

VisualAge[®] C++ Professional for AIX[®]



Integrated Development Environment

Version 5.0

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page v.

Edition Notice

This edition applies to Version 5.0 of IBM® VisualAge C++ and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

© Copyright International Business Machines Corporation 1998, 2000. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v
Programming Interface Information	vii
Trademarks and Service Marks	vii
Industry Standards	viii

About This Book.	ix
-----------------------------------	-----------

Chapter 1. IDE Features and Tips 1

Start the IDE	1
Workbook	1
Toolbars in the IDE	4
Navigating the IDE	5
Improve IDE Workbook Performance	6
Discard Unused Views	6
Save the codestore less frequently	7
Open or Close a Subproject in Memory	7
Change Font Size in the IDE	8
Hide Pane Title Bars in the IDE	9
Managing Memory Use in the IDE	9

Chapter 2. Working with Projects in the IDE 11

Create a Project	11
View a Build Setup for a Project	11
Add a Subproject to a Project	12
Set up Dependencies Between Subprojects	13
Set Build Options for a Subproject	14
Create a Configuration File	14
Change a Build Setup	15
View a Build Setup for a Subproject	16
Add or Remove a Target in the IDE	17
Move a Source File within a Subproject	18
Add Source Files to a Project	19
Remove Source Files from a Project	20
Set Build Options for Source Files or Targets	21
Build	21
Dynamic Default Build Setting	22
View a Subproject After Processing	23
Control Build Messages	24
Use an External Editor with the IDE	24

Contacting IBM 27

Notices

Note to U.S. Government Users Restricted Rights -- use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd.
1150 Eglinton Avenue East
Toronto, Ontario M3C 1H7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

® (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. ® Copyright IBM Corp. 1998, 2000. All rights reserved.

Programming Interface Information

Programming interface information is intended to help you create application software using this program.

General-use programming interface allow the customer to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification, and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and Service Marks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

AIX
AS/400
DB2
CICS
C Set ++
IBM
Network Station
Object Connection
OS/2
OS/390
OS/400
Open Class
Operating System/2
Operating System/400
PowerPC 403
PowerPC 601
PowerPC 603
PowerPC 604
Presentation Manager
RS/6000
S/390
SAA
Systems Application Architecture
TeamConnection
VisualAge
WebSphere
Workplace Shell

Lotus, Lotus Notes, and Domino are trademarks or registered trademarks of the Lotus Development Corporation in the United States, or other countries, or both.

Tivoli Management Environment, TME 10, and Tivoli Module Designer are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

Encina and DCE Encina Lightweight Client are trademarks of Transarc Corporation in the United States, or other countries, or both.

Microsoft, Win32, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the U.S. and other countries licensed exclusively through X/Open Company Limited.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, or other countries, or both.

C-bus is a registered trademark of Corollary, Inc.

PC Direct is a registered trademark of Ziff Communicatoins Company and is used by IBM Corporation under license

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.

Industry Standards

VisualAge C++ Professional for AIX, Version 5.0 supports the following standards:

- The C language is consistent with the International Standard C (ANSI/ISO-IEC 9899-1990 [1992]). This standard has officially replaced American National standard for Information Systems-Programming Language C (X3.159-1989) and is technically equivalent to the ANSI C standard. VisualAge C++ supports the changes adopted into the C Standard by ISO/IEC 9899:1990/Amendment 1:1994.
- The IBM Systems Application Architecture (SAA) C Level 2 language definition.
- The C++ language is consistent with the International Standard for Information Systems-Programming Language C++ (ISO/IEC 14882:1998).
- The ISO/IEC 9945-1:1990/IEEE POSIX 1003.-1990 standard.
- The X/Open Common Applications Environment Specifications, System Interfaces and Headers, Issue 4.

About This Book

The information in this PDF document is also available in the online help.

To find this information, or any topics listed in this document as Related Concepts, Related Tasks, or Related References, simply type the topic title into the search bar in the top frame of your browser in the online help.

For some topics, the suggested references may already be contained in this document. In such cases, there is a cross-reference to the page on which the related topic appears.

References to VisualAge or VisualAge C++ in this book should be interpreted as VisualAge C++, Version 5.0.

Chapter 1. IDE Features and Tips

Start the IDE

There are two ways to start the IDE for VisualAge C++:

- Open the IDE using the IDE icon in the VisualAge C++ program group.
- Type the command `vacide` and any parameters you want to specify on a command line. Type `vacide -h` to view help for the command, or see the link below.

