AIDA

Automotive and Industrial Diagnostic Assistance

Tutorial AIDA - Getting Started

Revision level: see Revision index

SIEMENS VDO
Automotive
VDO-Straße 1
D-64832 Babenhausen
Germany

bdk
Büro für Datentechnik GmbH
D-35418 Buseck
Germany

1 Contents

2 Revision index

Date	Author	Rev.	Ref.	Type	Description
2003-11-13	Carsten Hielscher	1.01.00	-	-	Insignificant extensions inserted
2003-11-05	Carsten Hielscher	1.00.00	-	-	First Release
2003-10-21	Carsten Hielscher	0.91.00	-	-	Revision
2003-10-10	Fouad Korkmaz	0.90.00	-	-	Initial Revision

Acronyms:

AIDA

Automotive and **I**ndustrial **D**iagnostic **A**ssistance. System that is used to implement computer supported diagnosis for control modules and bus systems.

BSK

The manufacturer of the AIDA-System.

POOL

Portable **O**bject **O**riented **L**anguage. Object oriented programming language by BSK. POOL is the programming language of the AIDA system.

IDE

Integrated **D**evelopment **E**nvironment.

SimProD

Simulation, **Pro**totypes and **D**iagnostics.

3 Introduction

This document should give you an easy startup with the AIDA system. You will use 'AIDA Studio' to create your first POOL program. We show you how to create a project, a source code file and how to compile the sources. Finally we explain the 'Commander' which is used to run your first POOL program. This document assumes that you have installed AIDA successfully. In case you have not installed AIDA yet, you can install it via the SimProD-Homepage. To install AIDA just execute the setup and follow the instructions of the installation wizard.

For further information on POOL see

- the BSK-Examples located in the AIDA installation directory
  (usually in the folder C:\Program Files\BSK\AIDA\examples),
- the POOL Tutorials which can be downloaded from the AIDA training section,
- as well as further Examples and FAQs located on the AIDA support area of the SimProD-Homepage.

4 The AIDA Studio

'AIDA Studio' is an IDE, allowing you comfortable application development.

'AIDA Studio' provides project management, an editor with syntax highlighting, automatic frame work generation in order to be able to add 'import declarations' and 'functions', project building with batch-files or ants (XML based scripts), tips and tricks in form of a list of interesting productivity features, bookmarks as a simple way to navigate to frequently used resources etc.

So let's start to create our first application, which is of course 'Hello World!'.

4.1 Starting AIDA Studio

Select 'AIDA Studio' from the 'BSK AIDA' menu to start 'AIDA Studio'.

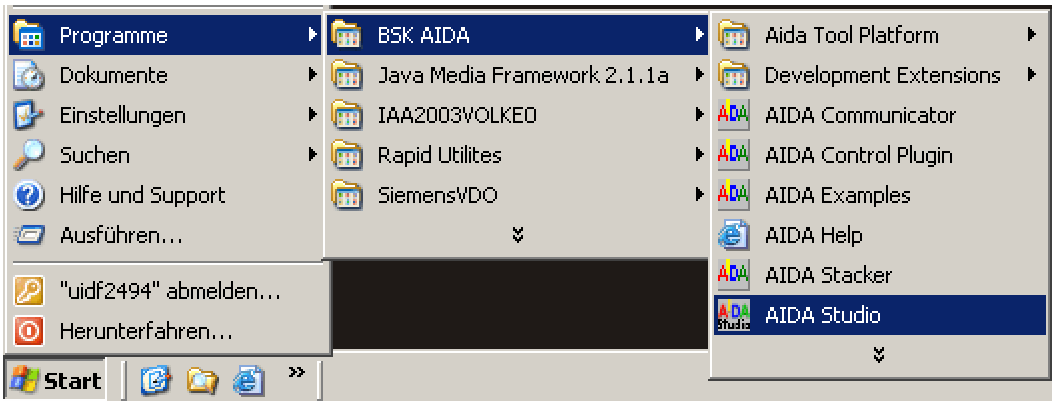


Figure 1: Starting AIDA Studio

4.2 Creating a new project

Select 'File' à 'New' à 'Project…' to create a new project.

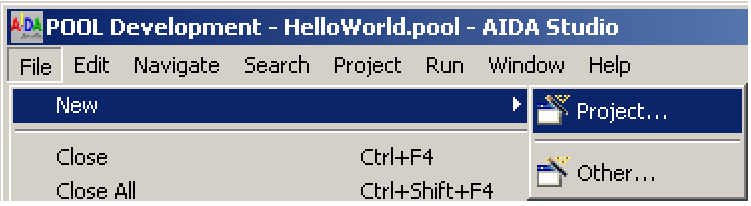


Figure 2: Creating a new project

Select 'POOL' in the wizard dialog and click on the 'Next' button.

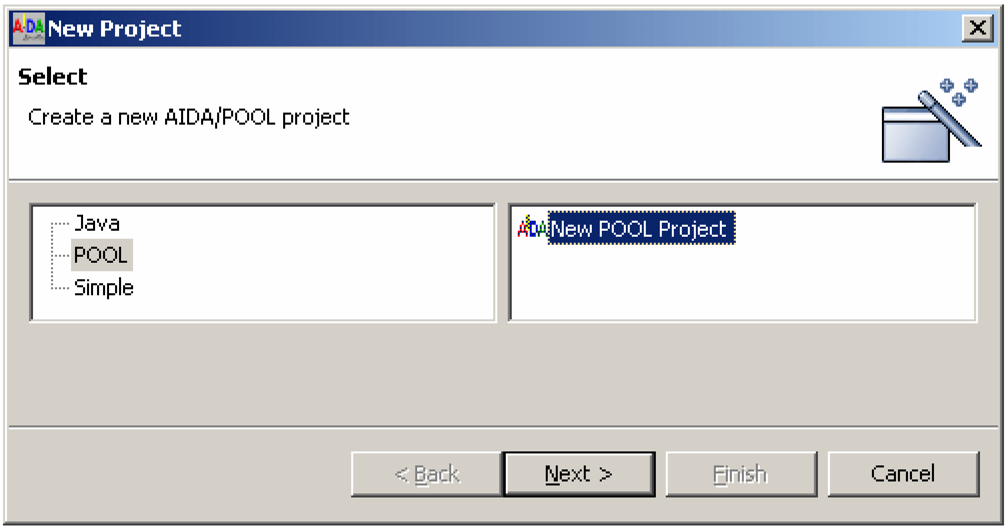
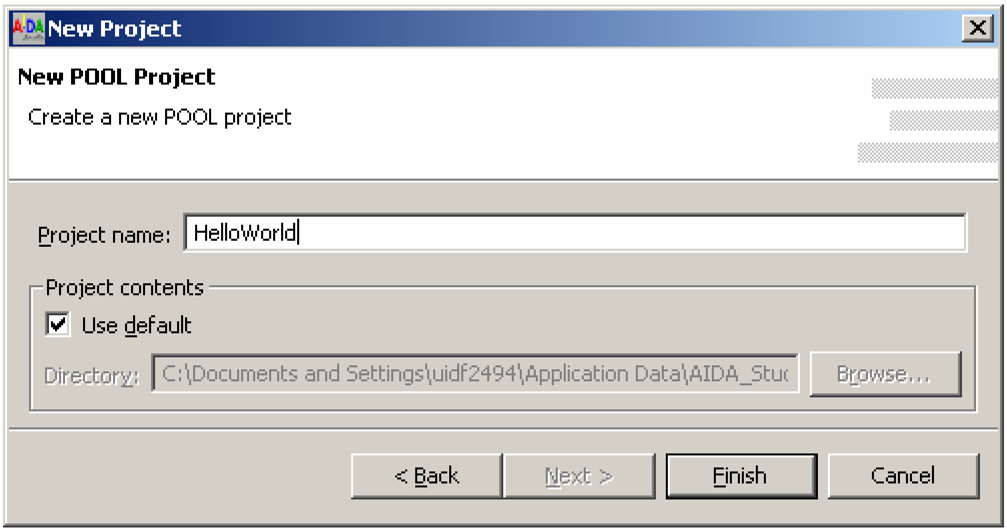


Figure 3: Selecting the project type

Enter a project name ('HelloWorld' in this case) and click on the 'Finish' button.

In order to be able to change the directory, you just have to deactivate ‘Use default’. In our case we are using the default settings. Thus a new 'HelloWorld' folder is generated in the workspace of 'AIDA Studio' containing all project data. We will change the location of the workspace later on.



