you have already successfully installed IBM ILOG® CPLEX Studio on your platform.

At this point you are ready to launch the CPLEX Studio IDE (or the IDE for short) as explained in Chapter 1, "Launching the CPLEX Studio IDE," on page 3.

Once the IDE is open, you can read:

- Tour of the Graphical User Interface in the *IDE Reference* to discover the graphical user interface.
- Chapter 10, "Creating a project," on page 47 if you feel familiar enough with the interface and want to start working on a project immediately.

Other useful documents to read are:

- A quick start to CPLEX Studio for an introduction to how the OPL language handles optimization problems.
- Part 1, "Introduction to the CPLEX Studio IDE," on page 1, the first section of this manual.
- Navigating in the documentation set for details of prerequisites, conventions, documentation formats, and other general information.

Chapter 10. Creating a project

Walks you through creating a project file, adding model and data, and setting mathematical programming options.

Purpose

What you are going to do in the tutorial.

After launching the IBM ILOG CPLEX Studio IDE as described in Chapter 1, "Launching the CPLEX Studio IDE," on page 3, you will want to solve a problem of your own. For this, you will first have to define a working document in the IDE. You can do this either by editing an existing project (see Reusing existing files and projects in the *IDE Reference*) or by starting your own project.

In this tutorial, you will start your own project. To do this, you will:

- Create a project: see "Creating an empty project" on page 49.
- Add an existing model or write a new one: see "The pasta production example."
- Add one or more existing data files or write new ones: see "Adding data" on page 55.
- Set mathematical programming options, if applicable: see "Changing an MP option value" on page 64.

Once your project is built, you will populate a run configuration, execute it, and study the results as explained in Chapter 11, "Executing a project," on page 59 and Chapter 12, "Examining a solution to the model," on page 69.

The pasta production example

Presents the production problem and shows the code for the model and data.

You could write your own model from scratch by following the syntax rules from the Language Reference Manual and the Language User's Manual, but since this tutorial does not aim at teaching you the modeling or scripting languages, you are going to reuse the pasta production example, described in A production problem in the *Language User's Manual*, for the purpose of this exercise.

Note:

The pasta production model is designed to be solved by the CPLEX engine. However, the content of this section would be the same for a model solved by the CP Optimizer engine, except where explicitly mentioned.

The problem is as follows. To meet the demands of its customers, a company manufactures its products in its own factories (*inside* production) or buys the products from other companies (*outside* production).

The inside production is subject to resource constraints: each product consumes a certain amount of each resource. In contrast, the outside production is theoretically unlimited. The problem is to determine how much of each product should be

produced inside the company and how much outside, while minimizing the overall production cost, meeting the demand, and satisfying the resource constraints.

The code extract below (product.mod file) shows an OPL model (the tuple version) for this example. This model is part of the production project, which is available at the following location:

<Install_dir>\opl\examples\opl\production

where <Install_dir> is your installation directory.

OPL model for the production planning example (product.mod)

```
{string} Products = ...;
{string} Resources = ...;
tuple productData {
   float demand;
   float insideCost;
   float outsideCost;
   float consumption[Resources];
}
productData Product[Products] = ...;
float Capacity[Resources] = ...;

dvar float+ Inside[Products];
dvar float+ Outside[Products];

execute CPX_PARAM {
  cplex.preind = 0;
  cplex.simdisplay = 2;
}


minimize
  sum( p in Products )
    (Product[p].insideCost * Inside[p] +
    Product[p].outsideCost * Outside[p] );
subject to {
  forall( r in Resources )
    ctInside:
      sum( p in Products )
        Product[p].consumption[r] * Inside[p] <= Capacity[r];
  forall( p in Products )
    ctDemand:
      Inside[p] + Outside[p] >= Product[p].demand;
}
```

The following code extract (product.dat file) shows the data declaration for the problem.

OPL data for the production planning example (product.dat)

```
Products =  { "kluski", "capellini", "fettucine" };
Resources = { "flour", "eggs" };
Product = #[
      kluski : < 100, 0.6, 0.8, [ 0.5, 0.2 ] >,
      capellini : < 200, 0.8, 0.9, [ 0.4, 0.4 ] >,
      fettucine : < 300, 0.3, 0.4, [ 0.3, 0.6 ] >
          ]#;
Capacity = [ 20, 40 ];
```

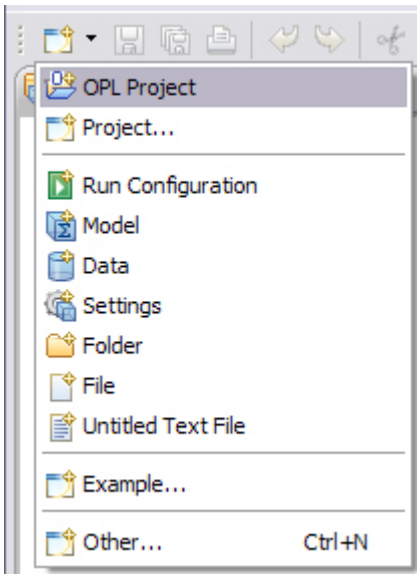
Creating an empty project

Walks you through creating a project “from scratch” and defining a model using the editing capabilities of the IDE.

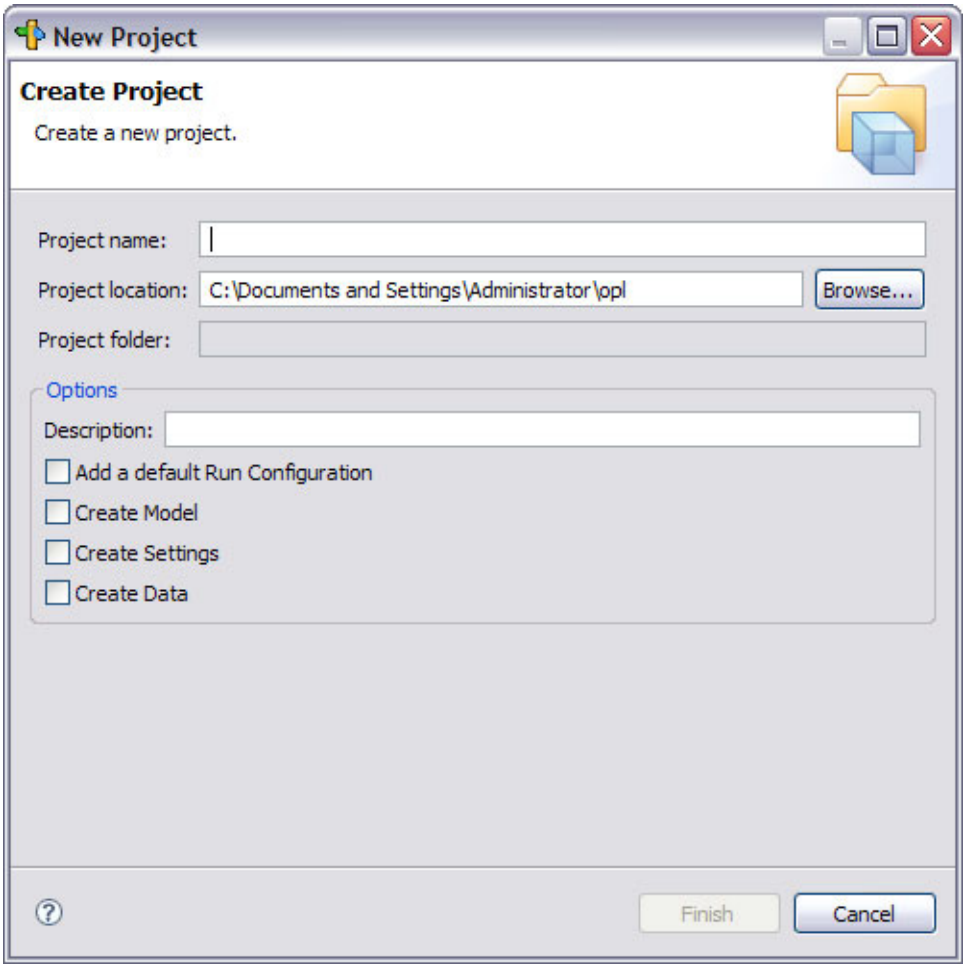
Procedure

To start from an empty project:

1. In the main menu, choose **File** > **New** > **OPL Project**, or click the **New** icon and select **OPL Project**.



2. The New Project wizard is displayed.



3.

Enter the following information in the New Project window:

- Type `myFirstProject` as the **Project Name** of your new project.
- Enter a destination **Project Location** for the project, *other* than the OPL examples directory. For example, create a directory named `C:\OPL_projects` and browse to select the directory.

  A folder with the project name is created in this directory.
- In the **Options** field:
  - Enter a **Description** for the project.
  - Check all the boxes except Create Data, because you will later be adding existing data files to the project rather than creating an empty one.

Your window should look similar to the one shown below.

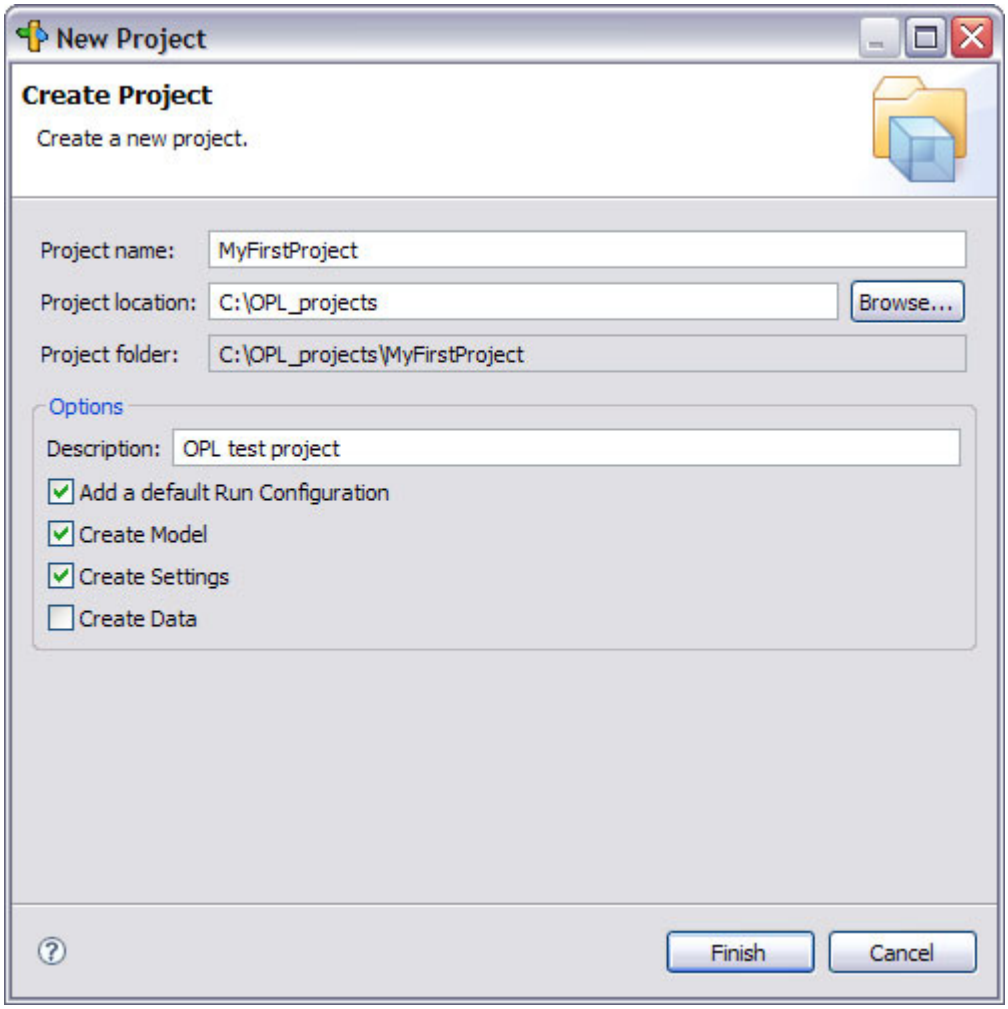


Figure 3. Creating a project

4. When you have entered all the information, click **Finish**.

   Note:

   If a project with the same name already exists in CPLEX Studio, a message warns you and the **Finish** button remains greyed out as long as you don't enter a unique name.