AIX **Note:** Your path must include `usr/vacpp/bin` in order for the **vacide** command to be found.

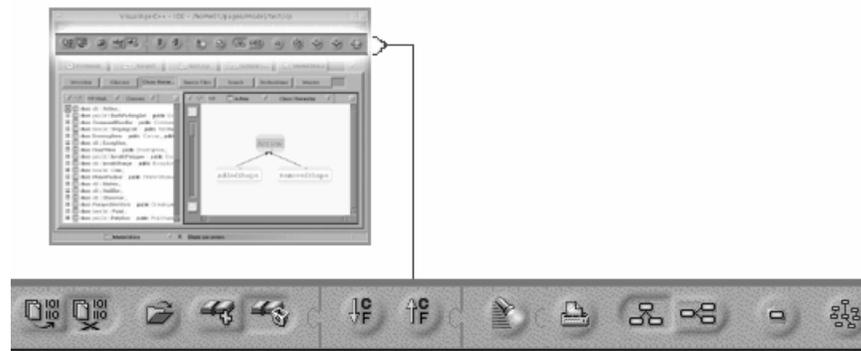
RELATED REFERENCES

`vacide` Command

Workbook

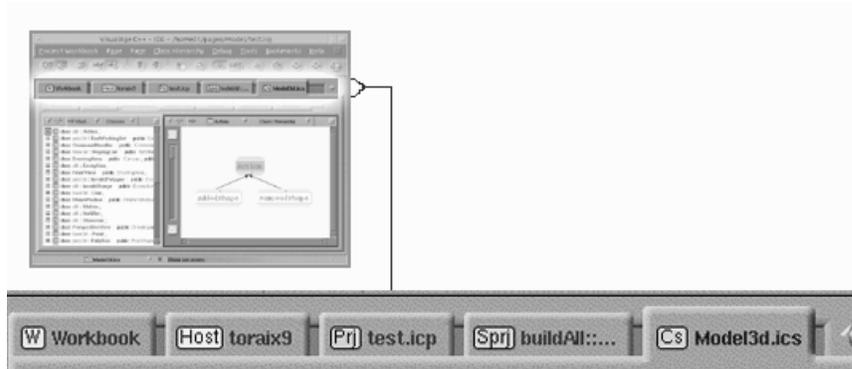
The workbook is the heart of the VisualAge C++ IDE. It has been carefully organized to make the most common coding tasks easier. Where we have not anticipated your needs, you can extend and customize the workbook to suit your particular approach.

The icons are grouped into *toolbars*:



Different toolbars are displayed in the toolbar area, depending on the object you select and the actions that you can perform on the object.

The workbook is divided into sections.



A section organizes information about an object. An object can be as large as the entire project, or as small as a class or a function. The object's name appears on the *tab* for the section.

- In the **Workbook** section, you view and change settings for the IDE.
- In the **Host** section, you view and work with files on your file system and add files to your project.
- In the **Project** section, you define builds and environments for the project at a high level.
- In the **Configuration** and **Codestore** sections, you work with code at the source level, and define specific targets and sources.

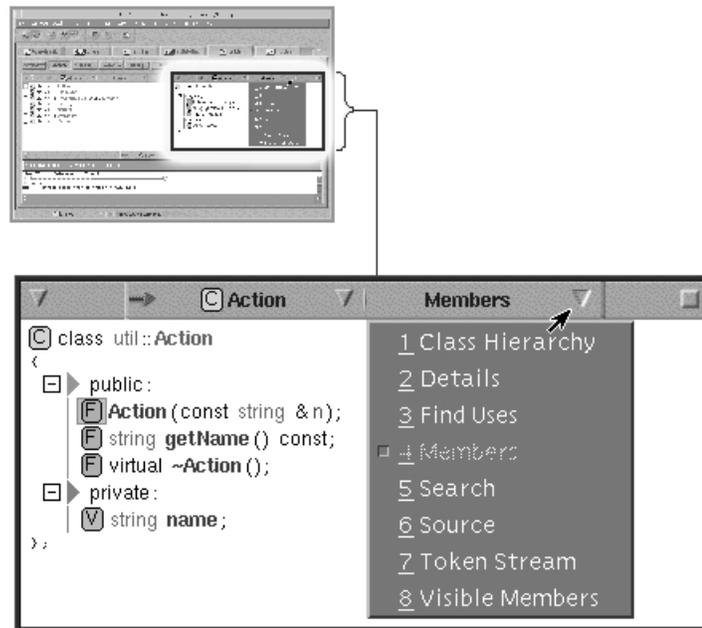
Many other types of sections may also be visible, depending on how you choose to display objects you are working with. You can add your own sections to the IDE at any time.