Project name: HelloWorld

☑ Use default

Directory: C:\Documents and Settings\uidf2494\Application Data\AIDA_Stu

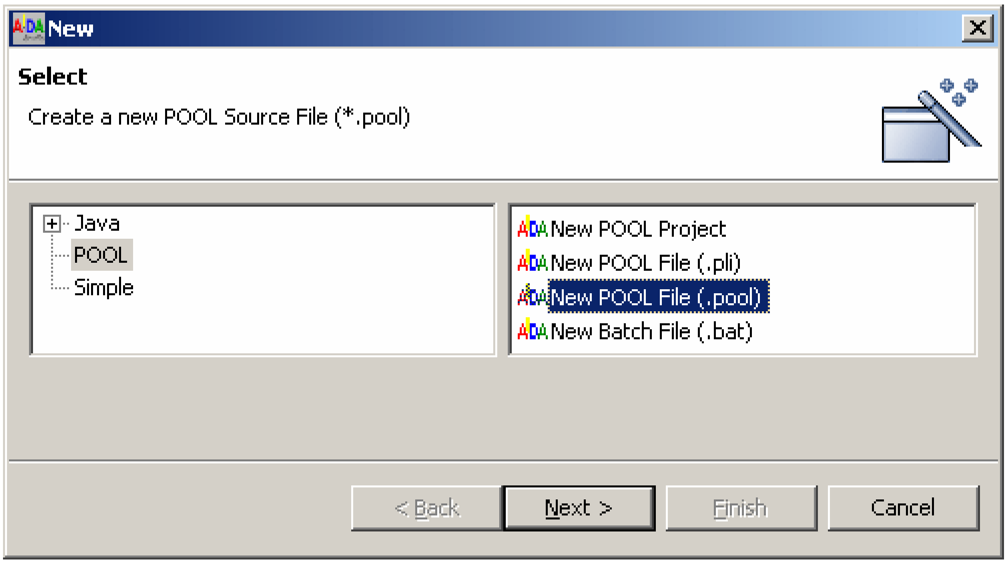
Figure 4: Defining the project name

4.3 Creating a source code file

Select 'File' à 'New' à 'Other…' to create a source code file.

Select 'POOL' as project type in the left list box.

Select 'New POOL File (.pool)' and click on the 'Next' button.



Select

Create a new POOL Source File (*.pool)

Figure 5: Selecting the source code type

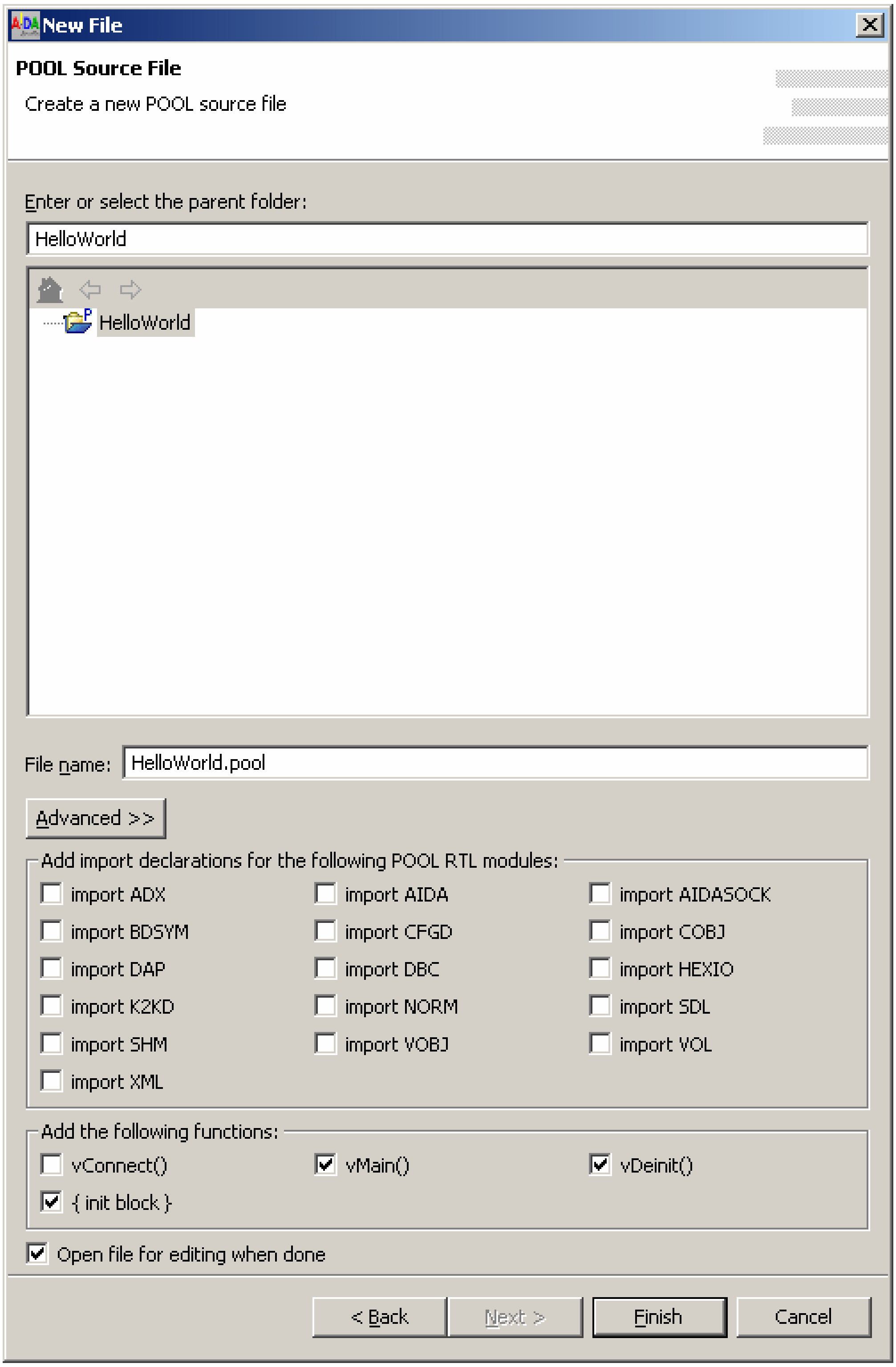
Enter the file name ('HelloWorld.pool' in this case) and click the 'Finish' button.

When you create a source code file, you can include 'import declarations' and bodies of 'functions'.

In Figure 6 you can see the available 'import declarations' this means libraries.

Examples:	import ADX	{ AIDA Data Exchange Library }
	import VOL	{ Visual Objects Library }

If you click on the 'Advanced' button that is shown below the file name, additional options are displayed.



POOL Source File

Create a new POOL source file

Enter or select the parent folder: HelloWorld

HelloWorld

File name: HelloWorld.pool

Add import declarations for the following POOL RTL modules:

☐ import ADX ☐ import AIDA ☐ import AIDASOCK
☐ import BDSYM ☐ import CFGD ☐ import COBJ
☐ import DAP ☐ import DBC ☐ import HEXIO
☐ import K2KD ☐ import NORM ☐ import SDL
☐ import SHM ☐ import VOBJ ☐ import VOL
☐ import XML

Add the following functions:

☐ vConnect() ☑ vMain() ☑ vDeinit()
☑ { init block }

☑ Open file for editing when done

Figure 6: Creating a new source file

Below you can find the code framework generated automatically.

Generated 'Hello World' framework

```
{=============================================================================

HELLOWORLD:
(c) 2003

Author:   ,

$Revision:  $
$Date:  $

Modification history:
2003-10-22 /
- First revision
=============================================================================
}
module HELLOWORLD;



private


{ vMain:  }
{ ---------------------- }
procedure vMain;
begin

end; { vMain }


{ vDeinit: Module deinit }
{ ---------------------- }
procedure vDeinit;
begin

end; { vDeinit }


{ Module init }
{ ----------- }
begin

end.

{ eof HelloWorld.pool }
```

Now, implement the 'Hello World!' functionality in the 'vMain' procedure (see below).

Implementing 'Hello World!'

```
{ vMain:  }
{ ---------------------- }
procedure vMain;
begin
  Writeln("Hello World!");
end; { vMain }
```

The program is already more complex than necessary. The 'Writeln' command could have been located in the 'Init' routine also. Don't worry if you don't understand the POOL code. It is the intention of this tutorial to teach you how to use the AIDA development environment.

A detailed introduction into the programming language POOL is subject of the 'POOL-Tutorial, Part 1'. It is a part of the AIDA help ('Start' à 'Programs' à 'BSK AIDA' à 'AIDA Help'.

4.4 Compiling the program

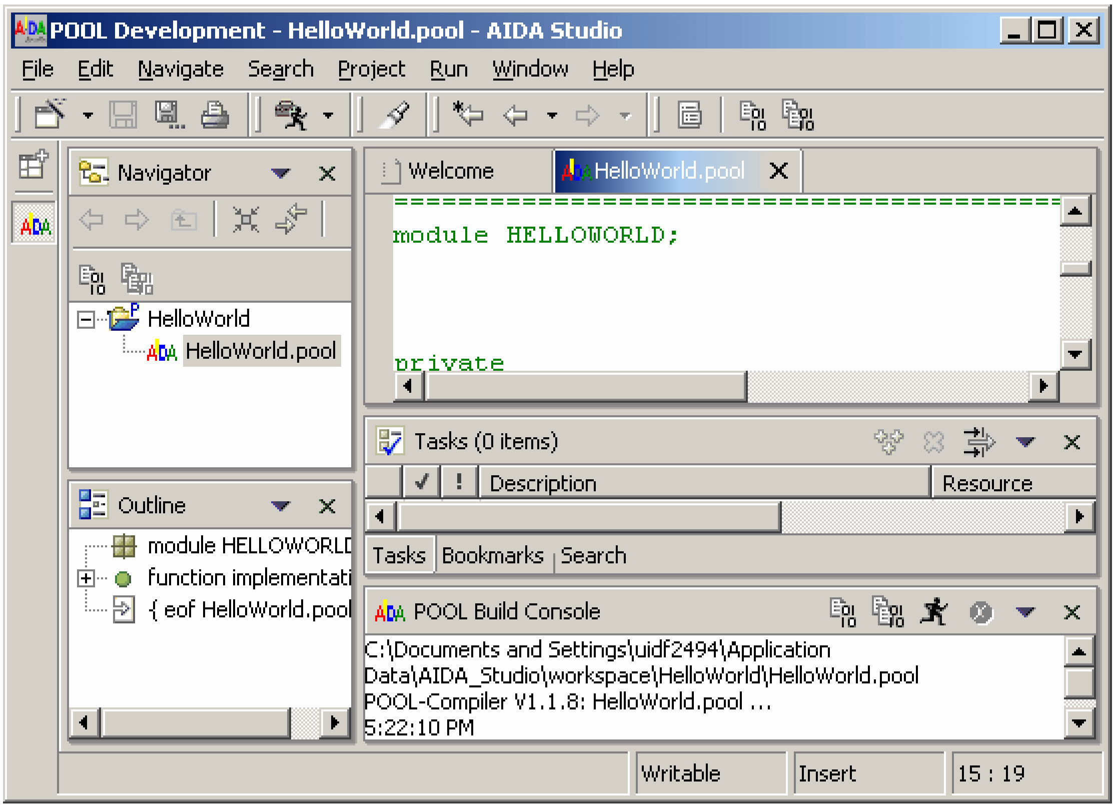
4.4.1 Compiling a single file

Before the program can be executed on the AIDA 'Commander', you first have to compile the source code. The compiler produces a *.pi file which can be executed by the P-code Interpreter pi.exe.