   The project is created, containing the model and settings files you specified, and appears in the OPL Projects Navigator. The new empty model file is displayed in the Editing Area:

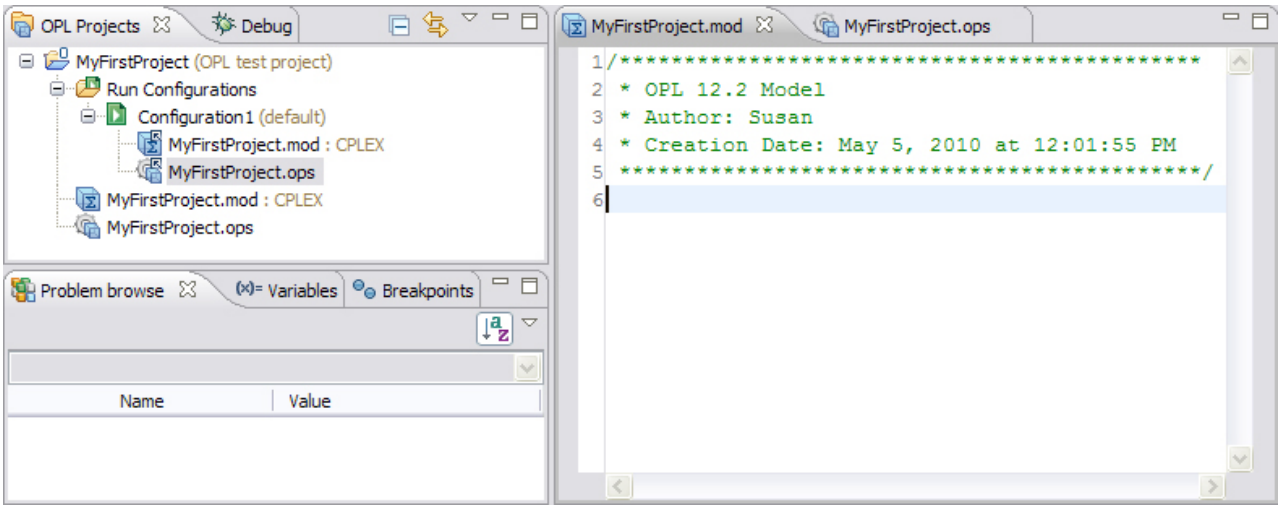


Figure 4. New project and new empty model in main window

Note that the `.mod` and `.ops` extensions are automatically appended to the file names in the OPL Projects Navigator and in Windows Explorer (see "File types" on page 15 and Understanding OPL projects).

Results

The OPL Projects Navigator displays a minimal tree containing:

- an empty `.mod` file: you are going to fill it with OPL statements in the next step, *Adding the model*
- an `.ops` file containing the default values for MP and CP options, and OPL settings: see "Changing an MP option value" on page 64 in this manual and Setting programming options in the *IDE Reference*.
- the **Run Configurations** folder
- one run configuration (set to default) containing the model and settings files you have just created (see Figure 4).

Important:

The only mandatory component in a project or run configuration is a valid model file. A project can contain more than one model, but a run configuration can contain only one.

See also The main window in the *IDE Reference* for reference information on the graphical user interface.

Adding the model

Walks you through entering a model in an OPL project.

About this task

You are going to copy-paste the pasta production model into the Editing Area.

Procedure

To add the pasta production model to the project:

1. Choose **File** > **Open File in Editor**, and browse to
   `<Install_dir>\opl\examples\opl\production\product.mod`
   then double-click the filename in the dialog box or select it and click **Open**.

   Note:

   There are two model files in this project. Make sure that you open `product.mod` and not `production.mod`.

   The OPL statements of the `product.mod` file appear in the Editing Area, in a separate window. If you click the tab of the `myFirstProject.mod` file, you can see that your empty model is in a different editor.
2. Click in the `product.mod` window and press **Ctrl-A** to select all the text, then **Ctrl-C** to copy the contents of the file.
3. Click the tab of the `myFirstProject.mod` file to redisplay the empty editing window for your own project.
4. Place your cursor after the header comments and press **Ctrl-V** to paste the copied statements into the file `myfirstproject.mod`.
   Note that an asterisk (*) appears in the tab of this window.

   This indicates that the file is unsaved. Click the **Save** button  to save the file.
5. Close the open window for the `product.mod` file in the Editing Area by clicking the button to the right of its tab. Leave the edit window for the `myFirstProject.mod` file open for the next exercise.

Dealing with errors

Describes how to take advantage of the automatic error detection feature.

About this task

For this very first start with an OPL model, you have copied and pasted an existing model for quicker results. In your real business life, however, you will enter OPL statements from the keyboard. By default, the IDE checks for syntax and semantic errors automatically as you type and error messages are displayed in the **Problems** tab at the bottom of the IDE.

Procedure

To observe the default behavior:

1. In the second line of `MyFirstProject.mod` displayed in the Editing Area, remove the “s” from the end of the word “Resources”.

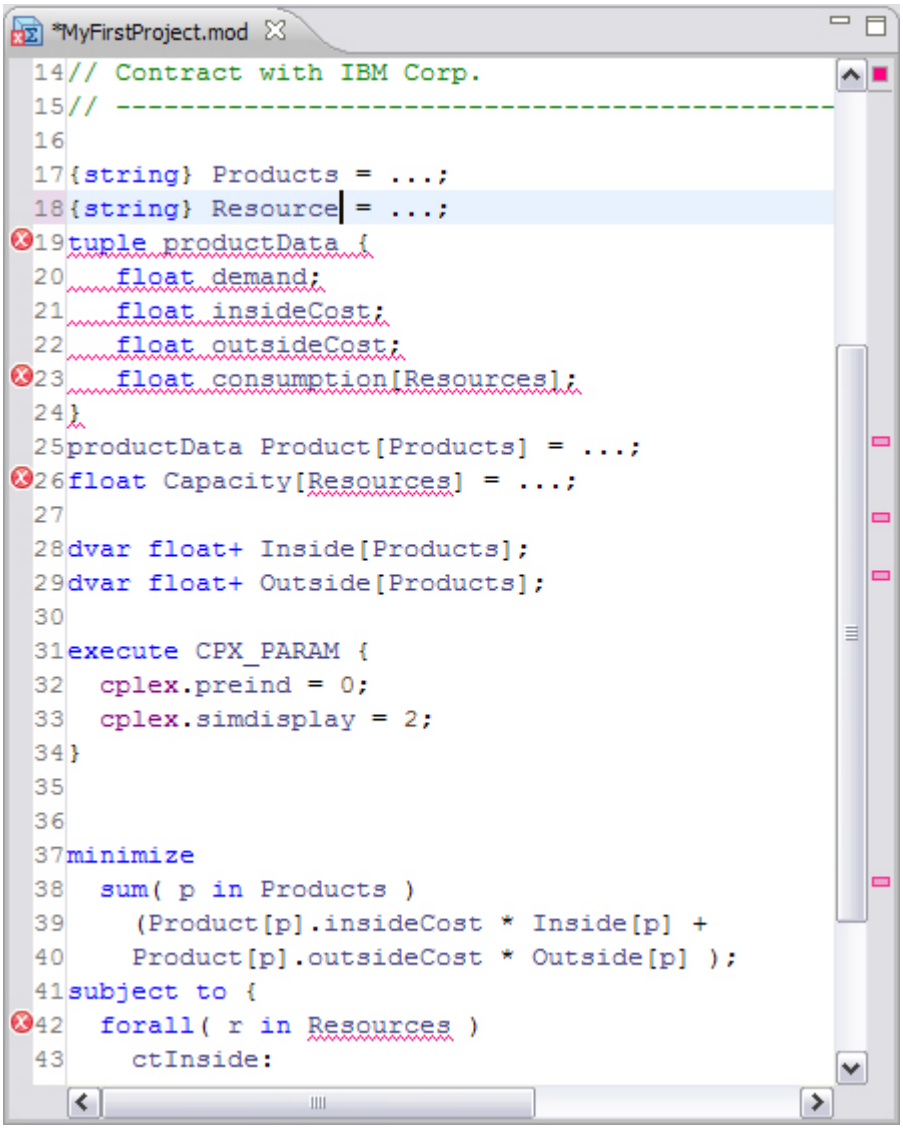


Figure 5. A syntax error

The line containing the error is highlighted and for this line any other line affected by the error, a red error symbol appears in the margin  . The **Problems** tab immediately displays the corresponding error messages, indicating the description, location, and source.

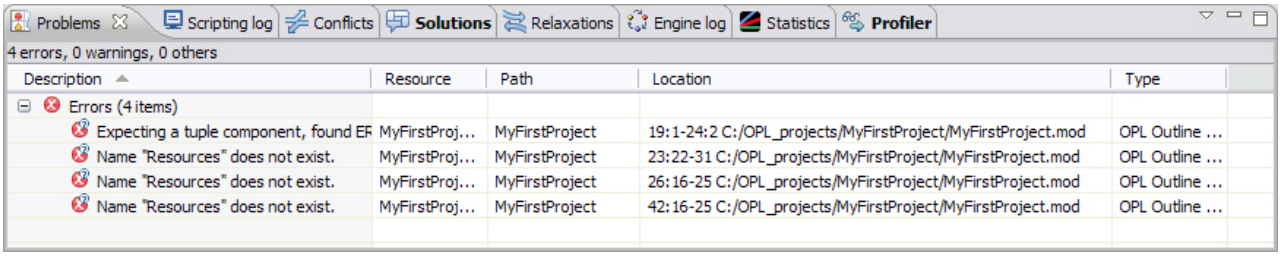


Figure 6. Problems tab

2. Remove the mistake. The error message disappears and the indicators in the editor disappear.
3. Press **Ctrl+S** or choose **File** > **Save** or press the **Save** button  in the standard toolbar to save your work.

Results

In the next step, you will add two data files to the project.

Adding data

Explains how to add a data file to an OPL project and fill it with data.

Before you begin

This part of the tutorial assumes you have created a project and is meaningful only if the model is not empty.

About this task

You can add more than one data file to a project or run configuration. If you do so, their order is meaningful. See “Ordering files within a run configuration” on page 25. You can either add existing data files or create them as you add them. In this tutorial, you are going to add two existing data files to your project.

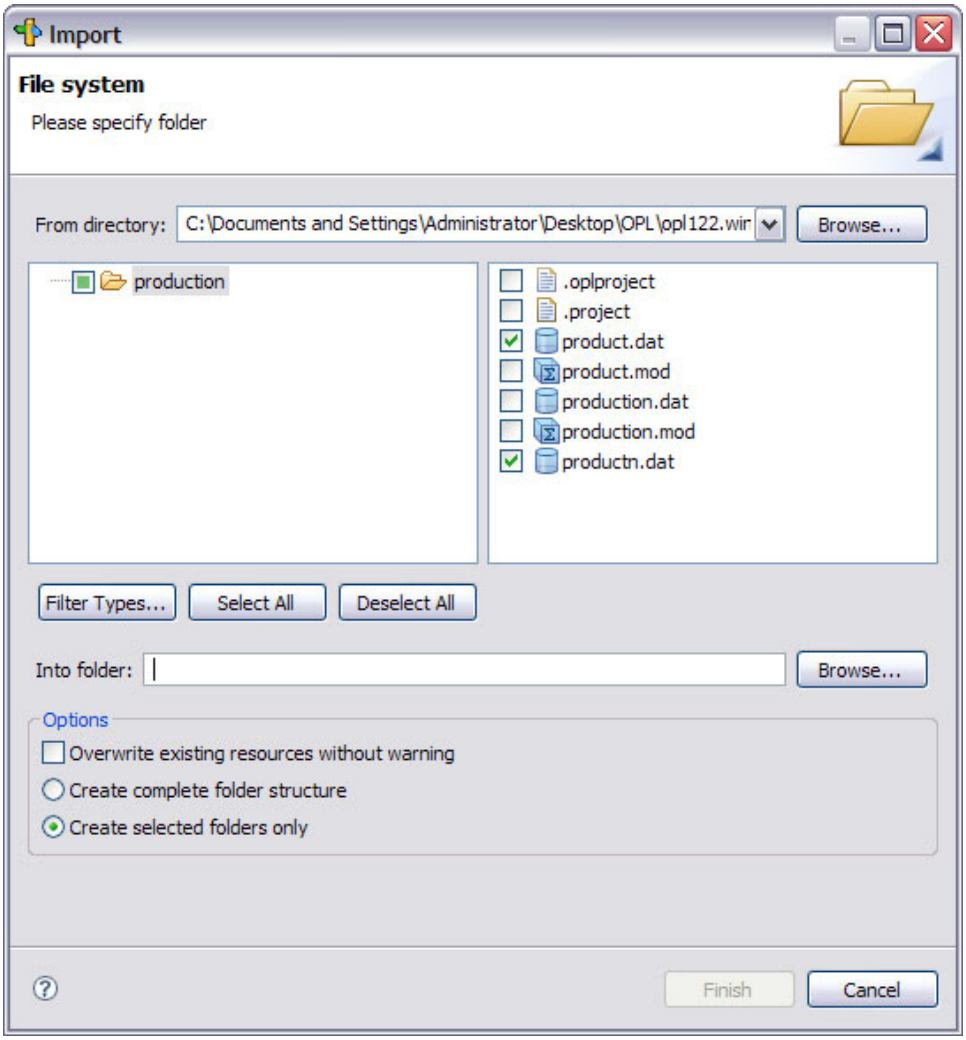
Note:

1. Until you add it to a project, a data file does not appear in the project tree.
2. All files pertaining to the same project must be stored within the same parent project directory.