Each section contains one or more *pages*. A page contains specific information about the object named on the tab, and is organized for a particular task. For example, a codestore section contains pages for working with source files, debugging or running code, finding uses of declarations, searching, and more. The name on the page button indicates the page's task. If one of the default pages is not set up the way you need it, you can create a new page or change an existing one.

Each page contains one or more *panes* which can display several types of information about an object. Each pane contains information about the object named in its title bar. The pane with the highlighted border has *focus*, and only one pane on a page can have focus at a time.

The information displayed in a pane is called a *view*. Each view has a name and a type, so each view can only display information about certain types of objects. The name of the view is on the right side of any pane title bar. The view can be changed to display different information about the object, but the object type

determines which views are available.

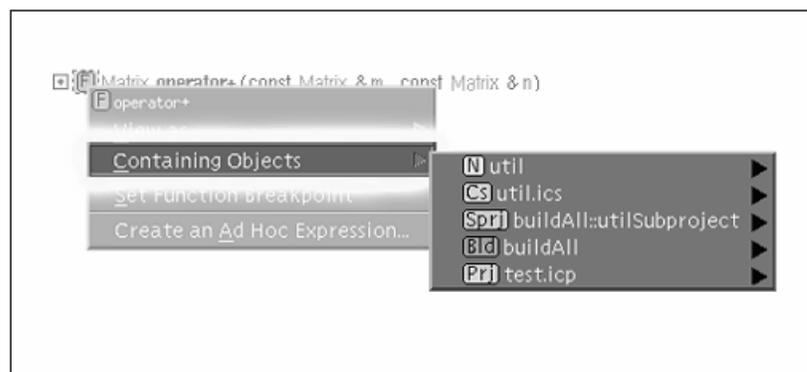


In a view, as you click on objects, they are *selected*. Selected objects are highlighted by a surrounding box with a dark background. When you select an object in a pane, the pane has focus.

Pull-down menus are located in the menu bar and in each pane title bar.

Each object has a *pop-up menu* that is displayed when you use the right mouse button to click on the object. This menu contains actions for the object. To get a quick overview of how this object fits into the project, select **Containing Objects**.

For example, the function selected in the example below () is contained the namespace util, the codestore util.ics, the subproject utilSubproject, the build buildAll and the project test.icp:



The view background also has a pop-up menu.

For more information on setting up the workbook to suit your working style, see the related links below.

RELATED CONCEPTS

Customizing the IDE — Overview
Links Between Panes

RELATED TASKS

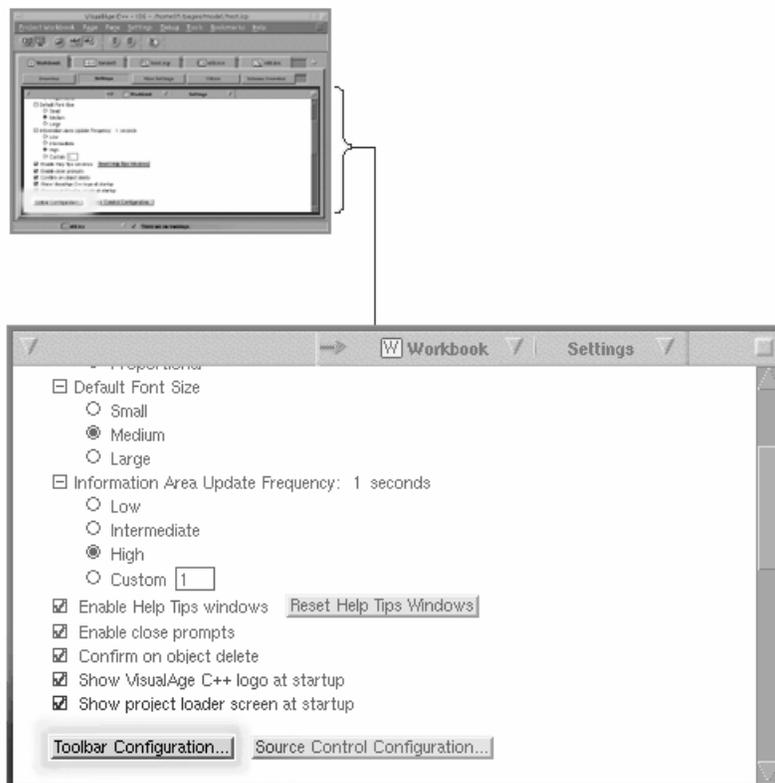
“Improve IDE Workbook Performance” on page 6
Maximize Screen Space in the IDE