Click the button in the toolbar to compile the source file (this button is only active if the source code 'HelloWorld.pool' is selected in the source code editor).

You can find this button either in the 'Navigator', in the toolbar at the top of the page or downright in the 'POOL Build Console'.

The compiled file 'HelloWorld.pi' is stored in the 'HelloWorld' directory. We will show how to execute it later on.



```
==========================================
module HELLOWORLD;


private
```

```
C:\Documents and Settings\uidf2494\Application
Data\AIDA_Studio\workspace\HelloWorld\HelloWorld.pool
POOL-Compiler V1.1.8: HelloWorld.pool ...
5:22:10 PM
```

Figure 7: AIDA Studio

4.4.2 Compiling using Pool-all batch

In order to compile more than one file at a time you have to use the Pool-all batch button. But first, execute the following steps to generate a 'PoolAll' example:

1. Generate a new project called 'PoolAll' (see also Chapter 4.2).
2. Select the new project in the 'Navigator'.
3. Add a source code file called 'NoSense.pool' (see also Chapter 4.3).
4. Add another source code file 'NonSense.pool' (see also Chapter 4.3, but this time generate code without 'vMain' – this means deselect the checkbox 'vMain()' in the wizard).

Figure 8 shows the resulting file structure.

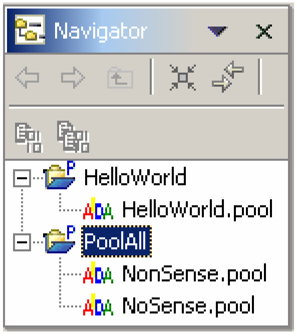


Figure 8: Structure of the second project 'PoolAll'

Now you have to create the 'pool_all.bat' file:

1. Right click the 'PoolAll' project in the 'Navigator' and select 'New' à 'Other…'. The 'New' file wizard opens.
2. This time select 'Simple' on the left hand side and afterwards 'File' on the right hand side (see also Figure 5) and click 'Next'.
3. Enter 'pool_all.bat' as 'File name' (confirm the appearing warning dialogues).
4. Right click 'pool_all.bat' and select 'Open With' à 'Text Editor' in order to edit the file.
5. First define the compiler options with the command `set POOLOPT= -q -m –f`. These settings will produce well formatted error messages.
   In order to gain some additional information on compiler switches open the 'Command Prompt' application and enter 'pool'.
6. To compile the file 'NonSense.pool' you have to call the compiler. This is done by the command `"%AIDABIN%POOL.EXE"` with the global system variable `%AIDABIN%` in which the AIDA installation path is stored. The compiler call must be followed by the name of the file to be compiled. In our case this is 'NonSense'. Thus the resulting command is `"%AIDABIN%POOL.EXE" NonSense`.
7. To compile the second file, the previous step must be applied to 'NoSense.pool' correspondingly. Now 'PoolAll' is ready for compilation.

Example 'pool_all.bat' for the 'PoolAll' project

```
@echo off

rem SETTING COMPILER PARAMETERS
set POOLOPT= -q -m -f

rem CALL COMPILER
"%AIDABIN%POOL.EXE" NonSense
"%AIDABIN%POOL.EXE" NoSense
```

8. Click on 'NonSense.pool' respectively 'NoSense.pool' in the 'Navigator' to open it. Click into the editor where the source code is displayed. Now the 'pool_all.bat' button on the upper toolbar is enabled. As an alternative this button is also located in the 'Navigator' (this one is enabled if 'pool_all.bat' is selected) and downright in the 'POOL Build Console' (this one is always enabled).

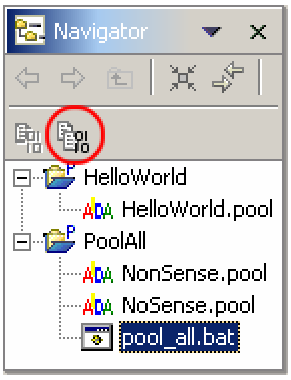
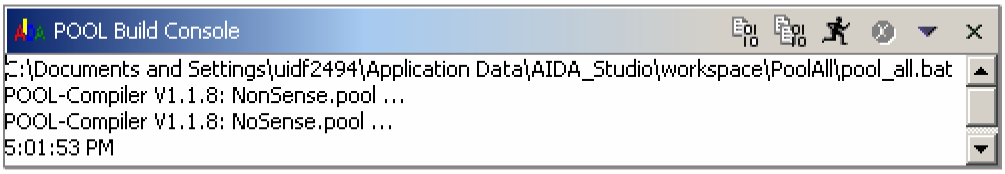


Figure 9: 'pool_all.bat' and batch file execution button

The text files (*.pool) are converted into the interim code of the virtual machine (.pi). This code can be executed by the **P**-code **I**nterpreter (**PI.exe**).

The following Figure shows the output on the 'POOL Build Console' after successful compilation.



```
C:\Documents and Settings\uidf2494\Application Data\AIDA_Studio\workspace\PoolAll\pool_all.bat
POOL-Compiler V1.1.8: NonSense.pool ...
POOL-Compiler V1.1.8: NoSense.pool ...
5:01:53 PM
```

Figure 10: Output on the 'POOL Build Console'

4.5 Changing the workspace

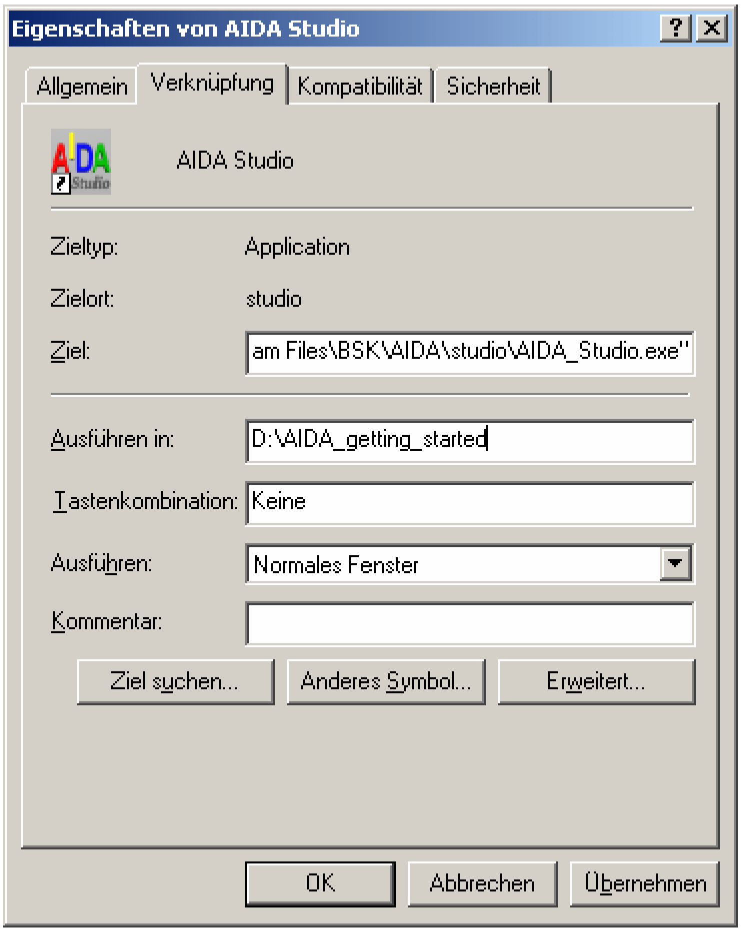
The central place for a user's data files is called a workspace. You can think of the platform workbench as a tool that allows the user to navigate and manipulate the workspace.

The workspace contains a collection of resources. There are three different types of resources - projects, folders, and files. A project is a collection of any number of files and folders. It is a container for organising other resources that are related to a specific area. Files and folders are just like files and directories in the file system. Workspace resources are organised into a tree structure, with projects at the top, and folders and files underneath.

A workspace can have any number of projects.