Procedure

To add a data file to the project:

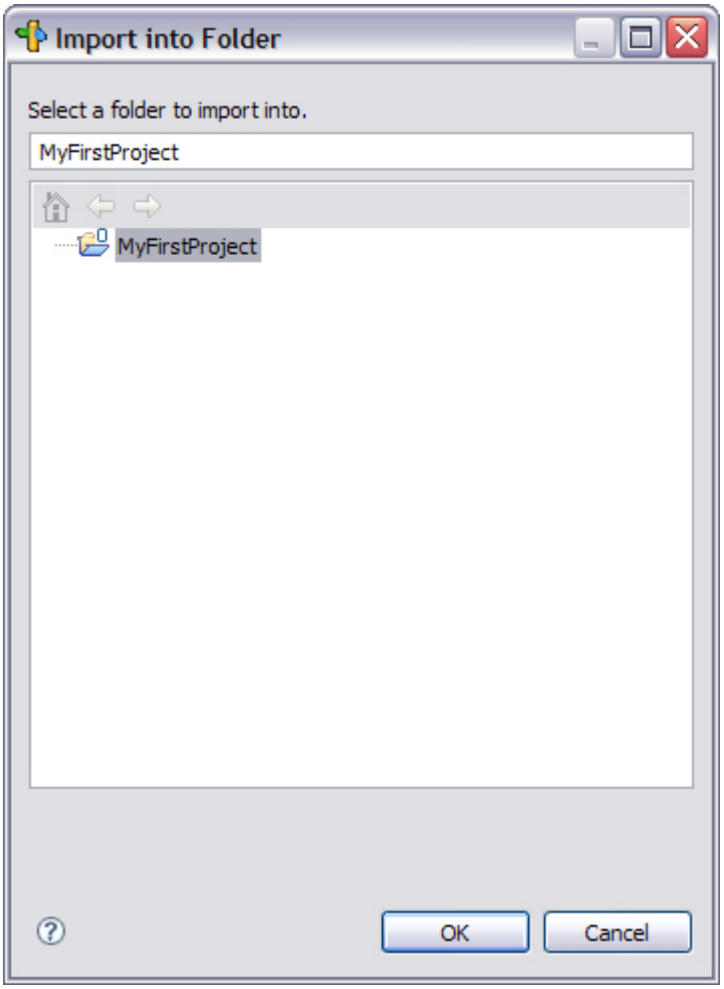
1. Select **File** > **Copy Files to Project** to display the **Import** window.



2. Navigate to the directory:

   `<Install_dir>\opl\examples\opl\production`

   Check the **production** folder box on the left. Clear all the boxes on the right except for `product.dat` and `productn.dat`.
3. Browse to select a folder to import into and click **OK**.



4. Click **Finish** in the **Import** window.

   The data file is added to the project but is not automatically added to the run configuration. You will do this later, as described in "Populating and executing the run configuration" on page 59.

   Note:

   You could also drag the data files file from Windows Explorer into the OPL Projects Navigator, and drop them in the **myFirstProject** project folder.
5. Your OPL Projects Navigator should now look like this:

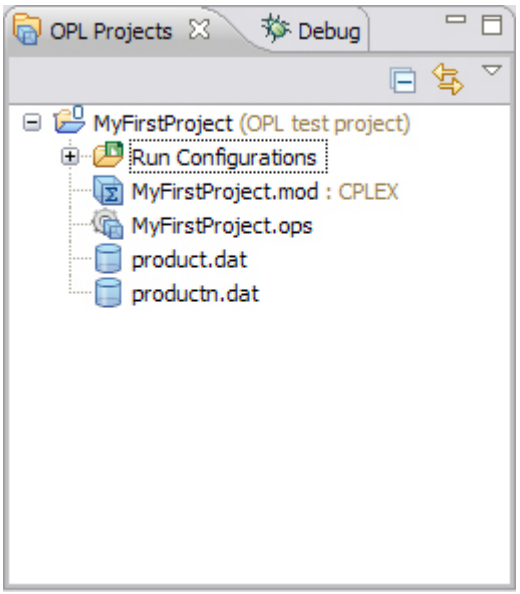


Figure 7. Adding data files to a project

Results

You are now going to execute your project. Later, you will modify the settings file so that you can use it to execute different run configurations of your project (see "Creating and executing a different configuration" on page 65).

Chapter 11. Executing a project

Walks you through populating and executing the run configuration, creating a different configuration, and understanding project execution.

What you are going to do

Describes the purpose of this part of the tutorial.

The CPLEX Studio IDE has a **Run** button  that starts the solving engine to find a solution to the problem expressed in the active model. Solving a model in CPLEX Studio consists of executing the corresponding project, more precisely a run configuration of it; that is, a subset of the model, data, and settings files that make up your project.

Clicking the **Run** button  executes the last run configuration launched.

If you want to execute any run configuration other than the last run configuration you launched, it is probably better to use the **Run** option of the right-click menus in the OPL Projects Navigator to launch the exact run configuration you want. See "The Run options" on page 35 for more details.

Note:

To execute a run configuration in debugging mode with breakpoints, you would use the **Debug** button  instead of the **Run** button. See Using IBM ILOG Script for OPL in *IDE Tutorials*.

To continue with the production planning tutorial, you are now going to:

- Populate the run configuration of your project: see "Populating and executing the run configuration"
- Create more configurations to execute your model with different data and/or settings: see "Creating and executing a different configuration" on page 65
- Learn more about model solving: see also What happens when you execute a run configuration in the *IDE Reference*.

Populating and executing the run configuration

Describes how to add files to a run configuration and execute that configuration.

Before you begin

A run configuration must contain at least a model and can contain only one model.

About this task

When you are finished creating an OPL project, the OPL Projects Navigator should typically look as shown in the figure . You have defined the project as a set of model, data, and settings but the run configuration contains only the `.mod` and `.ops` files. You need to add the data you want to try with your model. Populating

a run configuration consists therefore in adding data and/or settings files to the run configuration subtree.

Procedure

To populate a run configuration:

1. In the OPL Projects Navigator, drag and drop the `product.dat` file to the **Configuration1** run configuration.

   Note:

   If you inadvertently drop the wrong file or drop a file to the wrong place, you can at any time right-click it and choose **Delete**, then confirm. This does not remove the file from the disk.
2. The OPL Projects Navigator now displays the data file name in the run configuration:

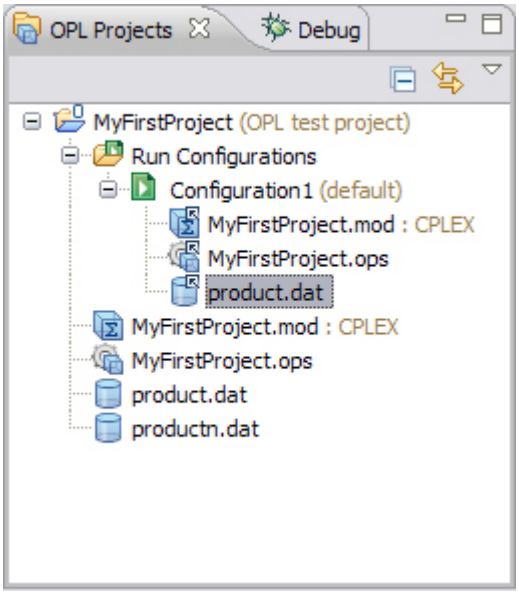


Figure 8. Adding a data file to a run configuration

3. Right-click **Run Configurations** and select **Run** > **Configuration1**. Alternatively, right-click **Configuration1** and select **Run this**.

   Note:

   A project can contain more than one run configuration. To make a configuration the default, right-click its name and choose **Set as Default**. But you can execute any of the other run configurations using the **Run** option.

   You can also execute any run configuration, whether default or not, by clicking the arrow next to the **Run** button  and selecting its name from the **Run** option.

   See “The Run options” on page 35 for more information on running OPL models.

See “The Status Bar” on page 38 for more information on obtaining run status information during the run.

4. Observe the Output Area.

Results

The highlighted tab names show which output panels have received content during execution. The pasta production model uses CPLEX as the solving engine.

- The **Solutions** tab displays one solution.

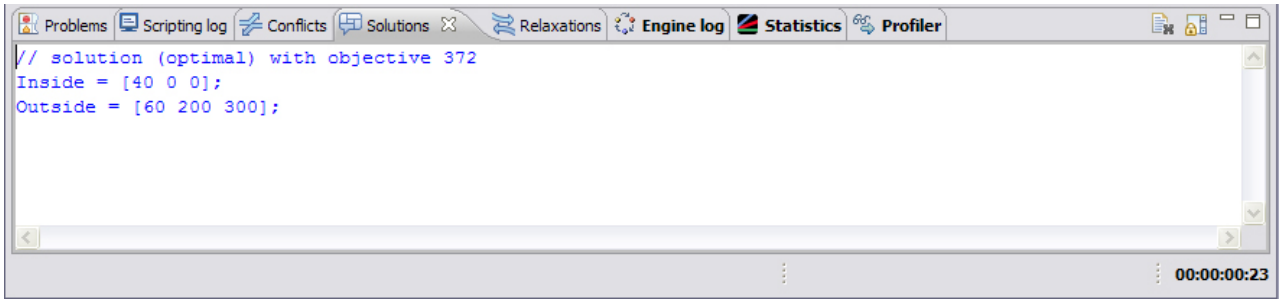


Figure 9. Solution for Configuration1

- The **Engine Log** tab displays details for each iteration.

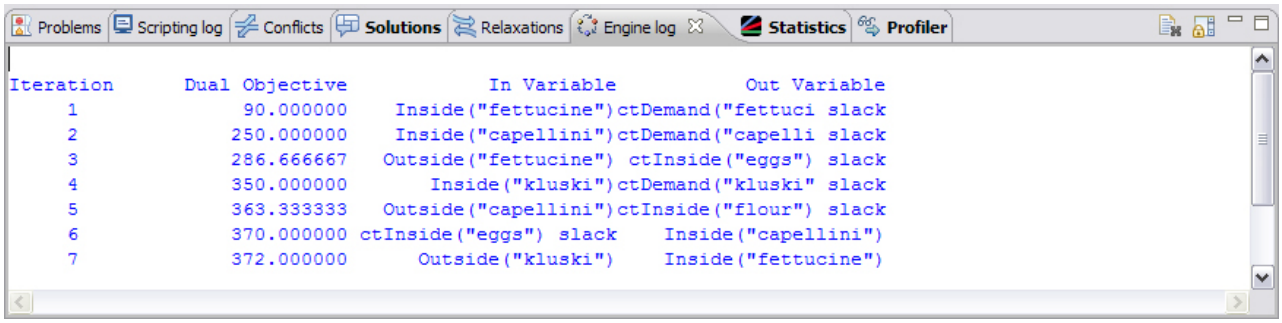


Figure 10. Engine Log for Configuration1

- The **Statistics** tab shows, among other information, the algorithm and the number of iterations.

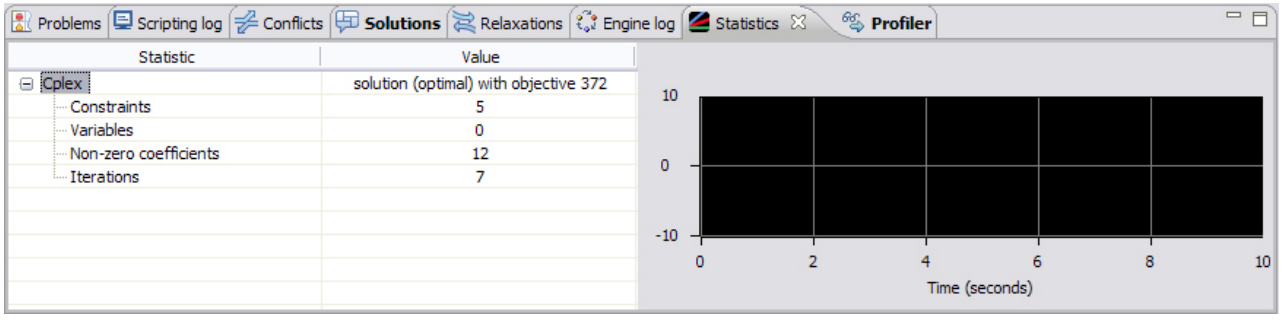
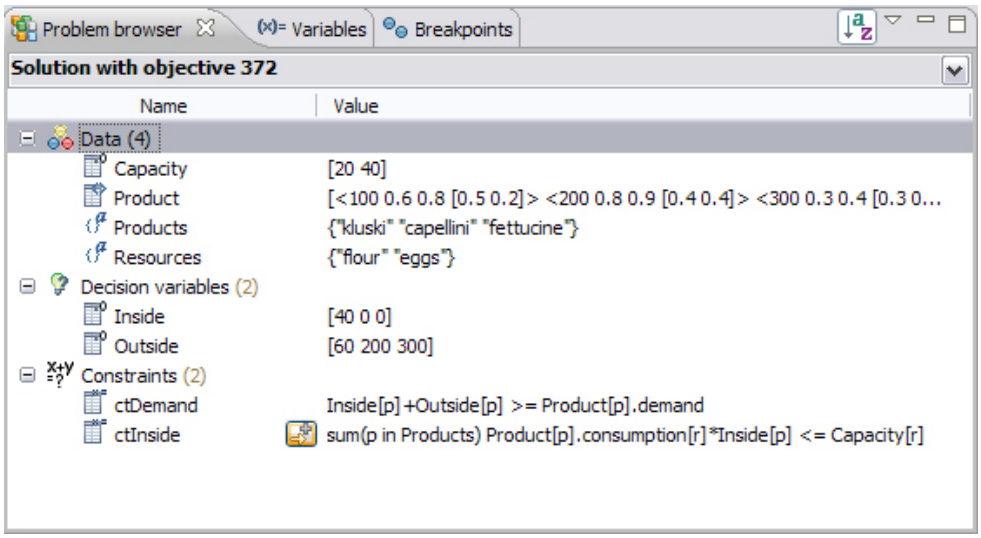


Figure 11. Statistics for Configuration1

- Also, notice that the Problem Browser now contains data for the problem, including the solution, displayed in the drop-down at the top.