Toolbars in the IDE

The icons in the IDE are grouped into four toolbars:

- The **Workbook** toolbar contains the global actions for the workbook, such as Build.
- The **Pane** toolbar contains actions to work with a pane, such as Detach Link.
- The **View** toolbar contains actions to work with a view. For example, when a Source view has focus, the Save action is available.
- The **Process** toolbar contains the actions to work with a process that is being debugged, for example, the Run and Stop actions.

Click **Toolbar Configuration** on the Settings page of the Workbook section to customize each of these toolbars:



You can change the buttons displayed, the style of the buttons, or the location of the toolbars within the IDE.

If a position indicator, a small window with buttons indicating the toolbar layout, appears as you drag a toolbar, then the toolbar is positioned as shown in the indicator. If you drop a toolbar without a position indicator displayed, the toolbar floats in its own window. Position indicators appear when the mouse pointer is over one of the workbook borders.

Toolbars can be positioned separately or put together. To put two toolbars together, you must drag the toolbar by the end that will fit into the other toolbar. Drop the toolbar when the positioning indicator has a single row of buttons.

RELATED TASKS

Customize Toolbars

Navigating the IDE

The mouse is the most direct tool for navigating through the IDE. Display pages by clicking the page buttons. Click a tab to move to a different workbook section.

Tab and Page Scroll Controls

A scroll control   is found immediately to the right of the page buttons or the tabs in the workbook. Click either of the two arrows to scroll to a page button or a tab that is not visible.

To display the page last viewed before the current one, click the previous page control  .

Click the Find Current control  to move the tab and page button for the current page back into view if they are not visible.

Bookmarks

The IDE uses bookmarks in the same way a Web browser does. You can set bookmarks on pages to return quickly to chosen locations. Add bookmarks and navigate to bookmarked pages from the **Bookmarks** menu.

Overview Page

The **Table of Contents** view on the Overview page in the Workbook section lists all the current sections and pages for the workbook. Double-click any page or section to display it.

Project Workbook Menu

The **Project Workbook** menu lists the Workbook, Host and Project objects. You can move to these sections by selecting an object from this menu. This is especially convenient if you removed a section for one of these objects earlier, and you want to recreate it.

The Project Workbook menu also features a submenu called **Actions for Recently Used Objects**, which lists the most recent objects selected in the IDE, and the actions you can perform on those objects.

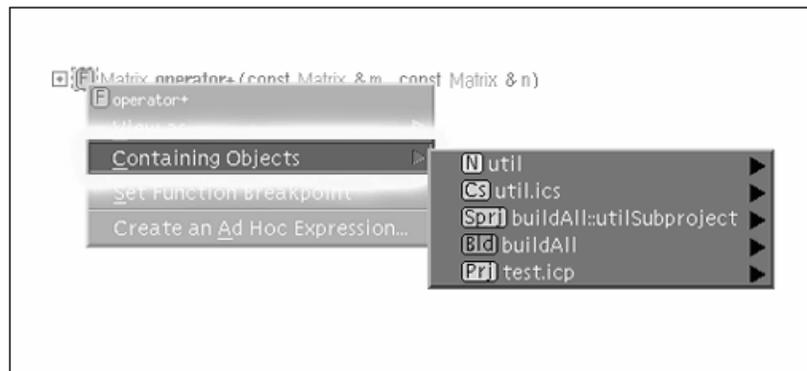
Page Menu

The **Page** menu on the IDE's menu bar contains items for all of the sections

currently in the workbook. Each section has a cascaded menu that lists all of the pages in that section. Select a page to display it. This menu is especially useful when you use the maximize control  to expand the workbook area by hiding the page buttons and tabs.