You can also change your workspace:

1. Open your file browser (e.g. 'Windows Explorer'), right click the 'workspace' folder (on Microsoft XP workstations this directory is usually located in 'C:\Documents and Settings\USERNAME\Application Data\AIDA_Studio') and select 'Cut' or 'Copy'.
2. Right click another directory and select 'Paste'. It is also possible to rename the 'workspace' directory.
3. Select 'Start' à 'Programs' à 'BSK AIDA' à right click 'AIDA Studio' à 'Properties'. The following Figure shows the 'AIDA Studio' properties panel.



AIDA Studio

Zieltyp: Application

Zielort: studio

Ziel: am Files\BSK\AIDA\studio\AIDA_Studio.exe"

Ausführen in: D:\AIDA_getting_started

Tastenkombination: Keine

Ausführen: Normales Fenster

Kommentar:

Figure 11: 'AIDA Studio' properties

4. Enter the new workspace directory (here 'D:\AIDA_getting_started') into the 'Start in:' box and press the 'OK' button.

From now on 'AIDA Studio' will use this newly defined workspace.

Remark: It is also possible to use different links to different workspaces.

Before we can start the program we have to discuss the 'Commander'. If you are familiar with the 'Commander' feel free to skip the following chapter.

5 The Commander

The 'Commander' is a shell to send POOL commands and to display results. These commands can be executed via menus, toolbars or command line inputs (commands from menus and toolbars are handed over to the command line). They are written in POOL, compiled automatically and interpreted by the PI runtime engine.

5.1 Commander modes

However, a specialty must be considered. The commander can run in two different modes. On the one hand, in the so called interactive mode where the command is interpreted and the requested functionality is executed. After completion the next 'Commander' instruction can be executed. On the other hand, the program keeps control. Thus it isn't possible to send commands to the running application.

What are the differences between these two approaches?

5.1.1 Program keeps control

In the following the program flow is described.

1. Execution of the 'init' section (see also Chapter 4.3).

Initialisation section

```
{ Module init }
{ ----------- }
begin
  { insert initialisation functionality here }
end.
```

2. Execution of the 'vMain' section (the functionality of the program is controlled by this procedure).

'vMain' section

```
{ vMain:  }
{ ---------------------- }
procedure vMain;
begin
  { implement program flow functionality here }
end; { vMain }
```

3. Execution of the 'vDeinit' section.

Deinitialisation section

```
{ vDeinit: Module deinit }
{ ---------------------- }
procedure vDeinit;
begin
  { insert deinitialisation functionality here }
end; { vDeinit }
```

4. Now another program can be started by the 'Commander' (step 1).

5.1.2 Interactive mode

Unlike the previous case the interactive mode doesn't have a 'vMain' routine. The following program flow results:

1. Execution of the 'init' section (see above).
2. The program remains in the interactive mode and waits for user input (e.g. functionalities can be called). Now the user takes over the role of 'vMain'!
3. The interactive mode is terminated by calling the procedure 'Halt'. This again is an interactive input from the console.
4. Execution of the 'vDeinit' section (see above).

Be aware that in the current version it is not possible for threads to send output data and commands (e.g. menu configurations) to the 'Commander' console because it is waiting for user input!

5.2 Setting up a Commander configuration

Select 'AIDA Examples' from the 'BSK AIDA' menu to start a 'Commander' containing an AIDA examples configuration.

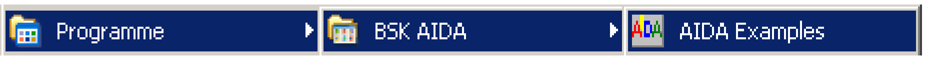
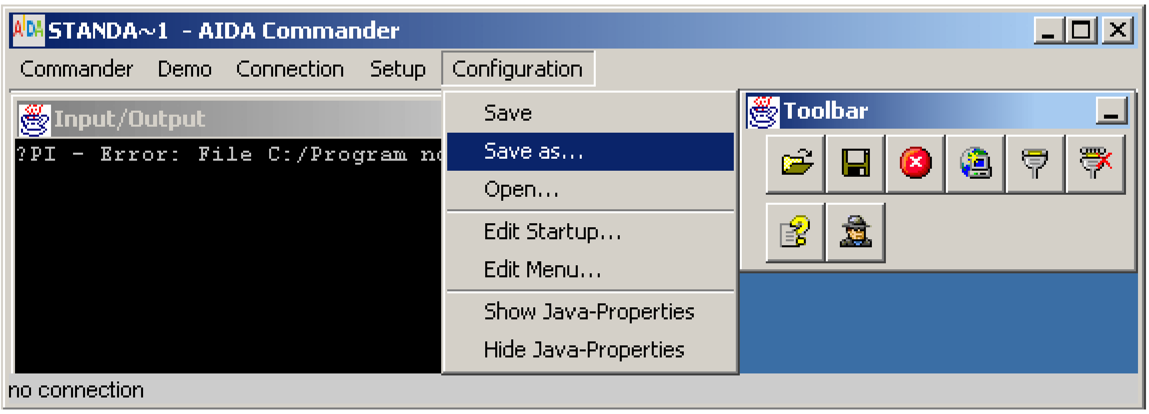


Figure 12: Starting the 'Commander'

Select 'Configuration' à 'Save as…', name the current configuration with 'HelloWorld.cmdr-cfg' and save the file in your 'HelloWorld' directory.



```
?PI - Error: File C:/Program no
```

no connection

Figure 13: Saving the 'Commander' configuration

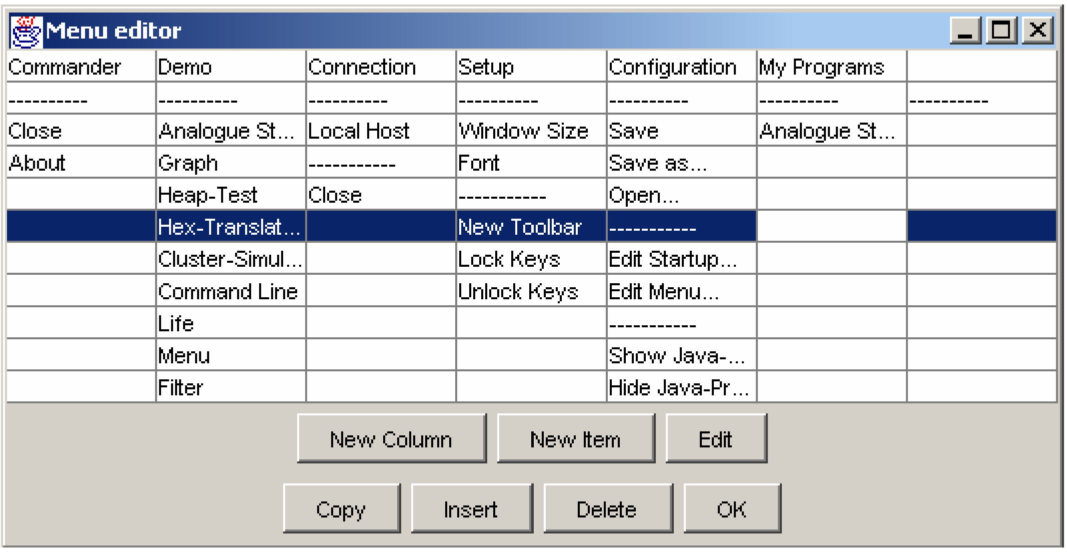
5.3 Execute the program

There are different ways to start a program.

5.3.1 Start program from Commander menu

You can add your program to the 'Commander' menu and start it using menu items. To do this, you have to execute the following steps:

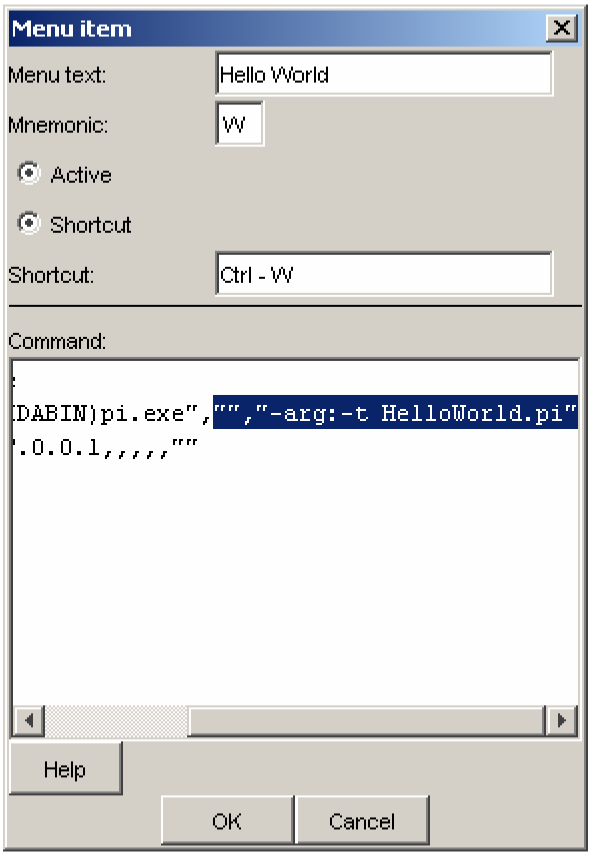
1. Double click the 'HelloWorld.cmdr-cfg' configuration if it is not already running.
2. Right click on the 'Commander's desktop and select 'Edit mode on'.
3. Select 'Configuration' à 'Edit Menu…' from the 'Commander' menu (see also Figure 13).
4. Click the 'New Column' button, enter the name for your menu (e.g. 'My Programs' and confirm with the 'OK' button.
   IMPORTANT: The menu items can't be entered directly. You must use the buttons at the lower end of the window.
5. Select the 'Analogue Stopwatch' field in the 'Demo' column and click the 'Copy' button.
6. Select an empty field in the column you have created in step 4 and click the 'Insert' button.