You will learn more about the Problem Browser in later sections of this tutorial and in the IDE Tutorials. See also The Output Area in the *IDE Reference*, and Understanding solving statistics and progress (MP models) in *IDE Tutorials*, for more information.

You can now continue with the tutorial and create another run configuration to learn more about project settings, or you may want to go first to What happens when you execute a run configuration in the *IDE Reference* to learn more on the execution process, then proceed to Chapter 12, "Examining a solution to the model," on page 69 to understand results.

Adding a settings file

Explains how to add a settings file to a project so as to be able to change the values of OPL options for language output, mathematical programming, or constraint programming.

About this task

A settings file is where you store user-defined values of OPL options for language output, mathematical programming, or constraint programming. It gives you access to the solver parameters and allows you to modify them. For more information, see Setting programming options in the *IDE Reference*.

Note:

If your model contains a `main` flow control script, the OPL values you set in the `.ops` file, as well as the settings set within the `main` script, apply to the current model only, not to the submodels loaded and solved at execution time.

Because you left the **Create settings** option checked in Step 2 of "Creating an empty project" on page 49, a default settings file already exists for your project and this is the one you used in "Populating and executing the run configuration" on page 59.

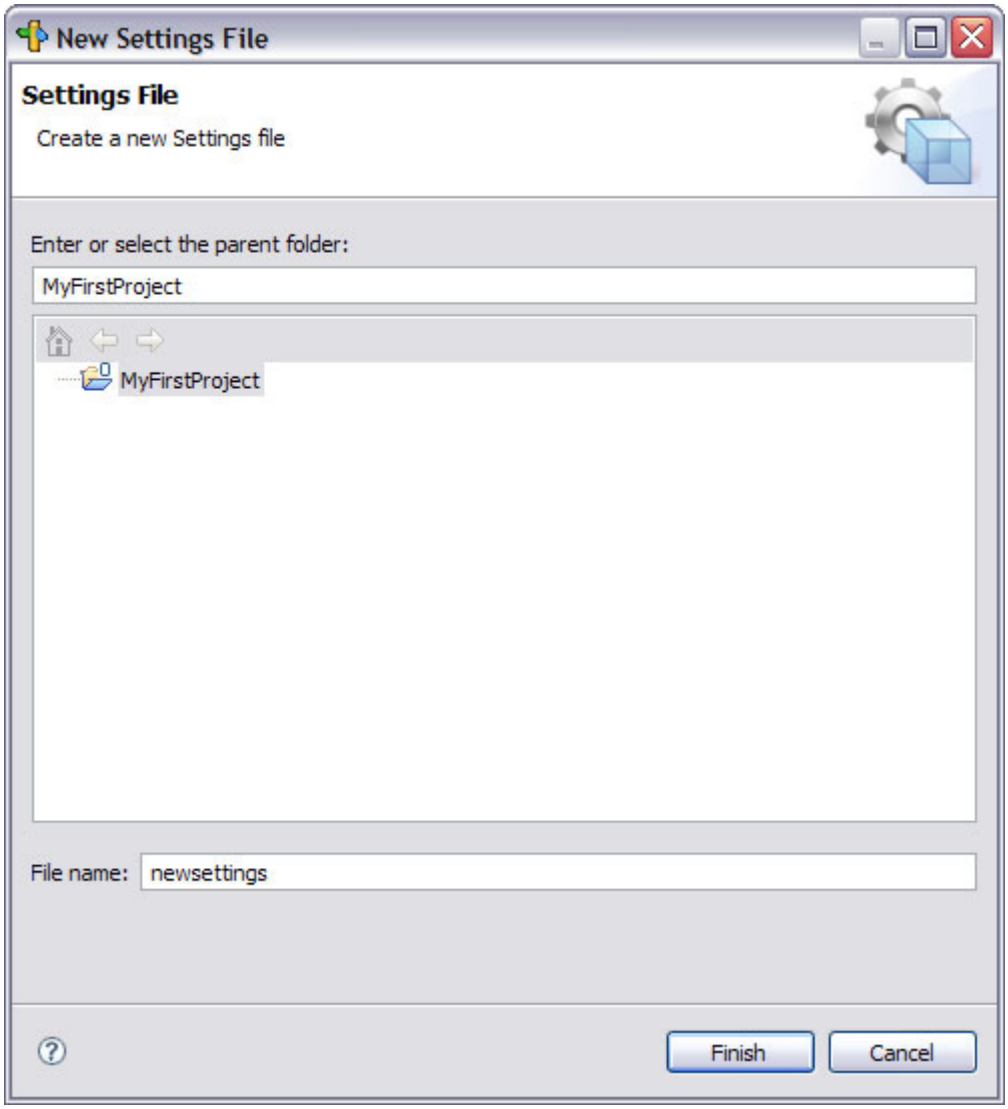
To practice with a different run configuration without losing your default settings, you are now going to add a second settings file to your project and use it to set a different value to one mathematical programming option.

This stage of the tutorial assumes you have at least a model in your project and want to be able to modify OPL, CPLEX, or CP Optimizer parameters.

Procedure

To add a settings file to an existing project:

1. Select the project name in the OPL Projects Navigator, then right-click and select **New** > **Settings**.
2. In the dialog box, select the parent project and provide a name for the new settings file. Then click **Finish**. The `.ops` extension will be added automatically.



Results

Notice the changes in IDE window:

- The `newsettings.ops` file is added to the project in the OPL Projects Navigator.

- The settings editor appears.
- The Outline window displays the settings outline. You can later access the modified settings directly from this window.

Each option available in the settings file is documented individually in OPL language options, Constraint programming options, and Mathematical programming options, in the *IDE Reference*.

You are now ready to use the new settings file to set a mathematical programming option with which you will then execute the model.

Changing an MP option value

Gives an example of how to use the IDE settings editor to change an option value.

About this task

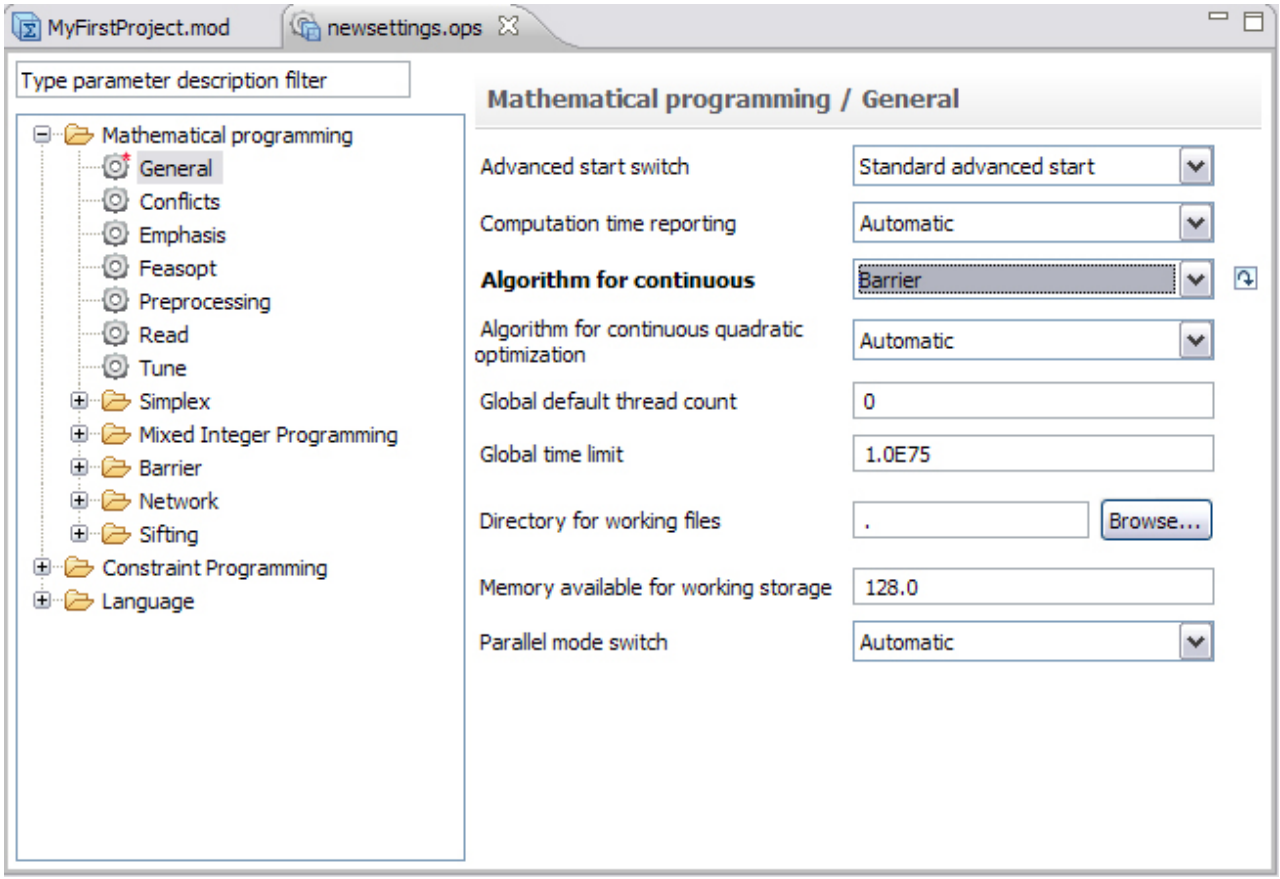
You are now going to modify the default value of one of the MP options.

Procedure

To change an option value in the IDE:

1. Double-click the new `.ops` file in the OPL Projects Navigator, if it is not already open.

   The panel for mathematical programming options is displayed in the Editing Area.
2. In the **Mathematical Programming /General** category, choose **Algorithm for continuous problems** and select **Barrier** from the dropdown list.



3. Choose **File** > **Save**.

Results

Your project now includes a model file, two data files, and two settings files. In the next step, "Creating and executing a different configuration," you will create a second run configuration to execute the model with different data and different settings.

Creating and executing a different configuration

Describes how to create a second run configuration, then populate and execute it.

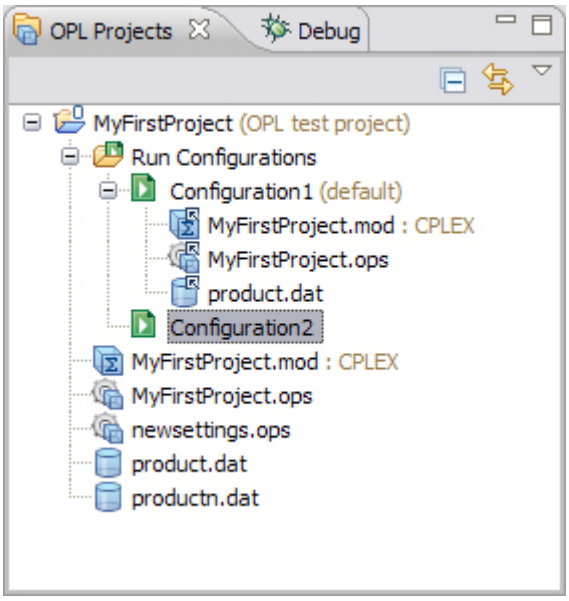
About this task

You will now create a second run configuration, then populate and execute it.

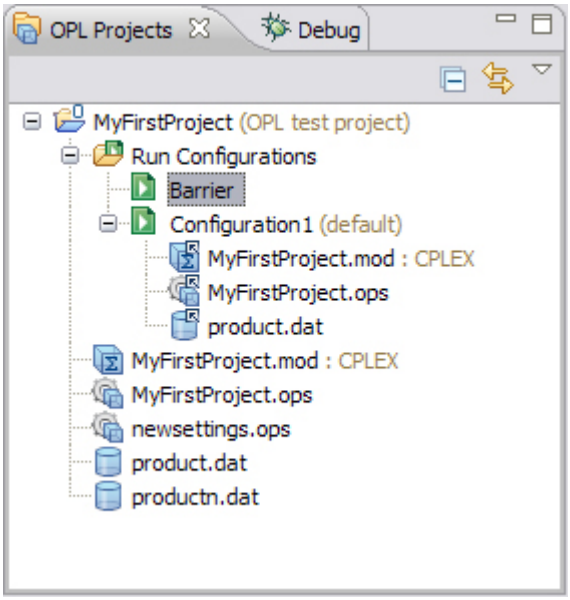
Procedure

To create and execute a second run configuration:

1. In the selected project, right-click the **Run Configurations** folder and choose **New** > **Run Configuration** from the menu. A new run configuration with the default name **Configuration2** is added.