Quick Context Finder

To see how any object fits into the overall project, right-click on the object and select **Containing Objects** from the pop-up menu. The resulting list will show you, in ascending order, the objects that contain the object you selected, ending with the project.



In the example above, the object selected is a function. The Containing Objects list shows which class contains the function, which namespace contains the class, and so on, up to the level of the project.

RELATED CONCEPTS

“Workbook” on page 1

RELATED TASKS

Add or Delete a Bookmark

Improve IDE Workbook Performance

You can improve workbook performance by:

- Discarding views that are not displayed
- Minimizing the number of times the codestore is saved to disk
- Turning off animation on pane title bars (see the Related Task below for more information)
- Closing unused objects in memory

Discard Unused Views

The number of views maintained has a direct effect on the time it takes to refresh. You can reduce this time by discarding undisplayed views.

To discard all views in the workbook other than the ones currently displayed, select **Flush Workbook** from the **Project Workbook** menu. For more information, see the Related Tasks below.

Save the codestore less frequently

It takes time to save the codestore to disk. To improve performance, you can limit how often the codestore is saved using the Settings view in the Workbook section:

- Set the number of builds to occur before automatically saving the codestore to a higher number. The default is 5 builds.
- Select **Save successful builds only** so only successful builds are saved to the persistent codestore.

Regardless of these settings, the codestore is always saved to disk after the initial build of a project, and again when the IDE is closed. If for some reason, such as a system failure, the codestore is not saved to disk before the transient (temporary) codestore is erased, your next build will be based on the most recently saved codestore.

Close Unused Objects in Memory

For information on closing a subproject, see the Related Task below.

Each time you select an object to view its source, the file containing the object is opened in memory. When you close a section of the IDE, however, the files displayed in that section are not necessarily closed in memory. To clear additional memory, you can close any files you are no longer using by selecting **Flush Workbook** from the **Project Workbook** menu. All files except those with unsaved changes will be closed.

RELATED CONCEPTS

Customizing the IDE Workbook

“Managing Memory Use in the IDE” on page 9

“Workbook” on page 1

RELATED TASKS

“Open or Close a Subproject in Memory”

Remove All Unused Pages in the IDE

Set Pane Title Bar Animation

Open or Close a Subproject in Memory

In order to minimize memory use, subprojects are not automatically opened in memory when you open a project. Even if you open a workbook section for a subproject object, the subproject is not necessarily available in memory.

Open and close subprojects from the Subprojects view on the Overview page, in the Project workbook section.

Opening a Subproject

Select the subproject object () and click the **Open** button at the bottom of the view.

Closing a Subproject

Once you have opened a subproject in memory, it remains in memory until you close it, even if you have removed all the workbook sections related to it. To close a subproject, select the subproject object in this view, and click the **Close** button at

the bottom of the page. When you close a subproject, the workbook sections associated with it (the Codestore section and Configuration section) will also be closed.

Determining Whether a Subproject is Open or Closed

If a subproject is closed in memory, the status line under a subproject summary says Closed. If Built, Building, Built with errors, or Built with messages appears in the status line, the subproject is open, even if no workbook sections are currently open to display it. When a subproject is open in memory, you will also notice that graphic objects ( , ) will appear to represent the configuration file and codestore in the summary.

Example of a Closed Subproject

```
[-]  Label: baseSubproject
  Configuration: /home01/pageee/model/mod2/base/base.icc
  Codestore: /home01/pageee/model/mod2/base/base.ics
  Environment:  buildAll::baseSubproject
  Status: Closed
```

Example of an Open Subproject

```
[-]  buildAll
  [-]  Label: utilSubproject
    Configuration:  /home01/pageee/model/mod2/util/util.icc
    Codestore:  /home01/pageee/model/mod2/util/util.ics
    Environment:  buildAll::utilSubproject
    Status: Built
```

RELATED TASKS

Close an Object in Memory in the IDE
“Improve IDE Workbook Performance” on page 6
Maximize Screen Space in the IDE

Change Font Size in the IDE

To change the size of the fonts used in pane title bars, section tabs, and the information area, go to the Workbook section.

Select the Settings page, and choose from **Small**, **Medium**, or **Large** settings in the Default Font Settings section, under Workbook Settings.

The new settings do not take effect until you restart the IDE.