Commander	Demo	Connection	Setup	Configuration	My Programs	
----------	----------	----------	----------	----------	----------	----------
Close	Analogue St...	Local Host	Window Size	Save	Analogue St...	
About	Graph	-----------	Font	Save as...		
	Heap-Test	Close	-----------	Open...		
	Hex-Translat...		New Toolbar	-----------		
	Cluster-Simul...		Lock Keys	Edit Startup...		
	Command Line		Unlock Keys	Edit Menu...		
	Life			-----------		
	Menu			Show Java-...		
	Filter			Hide Java-Pr...		

Figure 14: Menu editor

7. Click the 'Edit' button, modify 'Menu text' and the command according to the file to be executed (in our case to 'HelloWorld.pi' - see highlighted text in the Figure below). If you like you can also set 'Mnemonic' and 'Shortcut' keys.



Menu text: Hello World

Mnemonic: W

● Active

● Shortcut

Shortcut: Ctrl - W

Command:

```
:
DABIN)pi.exe","","-arg:-t HelloWorld.pi"
'.0.0.1,,,,,""
```

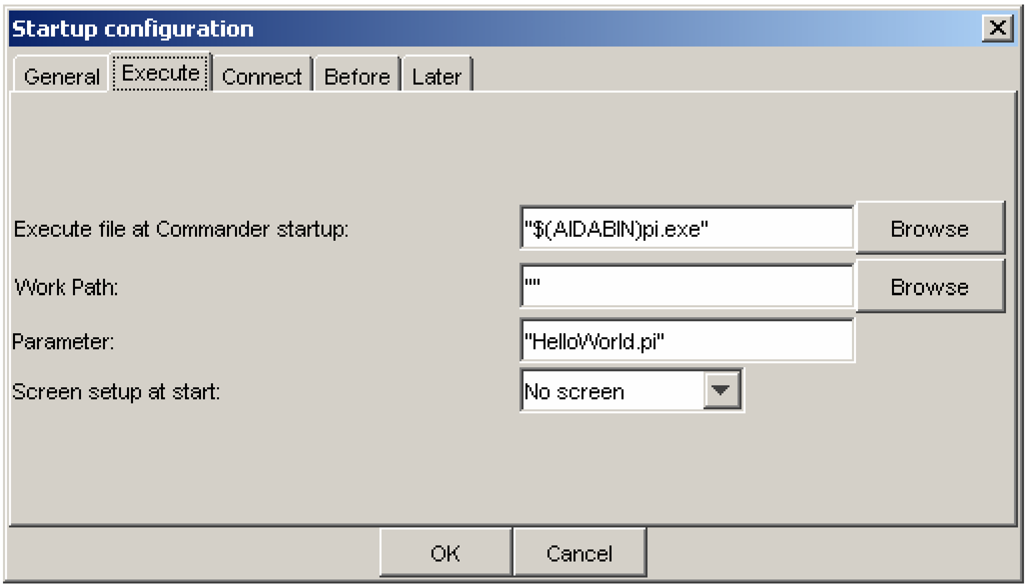
Figure 15: Modifying the menu logic

8. It is also possible to define a path (relative to the folder of your 'Commander' configuration). This has to be done between the first quotation marks of the highlighted text in Figure 15.
   IMPORTANT: Be aware that slashes instead of backslashes must be used for the path! The double quotes are necessary if the path contains spaces!
9. Close the 'Menu item' and the 'Menu editor' window by pressing their 'OK' buttons.
10. Save the new Commander configuration (see also 5.2)

Now you can start your program from the menu and 'Hello World!' should appear on the console.

5.3.2 Automatic start of a program

1. Right click on the 'Commander's desktop and select 'Edit mode on'.
2. Right click on the Commander's desktop and select ‘Edit start batches'.
3. Select the 'Execute' folder and insert 'Work path' as well as 'Parameter' (see Figure below).



Execute file at Commander startup: "$(AIDABIN)pi.exe"

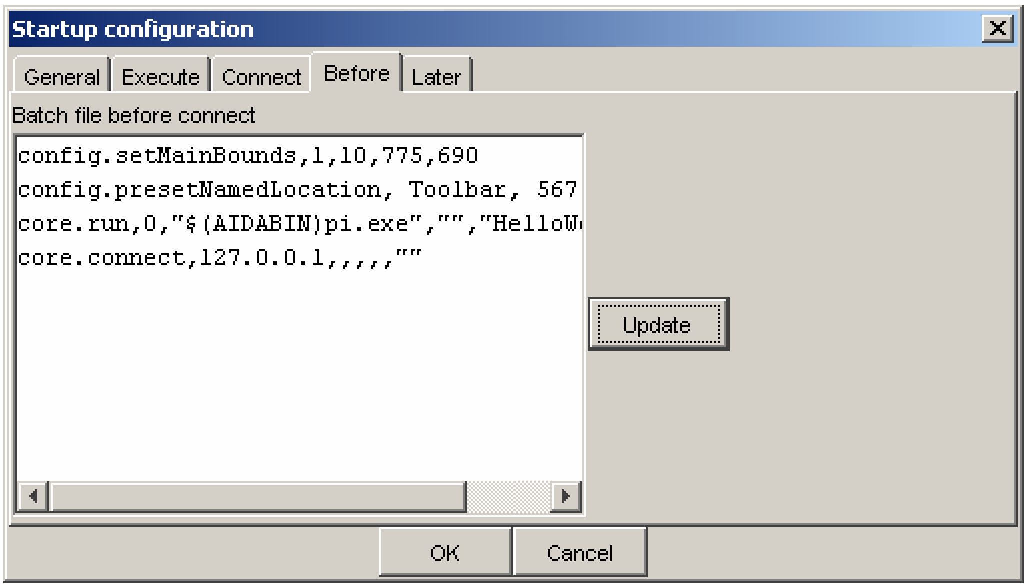
Work Path: ""

Parameter: "HelloWorld.pi"

Screen setup at start: No screen

Figure 16: Startup configuration

4. Select the 'Before' folder of the 'Startup configuration', click the 'Update' button and afterwards 'OK'.



```
config.setMainBounds,1,10,775,690
config.presetNamedLocation, Toolbar, 567
core.run,0,"$(AIDABIN)pi.exe","","HelloW
core.connect,127.0.0.1,,,,,""
```

Figure 17: Updating the 'Startup configuration'

5. Save the new 'Commander' configuration via 'Configuration' à 'Save' (see also Figure 13).
6. Close the 'Commander'.

After a double click on the 'HelloWorld.cmdr-cfg' configuration the 'Commander' is started and the 'Startup configuration' is executed. In our case 'Hello World!' should appear on the console.

5.3.3 Run a program from the AIDA Studio

There are two possibilities to start a program from the AIDA Studio.

5.3.3.1 Using the Run button

If the compilation was successful (see also Chapter 4.4), an executable file with the file extension '.pi' is generated. If not, an error message is displayed. The error treatment and the location of the output folder for compiled files can be specified with the help of compiler options.

1. Open any 'Commander' configuration e.g. the 'AIDA Examples' (see also Figure 12).
2. Disconnect the TCP/IP communication between the 'Commander' and the P-code interpreter (virtual machine) by pressing the 'Commander' toolbar button . The connection status is displayed in the lower left corner of the 'Communicator' no connection.

3. After 'HelloWorld.pool' was compiled successfully it can be started from the 'AIDA Studio' via the button . It is located in the 'POOL Build Console'.
   Now the code is running on the virtual machine.
4. Finally connect the P-code interpreter to the 'Commander' by pressing the toolbar button . After successful connection 'Hello World!' should appear on the console.

Disconnecting and connecting can be also done by the menu 'Connection' (see also Figure 13).

5.3.3.2 Using external tools

1. In the 'AIDA Studio' Select 'Run' à 'External Tools' à 'External Tools…' in order to create, manage and run configurations.