2. Optionally, rename the run configuration, by selecting it, right-clicking and choosing **Rename** from the menu. In this example, **Configuration2** is renamed **Barrier**.



3. Drag and drop the files `myFirstProject.mod`, `product.dat`, and your new settings file`newsettings.ops` into the new **Barrier** run configuration.

   The OPL Projects Navigator should look like this.

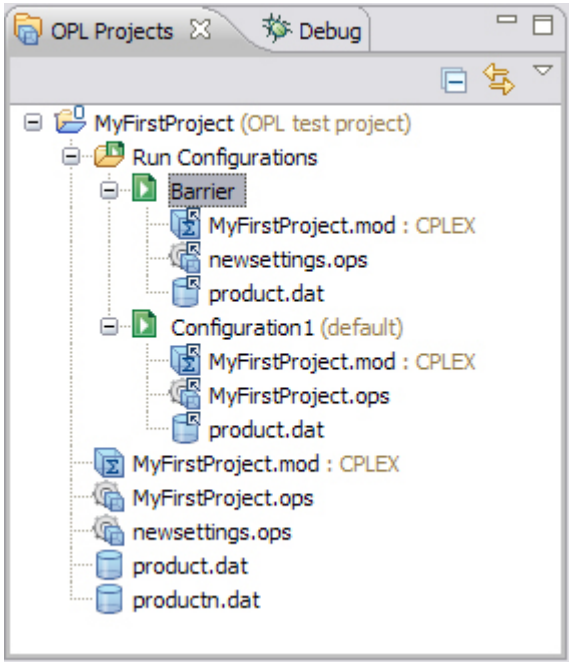


Figure 12. A run configuration with different settings

4. Right-click in the project and select **Run** > **Barrier** from the menu to execute the model with changed settings.

   The CPLEX engine now uses **Barrier** as the value of the algorithm for continuous problems.

Results

Observe the differences in the output tabs.

- The **Solutions** tab displays the same solution (see Figure 9 on page 61).
- The **Engine Log** tab shows a different report.

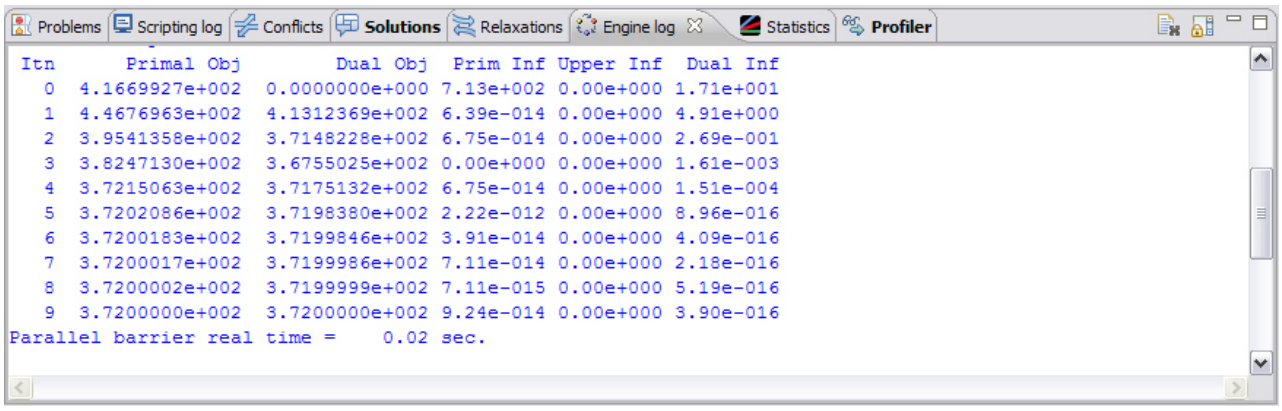


Figure 13. Engine Log for Barrier configuration

- The **Statistics** tab shows a different algorithm (Barrier) and number of iterations.

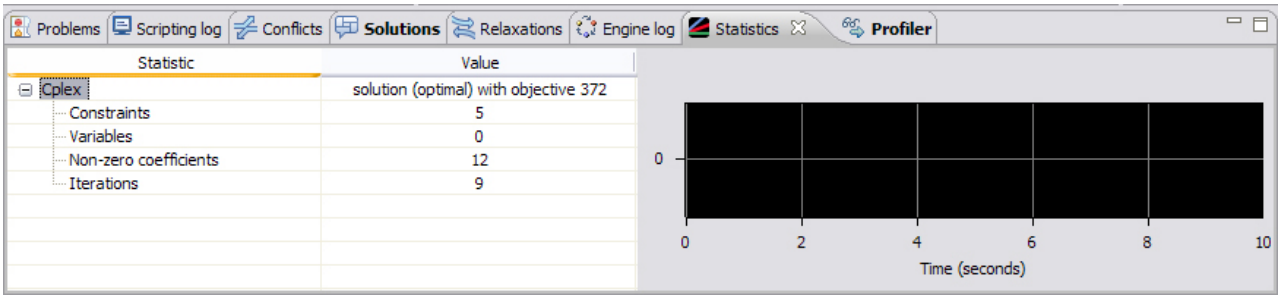


Figure 14. Statistics for Barrier configuration

You can proceed to Chapter 12, "Examining a solution to the model," on page 69 to learn more about the execution process and its results.

See also:

- What happens when you execute a run configuration, in the *IDE Reference*, for details on the execution process.
- Examining the statistics and progress chart (MP) in *IDE Tutorials*.

Chapter 12. Examining a solution to the model

Explains how to read the solutions in the output tabs and read details of the executed model in the Problem Browser.

Execution results

Explains how to examine the results in the IDE after executing a run configuration.

When you execute a run configuration to find the solutions to the problem expressed by the model, the IDE provides facilities for examining results — one for examining the solution and one for examining the details of your model.

- **To examine the solution**, browse the output tabs in the lower half of the main window, as explained in the next section "The Output tabs."

  You can customize the solution display: see OPL language options in the *IDE Reference*.
- **To examine the structure** of the model as solved by the engine, use the Problem Browser: see "Understanding the Problem Browser" on page 75 and Doing more with the Problem Browser in the *IDE Reference*.
- **Tuning parameters:** When working on a MIP project, possibly by trying various run configurations, you may want to test the performance of your model before deployment. The **Tune model** button in the IDE standard toolbar offers a convenient way to do so. See Using the performance tuning tool in *IDE Tutorials*.

Note:

For demonstration purposes, the illustrations used in this section are taken from various code samples from the product distribution. However, you can continue with the project you have just created.

The Output tabs

Describes how the IDE output tabs reflect the result of project execution.

After you execute a run configuration, the solving engine searches for the optimal solution, and when execution is complete, the IDE displays several output tabs in the main window.

These tabs are:

Problems

The **Problems** tab displays semantic and syntax errors as you type when you write a model manually, and internal errors, such as scripting or algorithm errors, when you solve a model.

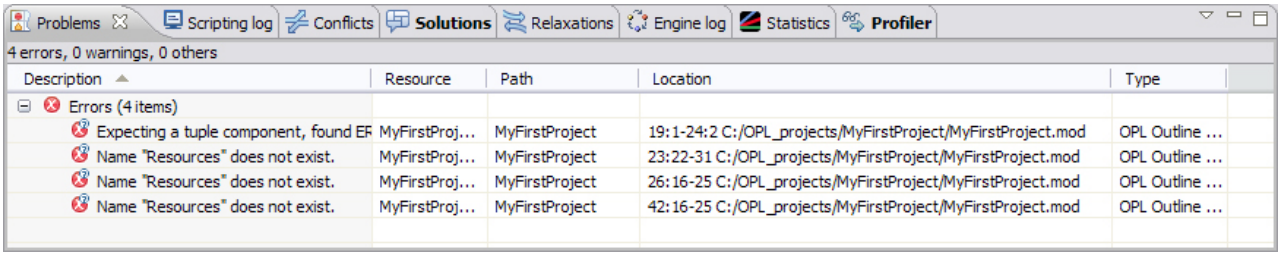


Figure 15. Problems tab

How to read messages:

- The **Description** column guides the user as to the nature of the error.
- The **Resource** column indicates which resource the error occurred in.
- The **Path** column provides the path to the current file.
- The **Location** column reflects the line that's affected by the error.

Scripting Log

The **Scripting log** tab shows execution output related to the IBM ILOG Script `main` or `execute` or `prepare` blocks of the model (if applicable).

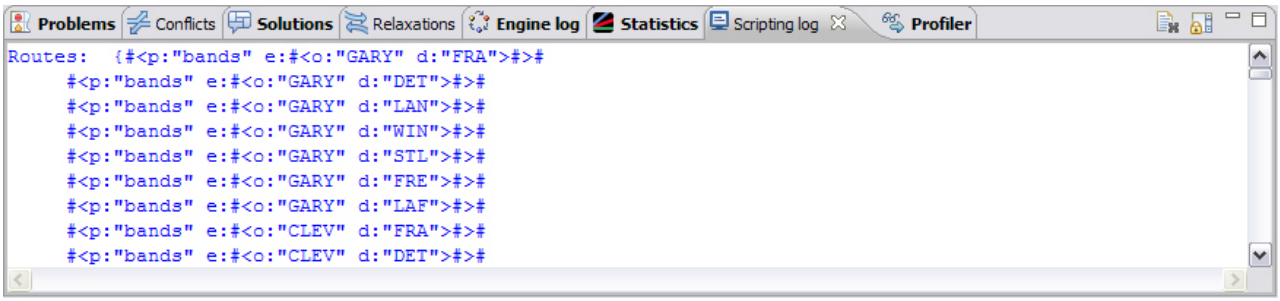


Figure 16. Scripting log tab (`transp4.mod`)

Solutions

The **Solutions** tab displays the final solution to a model and, if applicable, any intermediate feasible solutions found.

The variables are displayed as well by default, but you can disable this display, see OPL language options in the *IDE Reference*.

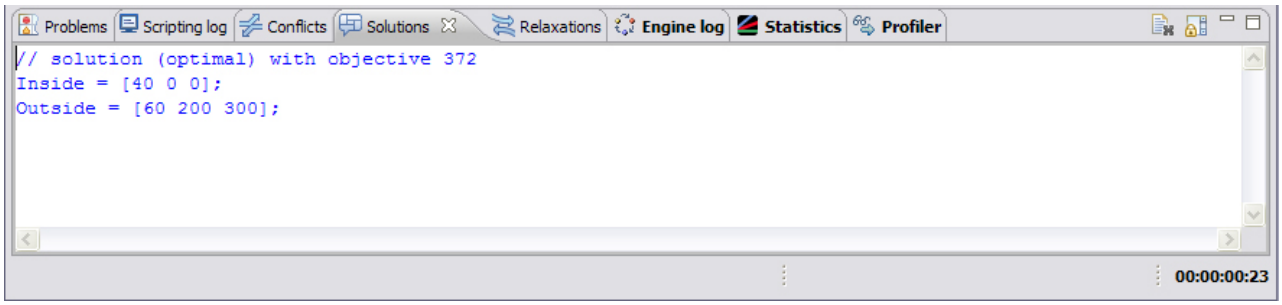


Figure 17. Solutions tab (basic configuration of `product.mod`)

CPLEX solution quality

CPLEX solution quality information is available on the **Solutions** tab. You will see an additional report such as:

```
// Quality Incumbent solution:
// MILP objective                                3.8300000000e+002
// MILP solution norm |x| (Total, Max)           1.40000e+001 1.00000e+000
// MILP solution error (Ax=b) (Total, Max)       0.00000e+000 0.00000e+000
// MILP x bound error (Total, Max)               0.00000e+000 0.00000e+000
// MILP x integrality error (Total, Max)         0.00000e+000 0.00000e+000
// MILP slack bound error (Total, Max)           0.00000e+000 0.00000e+000
//
// Branch-and-cut subproblem optimization:
// Max condition number:             4.2500e+001
// Percentage of stable bases:       100.0%
// Percentage of suspicious bases:   0.0%
// Percentage of unstable bases:     0.0%
// Percentage of ill-posed bases:    0.0%
```

The details of the MIP problem report can be controlled by using the OPL parameter `mipkappastats`. See MIP kappa computation in the *Parameters Reference Manual*.