RELATED TASKS

Maximize Screen Space in the IDE

Hide Pane Title Bars in the IDE

To increase available screen space, you can turn off pane title bars:

1. Go to the Settings page in the Workbook section.
2. Under the heading **Pane Options**, check the **Auto Hide Pane Title Bars** box.

Pane title bars will still appear if you place the mouse pointer near the pane's top border, but will disappear when you move the pointer away from the title bar.

RELATED CONCEPTS

Customizing Panes in the IDE

Maximize Screen Space in the IDE

Managing Memory Use in the IDE

By default, when you open a project, the subprojects in the project are not opened. You must explicitly open them on the Project Overview page in order to load them in memory. Unless they are opened, you cannot view the contents of any of the configuration files (.icc), or your source files. However, once you have opened them, they remain in memory even if you are not working with them, until you explicitly close them.

Unless you need to have more than one codestore open (for example, for cross-codestore debugging), you should close the subprojects you are not using, to reduce memory use:

1. Go to the Project section.
2. Select the Overview page.
3. Highlight the Subproject  you want to close.
4. Click **Close** at the bottom of the view. The status shown for the subproject should change to **Closed**. Any workbook sections associated with the selected subproject will also be closed.

Example of a Closed Subproject

```
☐  Label: baseSubproject
Configuration: /home01/pageee/model/mod2/base/base.icc
Codestore: /home01/pageee/model/mod2/base/base.ics
Environment:  buildAll::baseSubproject
Status: Closed
```

RELATED TASKS

“Improve IDE Workbook Performance” on page 6
Maximize Screen Space in the IDE

Chapter 2. Working with Projects in the IDE

Create a Project

A project in VisualAge C++ Version 5.0 is defined by a project file (.icp file). You do not necessarily need to have a project file in order to work with your code. However, no source file can be considered part of a build or part of a project unless it is defined in a subproject. You should consider having a project file if you intend to work with more than one subproject at a time, especially if the subprojects have dependencies on one another.

If you open a subproject alone, by selecting a .icc file or .ics file, a project section is created for you in the workbook. However, no project file is created.

To create an .icp file, select **Create Project** from the **Project Workbook** menu. This opens the Project SmartGuide, which creates a project file using your input.

You can create a new, empty project, or create a new project using existing configuration files and source files. The SmartGuides lead you through the process. You do not have to close your current subproject, if one is open, in order to create a project that contains the subproject.

RELATED CONCEPTS

Projects
Configuration Files
Project Files

RELATED TASKS

Open or Close a Project

View a Build Setup for a Project

To see all the subprojects currently open or in use in a build, use the Overview page in the Project section of the workbook (). This view does not show any subprojects that are defined in the project file but not used in a build.

To see more detail about each build in a project, use the Project File page.

The two panes on the left of the Project File page show two different views of the entire project. Both can be changed to give a different perspective on the project. For information on changing views, see Change the View Type in a Pane in the Related Tasks below.

By default, the upper pane shows a Builds view. This is a list of all builds defined in the project file. The lower pane shows a Details view. This is similar to a Source view, but it is a parsed view that displays all the directives in the project file as objects you can manipulate or make changes to. Directives are highlighted in this view.

Each time you select a directive, the linked pane on the right side of the window presents a view in which you can make changes to the object. For example, if you

select a subproject in the Details view, the Change Subproject view appears to the right, allowing you to change any aspect of a subproject, including its dependencies and variables.

RELATED CONCEPTS

Configuration Files
How Configuration Files Are Processed

RELATED TASKS

“Change a Build Setup” on page 15
Change the View Type in a Pane

RELATED REFERENCES

Configuration File Syntax

Add a Subproject to a Project

In order for a subproject to be considered a part of a project, it must be defined in the project file (.icp file). To add a subproject to a project, you can:

- add the definition to the project file directly
- add it using the IDE
- use the Project SmartGuide to create a new project and add the subprojects for you

Defining a Subproject in a Project File

There are two requirements for defining a subproject and ensuring it is considered part of a build in a project file:

1. The definition:
A subproject is defined with the subproject directive, in the following form:

```
subproject Subprojectname icc "configname.icc"[, ics  
"codestorename.ics"]  
{  
}
```

The subproject must be defined at global scope. That is, the definition cannot appear within the { and } of another object. The codestore specification is optional.
2. The use directive:
Once the subproject is defined, it must be included in a build with the use directive. The use directive appears between the { and } of a build directive, and takes the following form:

```
use Subprojectname
```

To create a configuration file (.icc file), see the Related Task below.