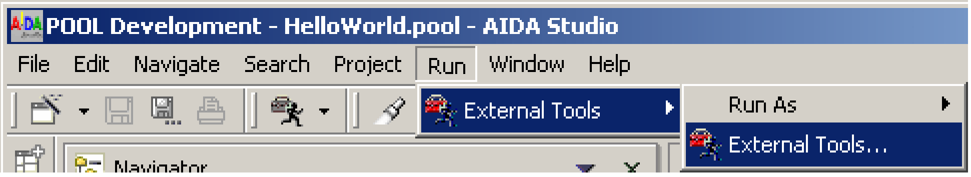


Figure 18: Opening the 'External Tools…' dialog

2. Select 'Program' as configuration in the left list box and click on the 'New' button.

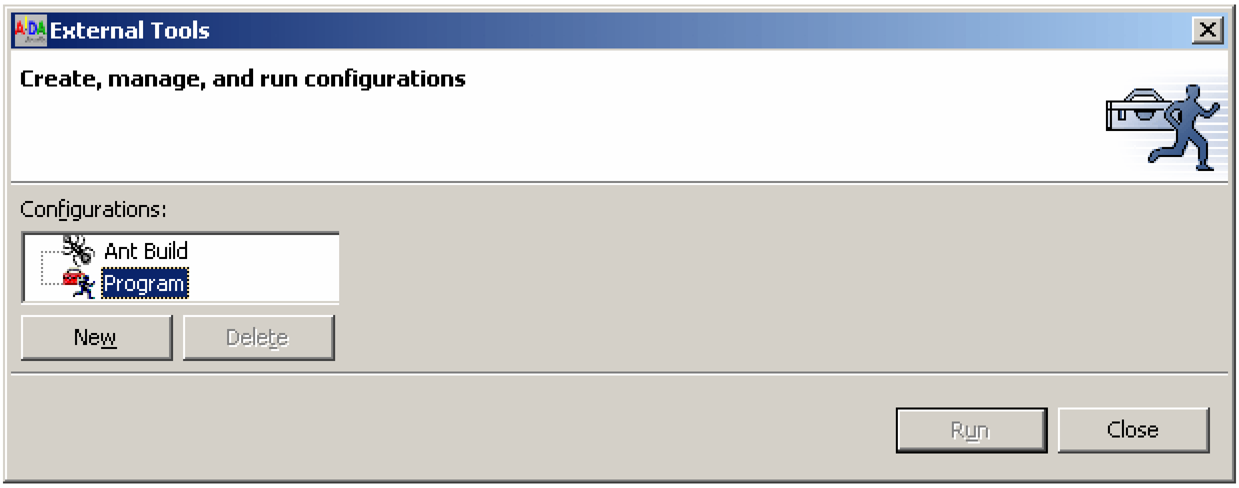
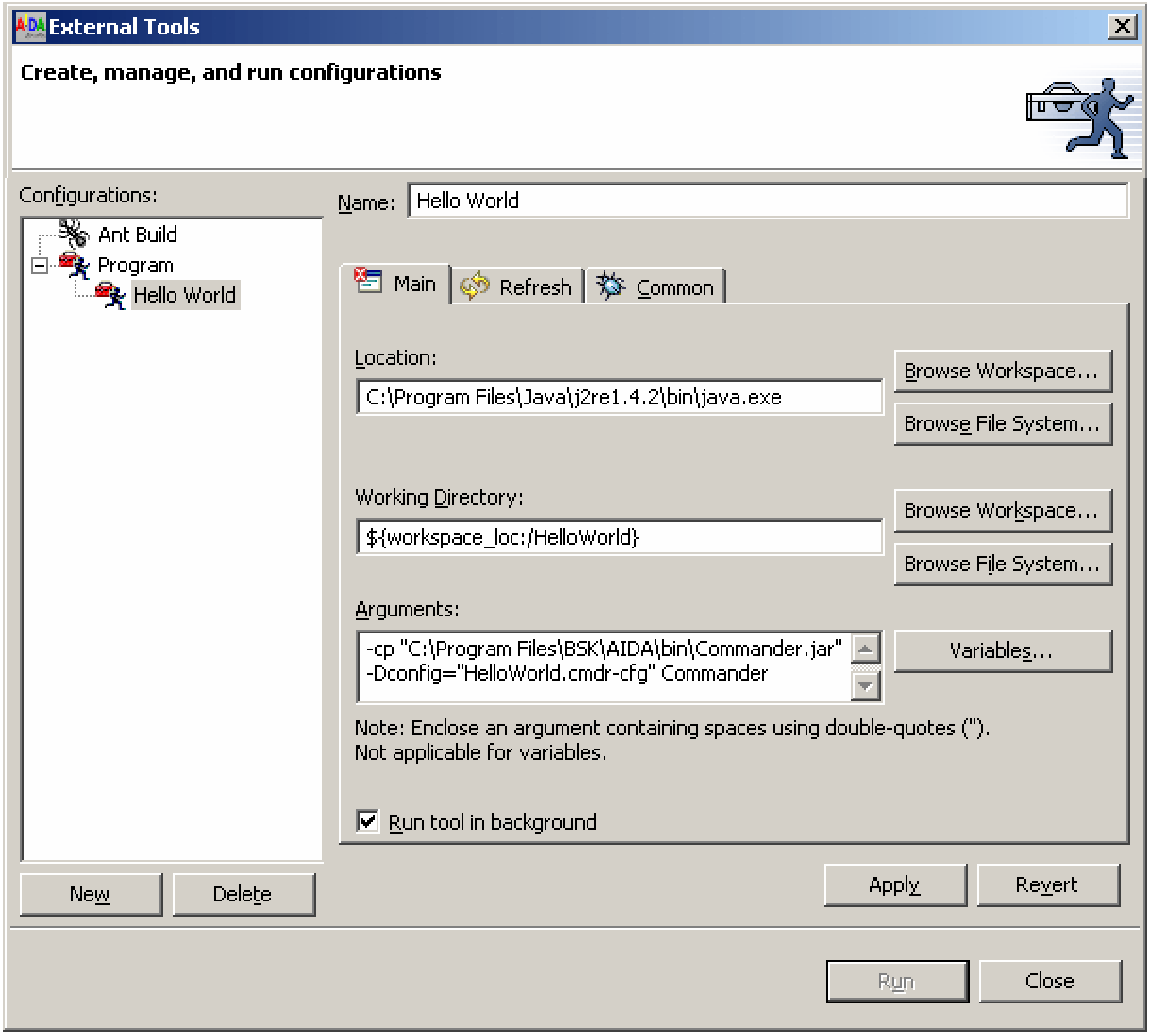


Figure 19: Building a new configuration

3. Enter your configuration name (e.g. 'Hello World'), the location of the JAVA runtime environment on your machine, the work directory (use the 'Browse Workspace…' button and select the corresponding workspace) and arguments (adjust the argument of the sample in the Figure below to the AIDA path of your machine and to the 'Commander' configuration that should be called).



External Tools

Create, manage, and run configurations

Name: Hello World

Location:
C:\Program Files\Java\j2re1.4.2\bin\java.exe

Working Directory:
${workspace_loc:/HelloWorld}

Arguments:
-cp "C:\Program Files\BSK\AIDA\bin\Commander.jar"
-Dconfig="HelloWorld.cmdr-cfg" Commander

Note: Enclose an argument containing spaces using double-quotes (").
Not applicable for variables.

☑ Run tool in background

Figure 20: 'External Tools' Configurations

4. After clicking on the 'Apply' button the entered instructions are executed. In case of the 'HelloWorld' application the 'Commander' should be started and 'Hello World!' should appear on the console.
5. Afterwards a new menu item ('Hello World') will appear in the 'Run' menu. As an alternative it can be selected also via the 'Run' button.

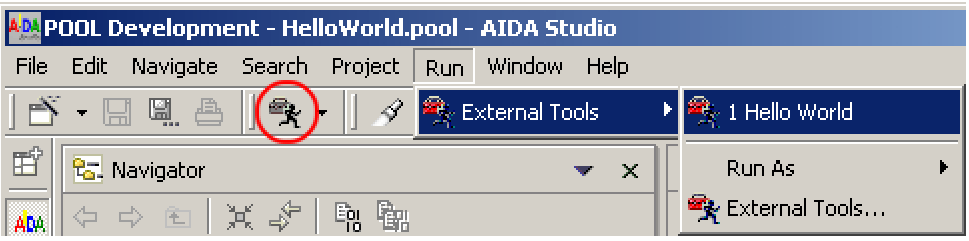


Figure 21: Starting an external application via 'Run' menu or 'Run' button.

Conclusion

Now you should be able to create and execute your own projects using AIDA Studio.

For further information about POOL see the tutorials located on the SimProD homepage.

For further information on AIDA Studio see the annex of this document and the Studio help.

6 Annex

A1 Special Features of AIDA Studio

A1.1 Tips and Tricks

This command will open a list of interesting productivity features that you may not have discovered. In order to open it, select 'Tips and Tricks…' from the 'Help' menu of the Studio.

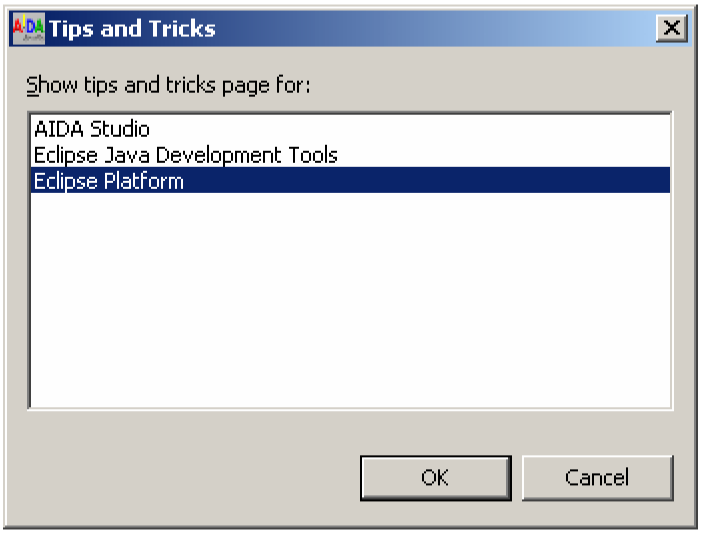

Tips and Tricks
Show tips and tricks page for:
AIDA Studio
Eclipse Java Development Tools
Eclipse Platform
OK
Cancel