Example of preprocessing scripting:

```
execute {cplex.mipkappastats = 2;}
```

Conflicts

When a CPLEX model proves infeasible, the **Conflicts** tab shows the places where you can change the data or the way filtering constraints are expressed in order to remove the incompatibilities that made the model infeasible. See Relaxing infeasible models in *IDE Tutorials* for details.

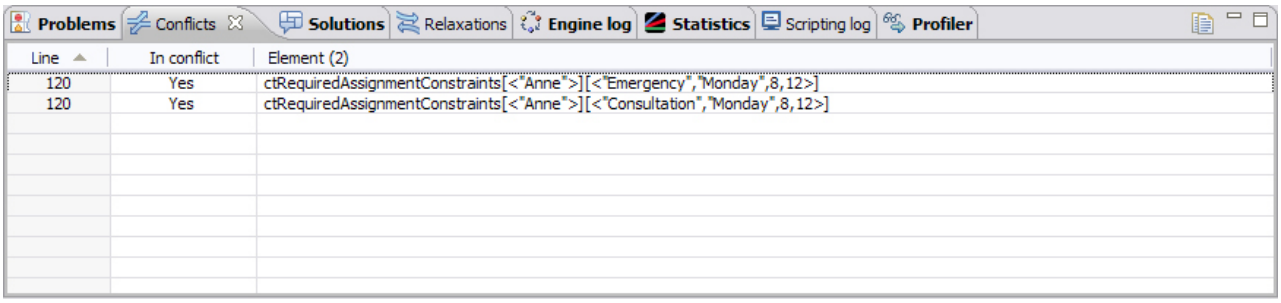


Figure 18. Conflicts tab (`nurses` project)

The **Conflicts** tab is empty after execution of `product.mod` because that project is not designed as infeasible.

Relaxations

When a CPLEX model proves infeasible, the **Relaxations** tab shows the places that constraints can be relaxed to remove the incompatibilities that made the model infeasible. See Relaxing infeasible models in *IDE Tutorials* for details.

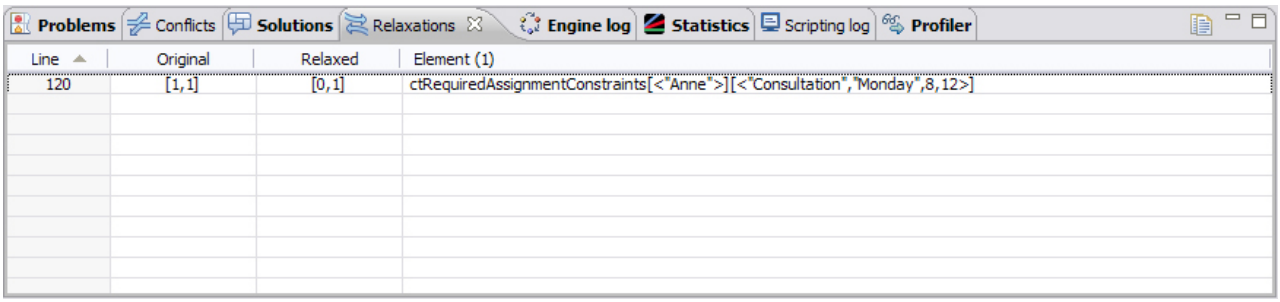


Figure 19. Relaxations tab (`nurses` project)

The **Relaxations** tab is empty after execution of `product.mod` because that project is not designed as infeasible.

Engine Log

The **Engine Log** tab displays information from the solving engine (CPLEX for `product.mod`) on the solving process and on the objective function (in this example, a `minimize` statement).

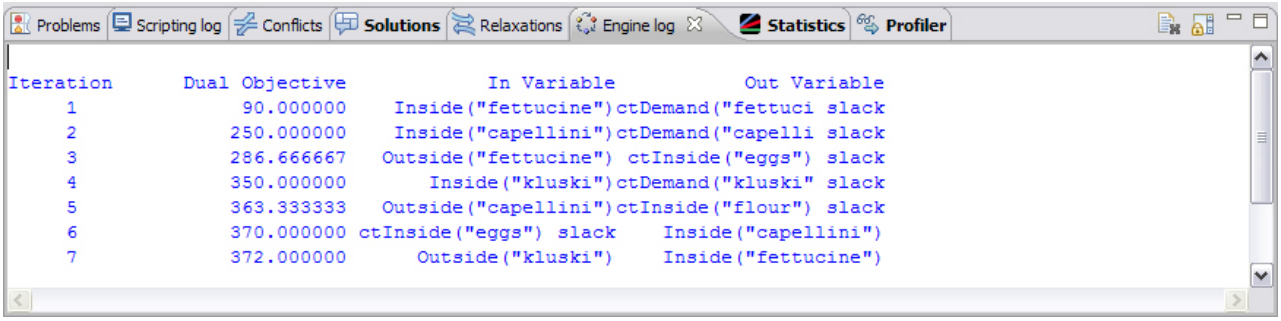


Figure 20. Engine Log for an MP model - CPLEX Dual Simplex (`product.mod`)

CPLEX users may recognize this information as what they see when executing CPLEX Interactive.

For comparison, the **Engine Log** for a constraint programming model looks like this:

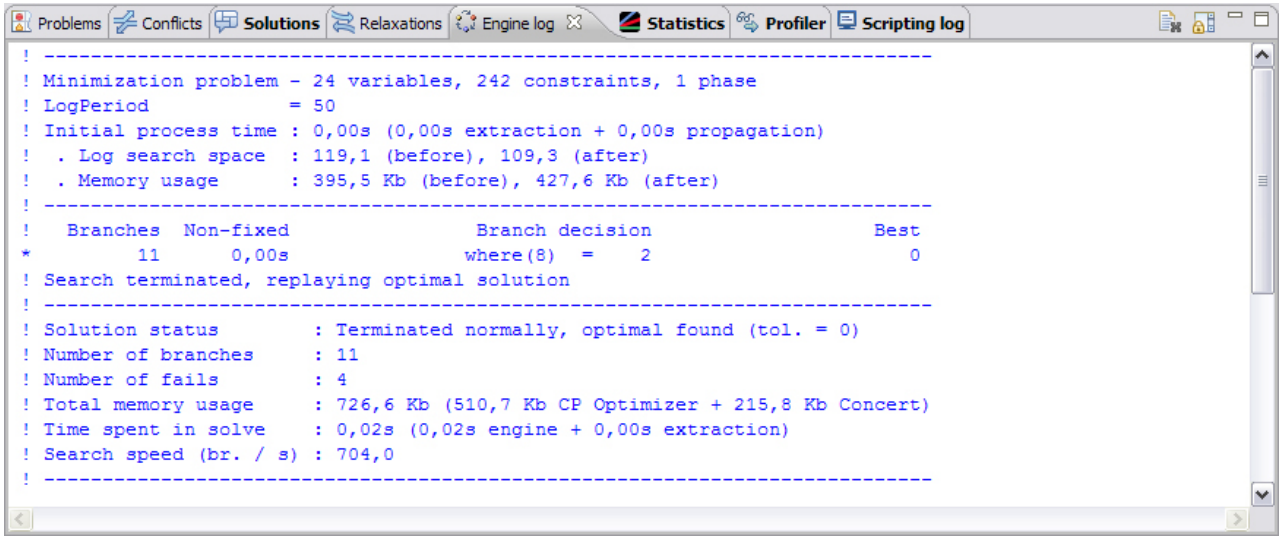


Figure 21. Engine Log for a CP model (`steelmill` project)

Statistics

The **Statistics** tab shows details of the algorithm used by the solving engine.

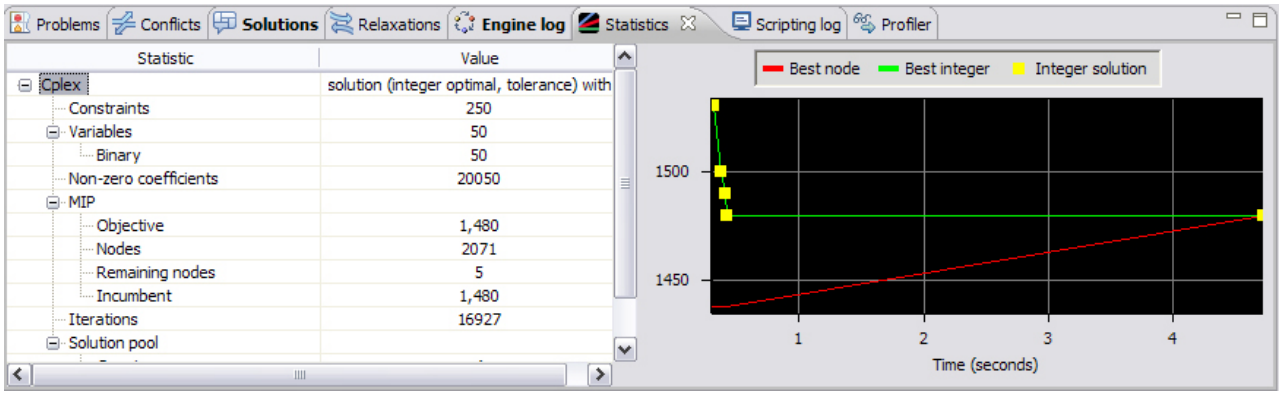


Figure 22. Statistics for an MP model (scalable configuration of `warehouse` project)

Profiler

The profiling tool computes the time and memory used by each execution step listed in the **Description** tree on the right and displays it as a table in the **Profiler** tab of the Output Area. You can use this information to improve the model so that it executes faster and consumes less memory. The Profiler table also displays details of model extraction and engine search during the solving phase. See Profiling the execution of a model in *IDE Tutorials*.

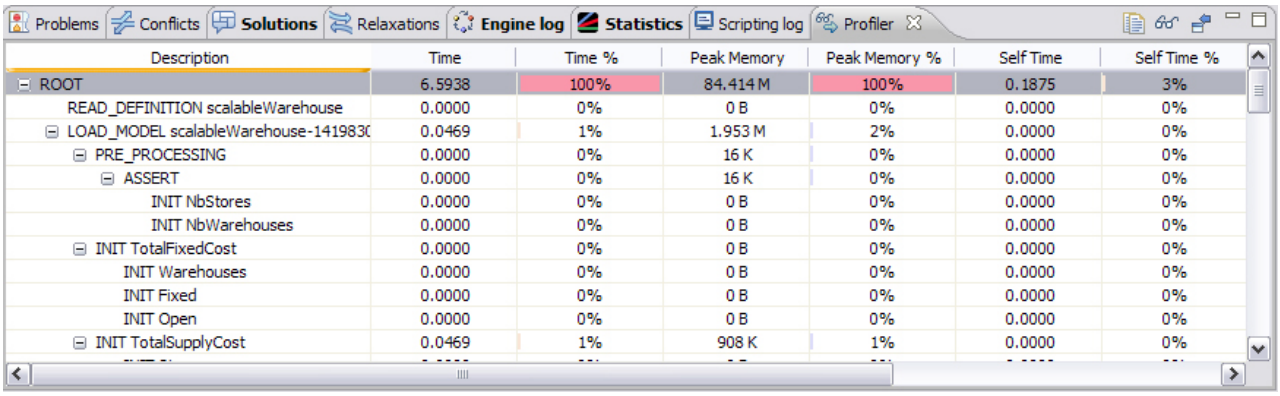


Figure 23. Profiler table for an MP model (scalable configuration of `warehouse` project)

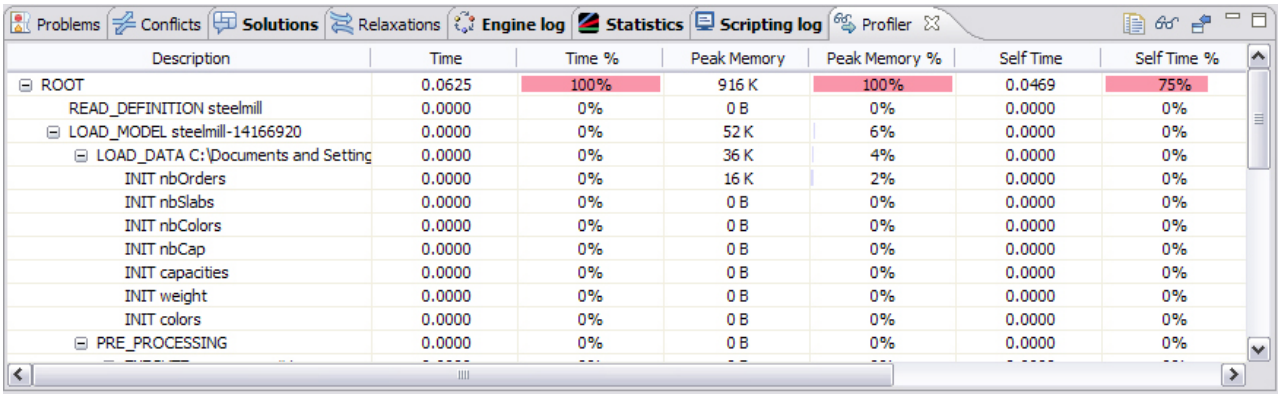
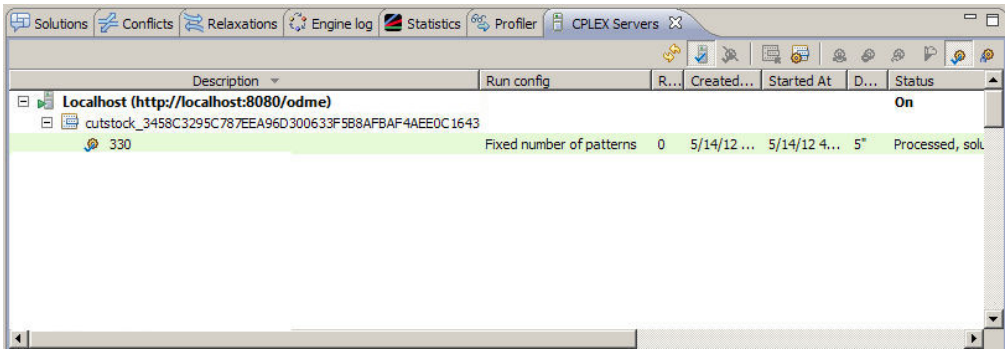


Figure 24. Profiler table for a CP model (`steelmill.mod`)

CPLEX Servers

This tab can be displayed using the command **Window > Show View > CPLEX Servers**. The names and locations of servers you added via the command **Window > Preferences > OPL > CPLEX Servers** will be displayed,



For more information, see IDE connection to CPLEX servers.

Understanding the Problem Browser

Describes the information displayed in the Problem Browser before and after execution.

The Problem Browser shows a structured view of the problem expressed by the model. See The Problem browser in the *IDE Reference* and Doing more with the Problem Browser in the *IDE Reference* for a complete presentation.

When you first open a project, the Problem Browser is empty. You can use it to browse the model before any execution and to examine the values after execution.

When you execute a run configuration, the Problem Browser provides a way for you to examine the solution to your model, in addition to what you see in "The Output tabs" on page 69. It summarizes information about the data structures defined in the model to express the optimization problem.

As an example, you can open the `production` project and run the **Named data** configuration. After the execution, the Problem Browser displays values for the model elements.

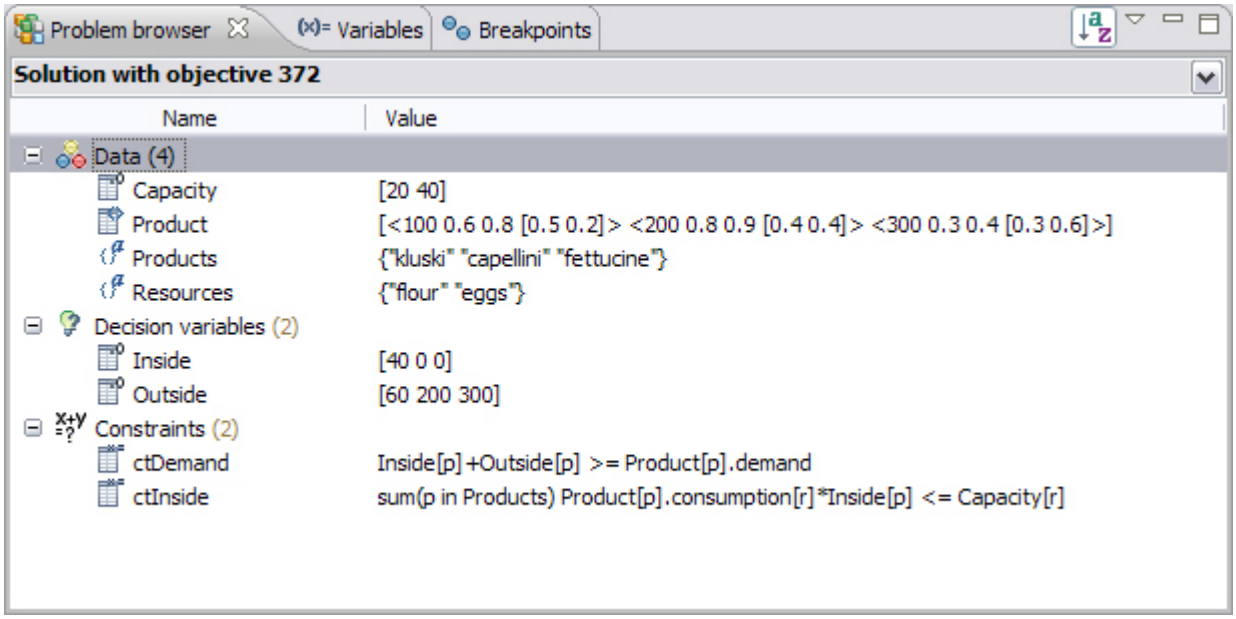
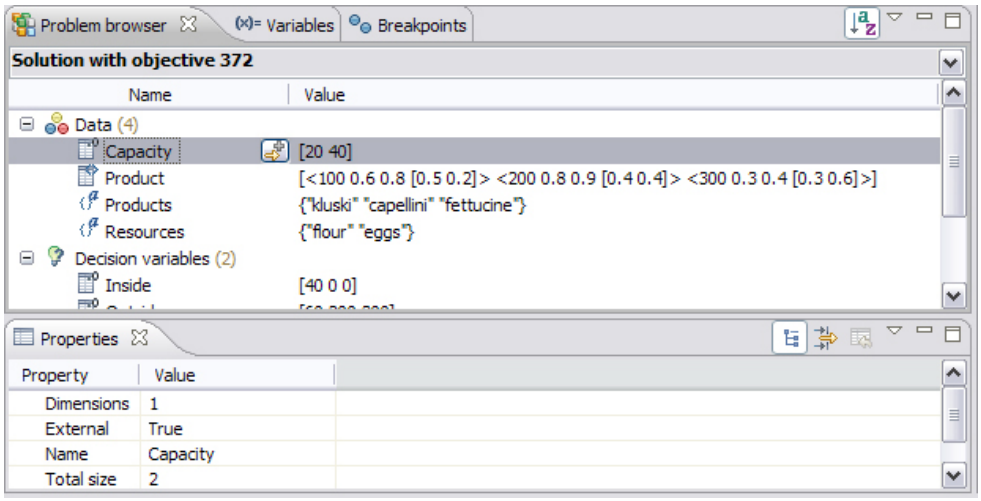


Figure 25. Problem Browser after execution (`product.mod`)

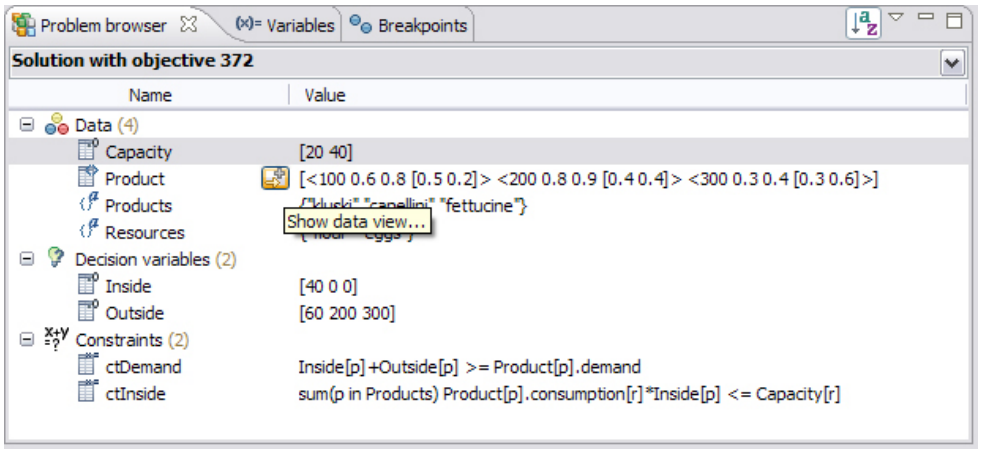
Observe the Problem Browser window.

- The drop-down list at the top displays the final solution, which is the only solution it contains after the execution of the **Named data** run configuration. For run configurations that generate more than one solution, the list displays the solution pools that were computed by the engine. Selecting one of these solutions from the drop-down list displays data for that solution in the lower part of the Problem Browser. See Working with the solution pool in *IDE Tutorials*.
- The categories in the **Name** column (Data, Decision Variables, Constraints) are populated with model objects and expanded to show the corresponding values. The order within each category is alphabetical.

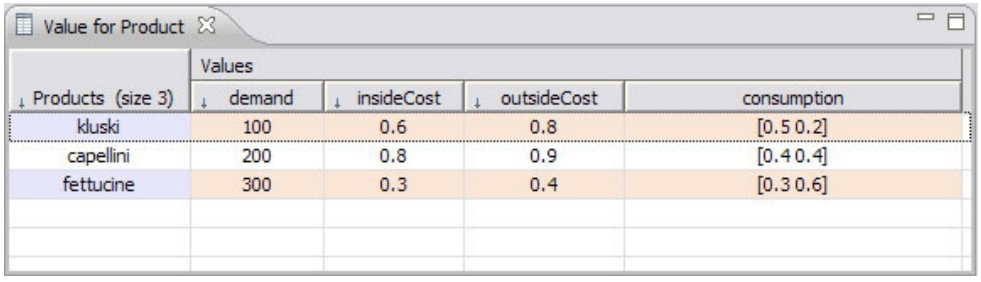
- The **Value** column shows values for the model objects. The Property name and Property value columns remain empty until you select a model element.
- If you select an item in the main Problem Browser window (in this case, **Capacity**), properties for that object are shown in the Properties window.



- If you slide the cursor over a data object (in this case, **Products**), a Show data view button appears (shown below with its tooltip visible):



Clicking this button displays the data view in the main editing area:



For a large tuple set, the values may not all be visible within the window. In this case, an ellipsis appears at the end of the cell. Pass the cursor over the column to display all the values in a tooltip.

Note:

If the model has only unlabeled constraints, the **Constraints** line is empty. To observe this, comment out the constraint labels and execute the project again. See Constraint labels in the *Language Reference Manual*.

Viewing the results of scheduling problems

The results of solving a scheduling model can be displayed in a Gantt chart.

When a problem has been solved, the Problem browser view shows both data and decision variable values. If the problem includes a scheduling variable, the value of the variable is displayed. For an example of a scheduling problem solved with CP Optimizer, see Viewing the results of sequencing problems in a Gantt chart.

To display a solution to a scheduling problem that is not solved with CP Optimizer, the tuple schema in the OPL model must contain:

- an `int` column with the name `present` or `_present`
- an `int` column with the name `start` or `_start`
- an `int` column with the name `end` or `_end`
- an `int` column with the name `size` or `_size`

and optionally

- a `string` column with the name `_label`

Columns can be in any order.

If these conventions are met, arrays of these tuples (as intervals) can be displayed in a Gantt chart.

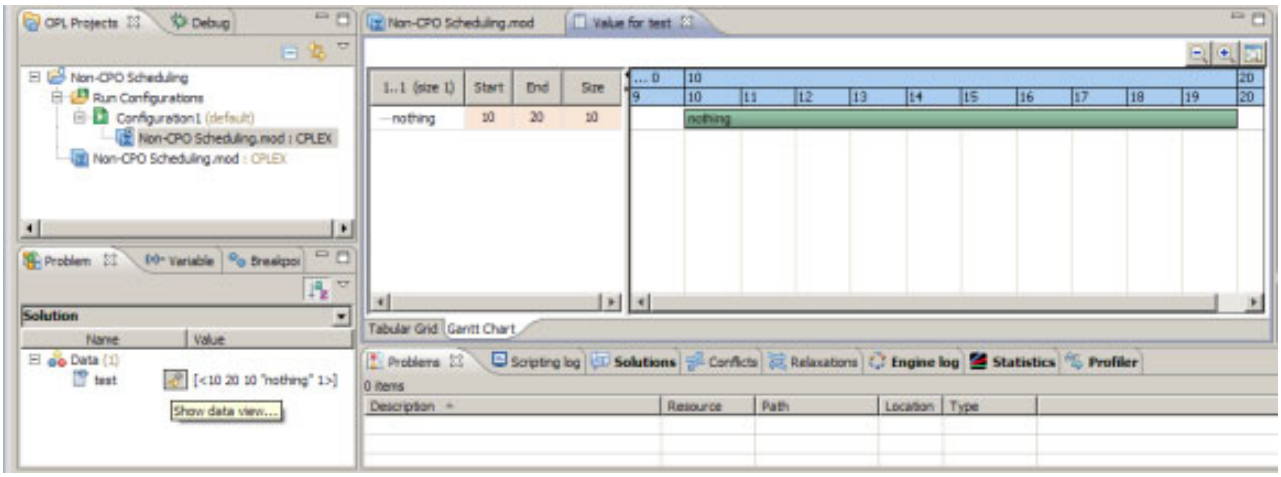
For example, the model:

```
tuple noncposched {
   int start;
   int end;
   int _size;
   string _label;
   int present;
}

noncposched test[1..1] = [<10, 20, 10, "nothing", 1> ];

execute {
   test;
}
```

will display the solution in the following way:



The **Show data view** button will appear in the Problem browser when the mouse cursor is placed over the data name `test`. Click the button to display the Tabular Grid, then click the tab Gantt Chart.