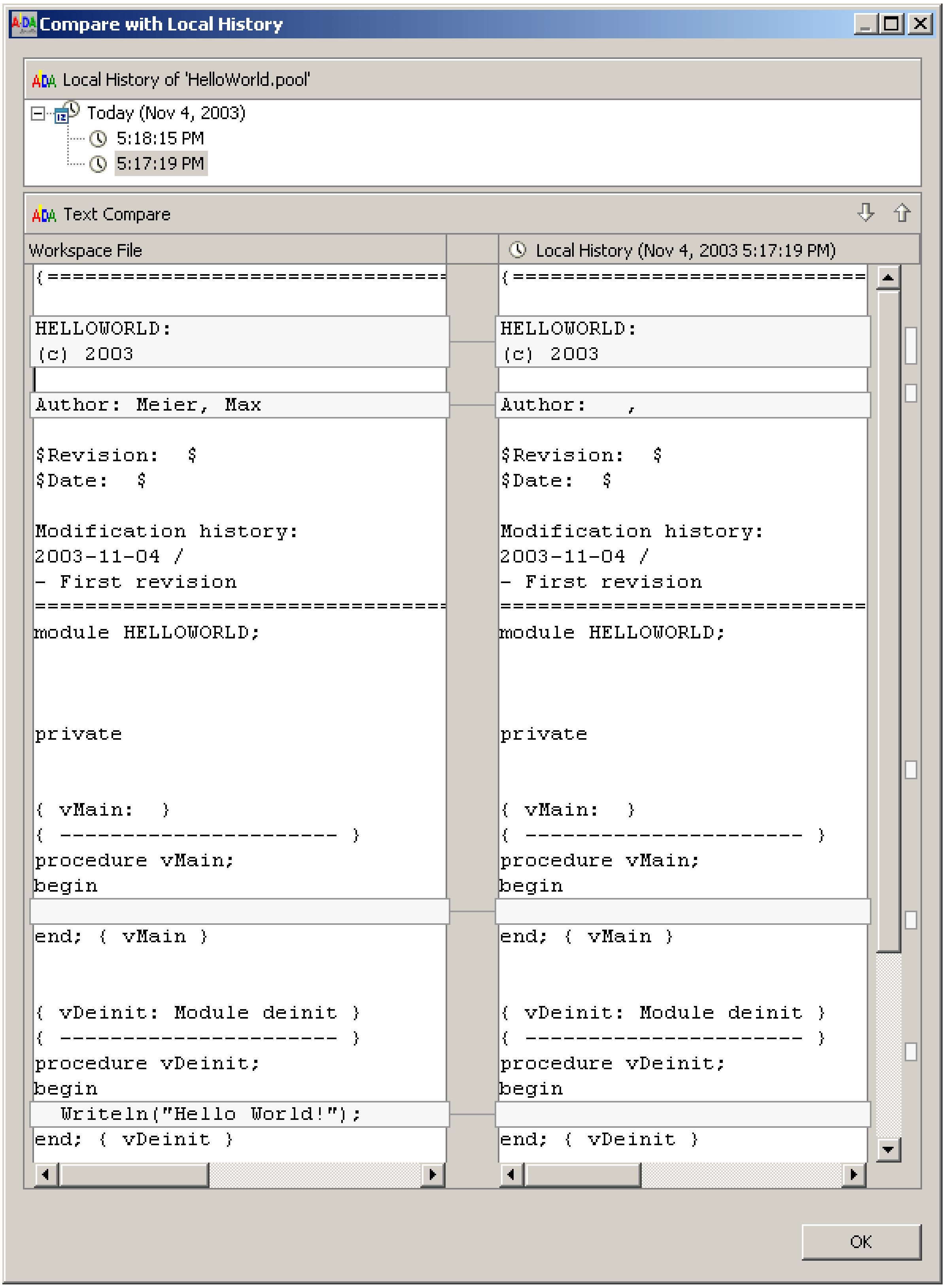
Figure 22: Tips and Tricks

A1.2 Local History

Local history of a file is maintained when you create or modify a file. Each time you edit and save a file, a copy of it is saved. This allows you to compare your current file state to a previous state, or replace the file with a previous state. Each state in the local history is identified by the date and time the file was saved.

To view the local history of a file, right click it in the 'Navigator' and choose 'Compare with' > 'Local History...'.

You can select different local states in the list (see Figure below), which are compared against the current file. You can also revert to the local history when you select a file and select the 'Replace With' > 'Local History...' menu item.


Compare with Local History
Local History of 'HelloWorld.pool'
Today (Nov 4, 2003)
5:18:15 PM
5:17:19 PM
Text Compare
Workspace File
Local History (Nov 4, 2003 5:17:19 PM)
OK

```
{=================================

HELLOWORLD:
(c) 2003

Author: Meier, Max

$Revision:  $
$Date:  $

Modification history:
2003-11-04 /
- First revision
==================================
module HELLOWORLD;



private


{ vMain:  }
{ ---------------------- }
procedure vMain;
begin

end; { vMain }


{ vDeinit: Module deinit }
{ ---------------------- }
procedure vDeinit;
begin
  Writeln("Hello World!");
end; { vDeinit }
```

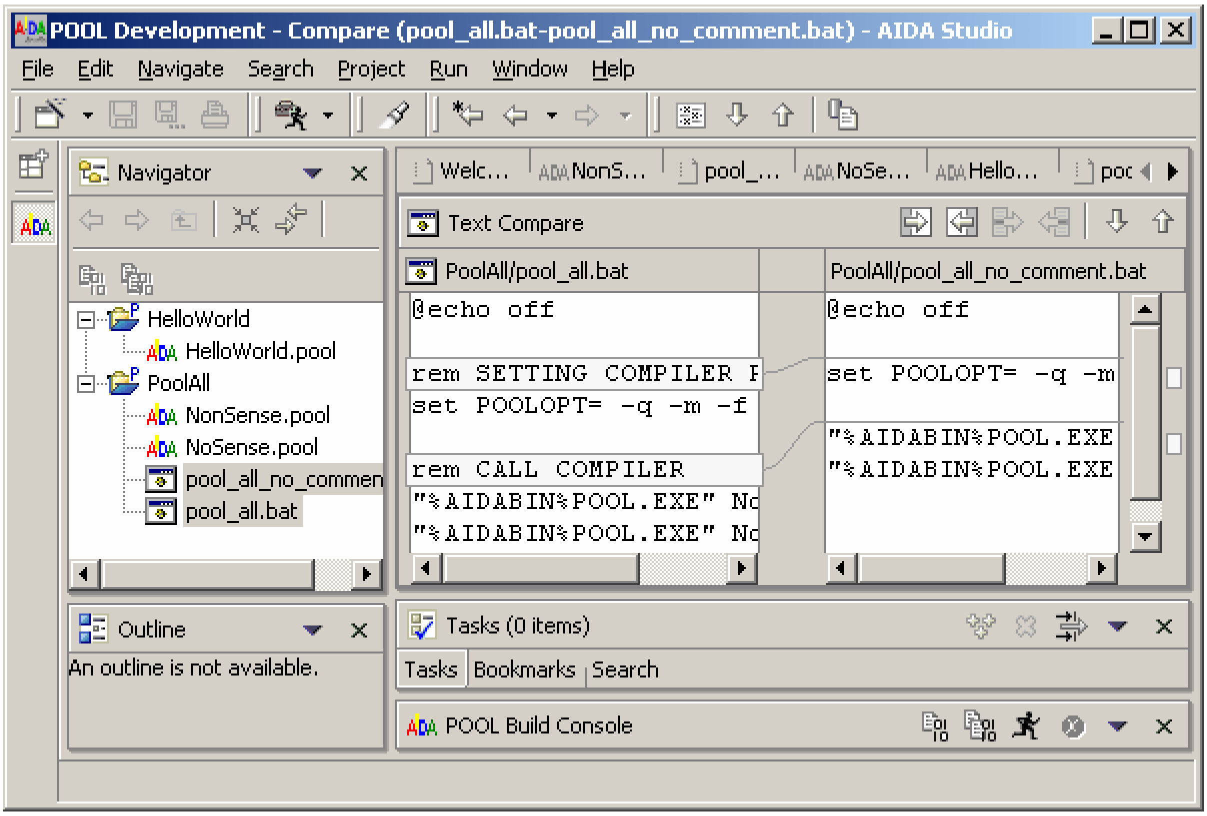
```
{============================

HELLOWORLD:
(c) 2003

Author:  ,

$Revision:  $
$Date:  $

Modification history:
2003-11-04 /
- First revision
=============================
module HELLOWORLD;



private


{ vMain:  }
{ ---------------------- }
procedure vMain;
begin

end; { vMain }


{ vDeinit: Module deinit }
{ ---------------------- }
procedure vDeinit;
begin

end; { vDeinit }
```

Figure 23: 'Compare with Local History'

A1.3 Compare functionality

You can also compare two files with each other.

Select the two files in the 'Navigator', right click them and select 'Compare With' à 'Each Other'.