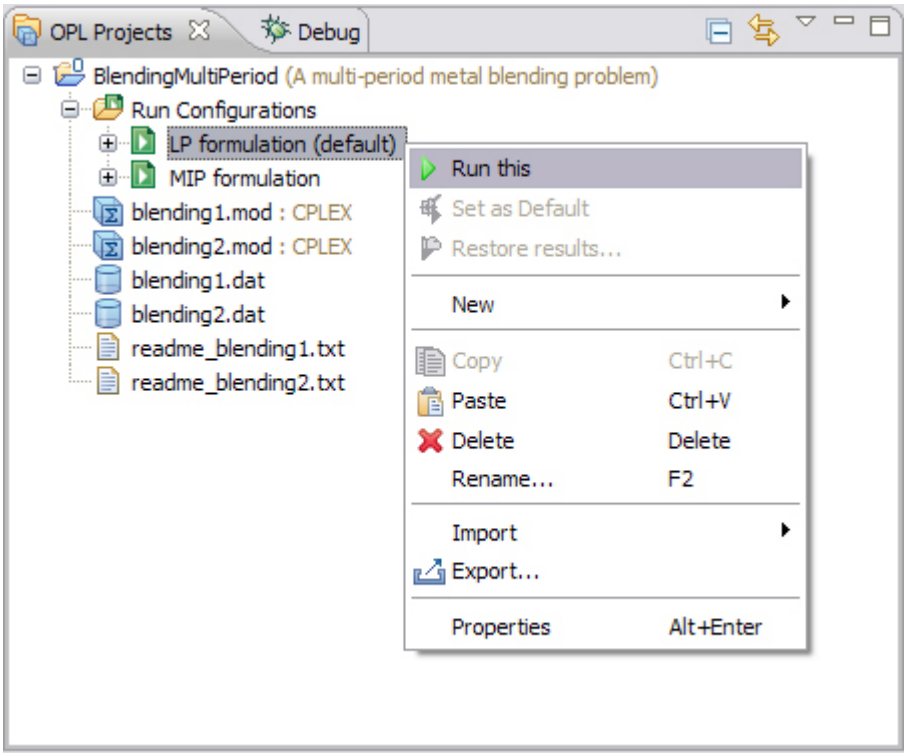
Chapter 13. Saving and restoring results

You can save the results of a solve, keep a history of the saves and restore a saved version in the IDE.

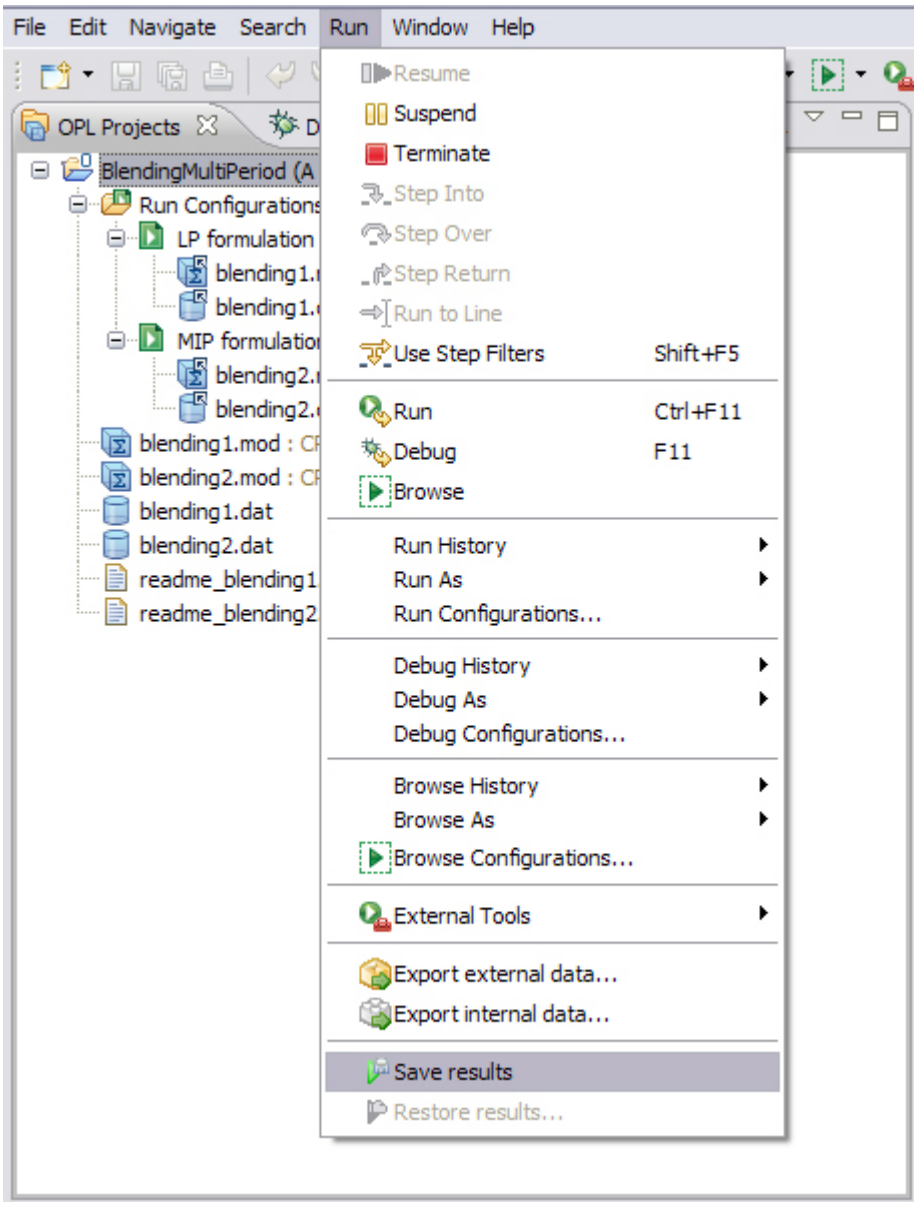
After executing a model, you can save the results, keep a history of the saves and restore a saved version in the IDE. Here we use the `BlendingMultiPeriod` example to illustrate this feature.

Note: In the case of a solution pool, only the best solution is saved.

1. Run the default configuration, either from the **Run** command in the menu bar, or from the contextual menu, as shown.

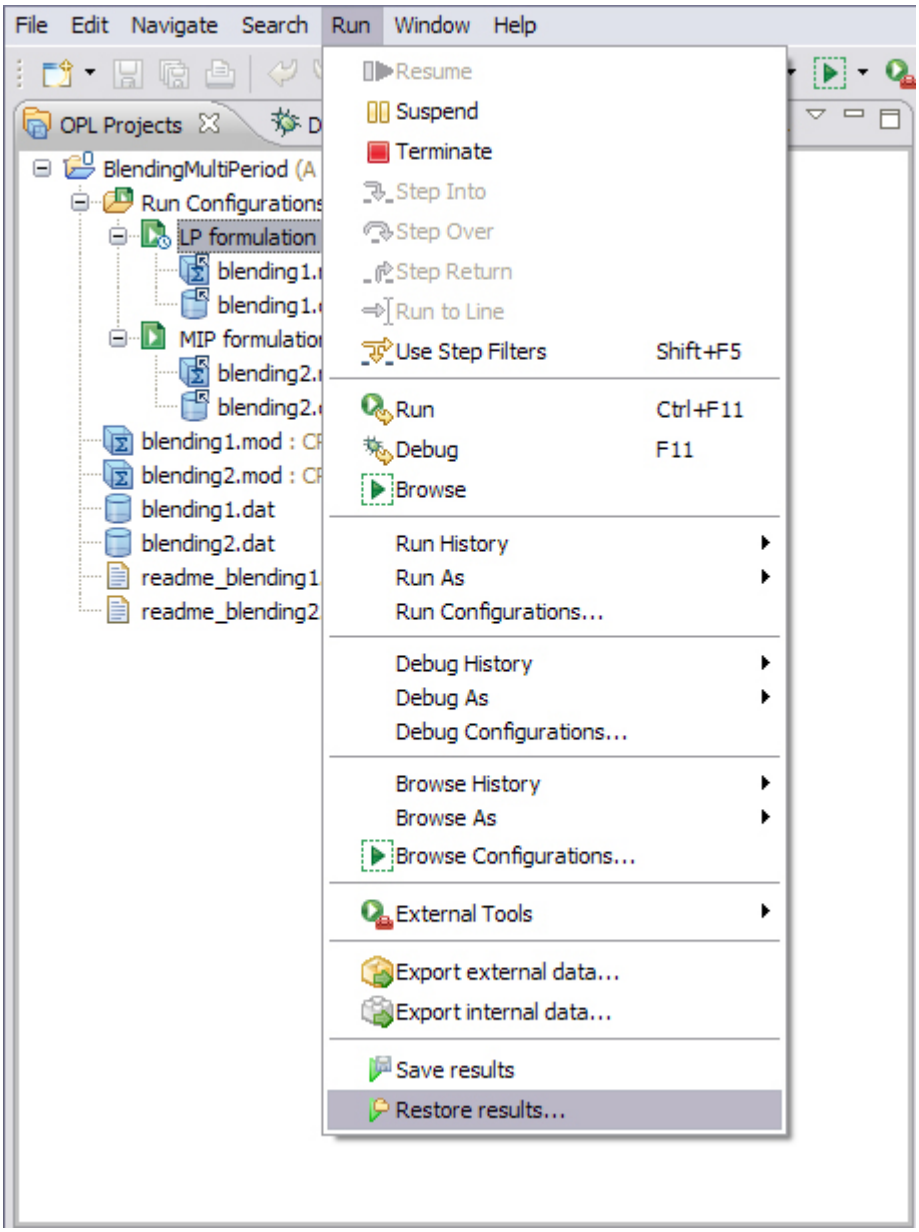


2. From the menu bar, select **Run** > **Save results**.



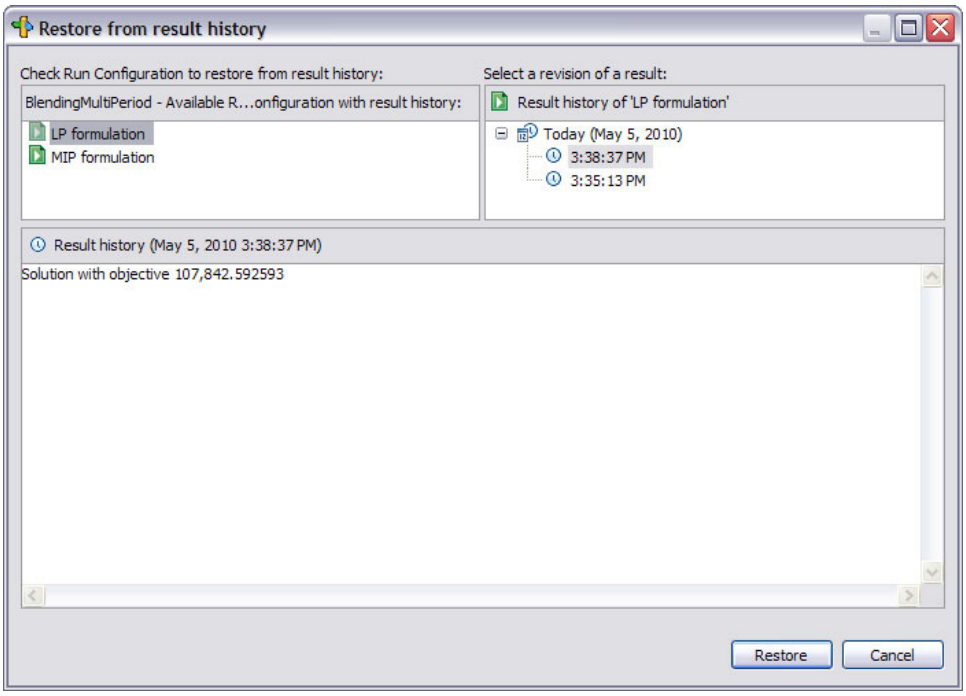
This action saves the results as displayed in the problem browser (data, decision variable values, expressions, constraints), as well as the `.mod` file and the log files (scripting log, solutions log, conflicts, relaxations, engine log, statistics and profiler).

3. After you have saved a result, the **Restore results** option becomes active, either in the **Run** menu or in the contextual menu of the Run Configuration.



The icon next to the **LP formulation** run configuration has a tiny clock attached to it, to indicate that it has saved results. Click **Restore results**.

4. The history of saved results is displayed, and a summary of the selected result is shown.



Click the **Restore** button to display the results in the Problem Browser. You can then click on an object in the Problem Browser to display a read-only version of the corresponding `.mod` file in the editor. All the logs mentioned above are also restored.

Part 3. Appendixes

Index

T

V

W

Printed in USA