POOL Development - Compare (pool_all.bat-pool_all_no_comment.bat) - AIDA Studio
File Edit Navigate Search Project Run Window Help
Navigator
HelloWorld
HelloWorld.pool
PoolAll
NonSense.pool
NoSense.pool
pool_all_no_commen
pool_all.bat
Welc… NonS… pool_… NoSe… Hello… poc
Text Compare
Outline
An outline is not available.
Tasks (0 items)
Tasks Bookmarks Search
POOL Build Console

PoolAll/pool_all.bat

```
@echo off
rem SETTING COMPILER F
set POOLOPT= -q -m -f
rem CALL COMPILER
"%AIDABIN%POOL.EXE" Nc
"%AIDABIN%POOL.EXE" Nc
```

PoolAll/pool_all_no_comment.bat

```
@echo off
set POOLOPT= -q -m
"%AIDABIN%POOL.EXE
"%AIDABIN%POOL.EXE
```

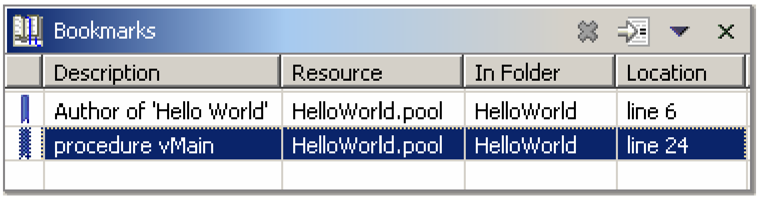
Figure 24: Comparing two files.

A1.4 Bookmarks

Bookmarks are a simple way to navigate through resources you are using frequently. You can set, remove and view them.

A Bookmark can be added to the currently selected editor row via 'Edit' à 'Add Bookmark…' or at the cursor position by means of the context menu (right click on the grey frame at the left side of the editor).

They can be viewed by selecting 'Window' à 'Show View' à 'Other…' à 'Basic' à 'Bookmarks' from the Studio menu. The displayed 'Bookmarks' view shows all bookmarks of the workspace.



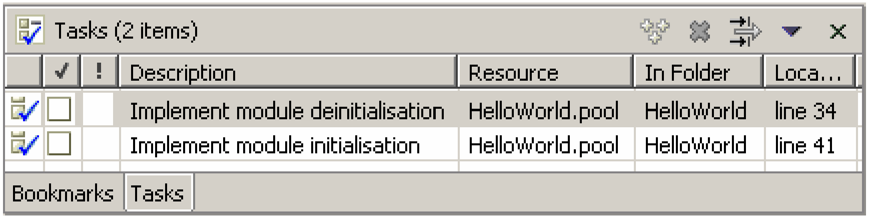
	Description	Resource	In Folder	Location
	Author of 'Hello World'	HelloWorld.pool	HelloWorld	line 6
	procedure vMain	HelloWorld.pool	HelloWorld	line 24

Figure 25: 'Bookmarks' view

A1.5 Tasks

With the help of tasks, open issues like to-do items can be tracked easily.

By adding a task manually ('Edit' à 'Add Task…'), you have the option of associating it with a resource so that you can use the 'Tasks' view ('Window' à 'Show View' à 'Other…' à 'Basic' à 'Tasks') to quickly open that resource for editing.



	✓	!	Description	Resource	In Folder	Loca…
	□		Implement module deinitialisation	HelloWorld.pool	HelloWorld	line 34
	□		Implement module initialisation	HelloWorld.pool	HelloWorld	line 41

Figure 26: Tasks view

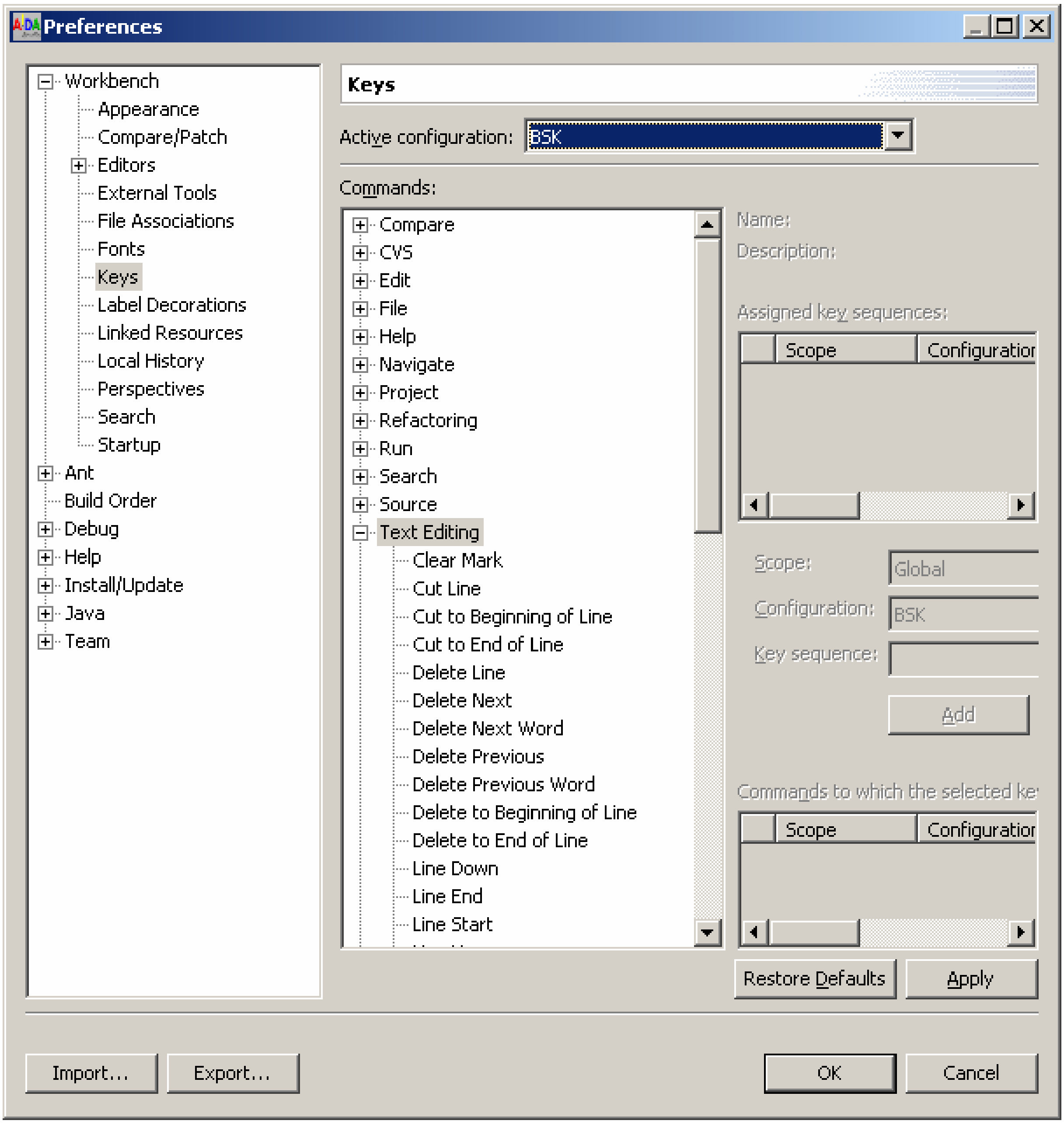
You can also add a tasks without resource association. This can be done by selecting the 'New Task' button of the 'Tasks' view.

A1.6 Shortcuts

All keyboard shortcuts can be adapted to user needs.

In the table below some standard shortcuts of the 'BSK' configuration are listed. In order to use this configuration select 'Windows' à 'Preferences', afterwards in the left tree 'Workbench' à 'Keys' and at last 'BSK' in the 'Active configuration' dropdown list.

Select 'apply' to use these settings.



Active configuration: BSK

Scope: Global

Configuration: BSK

Key sequence:

Figure 27: Selecting a key configuration

Category	Description	Key Sequence
Edit	Add Bookmark	Ctrl+F2
	Add Task	Ctrl+F3
	Copy	Ctrl+C
	Cut	Ctrl+X
	Delete	Delete
	Find and Replace	Ctrl+F, Ctrl+H, F5, F8
	Find Next	Ctrl+L, F3
	Find Previous	Shift+F3
	Paste	Ctrl+V
	Redo	F10
	Select All	Ctrl+A
	Undo	Ctrl+Z
File	Close	Ctrl+F5
	Close All	Ctrl+Shift+F5
	Properties	Alt+Enter
	Save	Ctrl+S
	Save All	Ctrl+Shift+S
Navigate	Go to Line	Ctrl+Alt+G
	Next	F4
Text Editing	Clear Mark	Ctrl+G
	Delete Line	Ctrl+Y
	Delete to Beginning of Line	Alt+0 Ctrl+K, Esc 0 Ctrl+K
	Delete to End of Line	Ctrl+Shift+Delete
	Line Down	Ctrl+N
	Line End	End
	Line Start	Home
	Next Column	Ctrl+M
	Previous Column	Ctrl+B
	Previous Word	Alt+B, Esc B
	Select Line End	Shift+End
	Select Line Start	Shift+Home
	Select Text End	Ctrl+Shift+End
	Select Text Start	Ctrl+Shift+Home
	Set Mark	Ctrl+2
	Swap Mark	Ctrl+W
	Text End	Ctrl+End
	Text Start	Ctrl+Home
	Toggle Overwrite	Insert
	Window End	Alt+End
	Window Start	Alt+Home
Window	Activate Editor	F12
	Next Editor	Ctrl+Tab
	Next Perspective	Ctrl+F8
	Next View	Ctrl+F6
	Previous Editor	Ctrl+Shift+Tab
	Previous Perspective	Ctrl+Shift+F8
	Previous View	Ctrl+Shift+F6
	Show System Menu	Alt+-
	Show View Menu	Ctrl+F10
	Switch to Editor	Ctrl+Alt+W

Figure 28: Keyboard shortcuts of the 'BSK' configuration