Adding a Subproject to a Project in the IDE

1. In the **Project** section, select the **Project File** page.
2. In either the Builds view or the Details view, right-click on the object after which you want to add the subproject. To add the subproject at the top of the project file, right-click in the background of one of the views, instead of selecting an object.
3. From the pop-up menu, select **Add** or **Add directive after**, and **Subproject**.

4. In the **Add Subproject** dialog, supply the name of the subproject and the name of the configuration file. Optionally, you can specify the name of the codestore file, which builds to use the subproject in, dependencies on other subprojects, and any variables to use when building the subproject.

Note: the subproject must be defined before it can be used in a build.

To create a configuration file (.icc file), see the Related Task below.

Adding a Subproject to a New Project Using the SmartGuides

1. Select **Create Project** from the **Project Workbook** menu.
2. From the Project page in the Project SmartGuide, choose either **Create a new project using new and existing configuration files** or **Advanced Setup**.
3. The next page is the Configurations page. Select configuration files (.icc files) here, and add them to the project by clicking the **Add>>** button. To create new configurations, click **Create**. This will launch the Configuration SmartGuide.
 - You do not have to supply a name for the corresponding codestore files (.ics files) if you chose the second radio button. By default, the codestore files will have the same name as the configuration files.
 - If you chose the Advanced Setup, you will further define the details of the subprojects in later pages of the SmartGuide. For example, you can associate the configuration files with specific codestore files, and you can set up builds and build variables.

RELATED REFERENCES

Project File Directives

RELATED TASKS

“Create a Configuration File” on page 14

Set up Dependencies Between Subprojects

depends on directives connect subprojects to other subprojects.

There are several reasons why one subproject may have a dependency on another. For example, one subproject may link with a library that is produced by another subproject.

Dependency relationships between subprojects are defined by *depends on* directives in the project file. For example, the following sample subproject directive defines a subproject *animals*, which has a dependency on the subproject *zoo* (that is, the target of *zoo* must exist before the target of *animals* can be built):

```
subproject zoo icc "zoo.icc" ics "zoo.ics"
subproject animals icc "animals.icc" ics "animals.ics"
{
  depends on zoo
}
```

Setting up Dependencies Between Subprojects in the IDE

1. Go to the Project File page in the Project section of the workbook.
2. From the Builds view or Details view, click on the subproject that depends on another subproject. The view on the right will then become a Change Subproject view.
3. Expand the **Dependencies** interactor (the plus symbol) and click on **Depends on subproject**.

4. Find the required subproject, select it and click **Add**.
5. Click **Apply**.

To see the results of your changes, refresh the Project File views.

Setting up Dependencies Between Subprojects in the Project File

1. In a Source view of the project file, find the definition of the subproject which has a dependency on another subproject.
2. In the directive block (between the { and } of the definition), insert depends on followed by the name of the subproject on which this subproject depends.

To see the results of your changes, refresh the Project File views.

RELATED CONCEPTS

Project Files

RELATED REFERENCES

Project File Directives

Set Build Options for a Subproject

Set build options that apply to a subproject using the Subproject Options page in the Configuration section.

1. In the Subproject Options view, set the values for options that you want to apply to the project. See the Related Concept below for more detail on option states.
2. If you decide you need to start over while you are setting options, click **Reset** to restore the options to the values in the configuration file.
3. When you have finished setting options, click **Apply** to save the changes to the configuration file. An option group directive with the name SubprojectOptions is added to the top of the configuration file if it does not exist already. If the SubprojectOptions option directive exists, the options are added to or changed in that directive. SubprojectOptions contains the options you set and applies to the entire configuration file. If you add source files or targets outside the SubprojectOptions options group directive, the subproject options are not applied to those files.

RELATED CONCEPTS

Setting Build Options

RELATED TASKS

“Set Build Options for Source Files or Targets” on page 21
Change an Option Directive

RELATED REFERENCES

Build Options
Configuration File Syntax

Create a Configuration File

When you create a configuration, you create an independent subproject. To create a configuration file (.icc file), follow these steps:

1. From the **Project Workbook** menu, select **Create configuration**. This launches the Configuration SmartGuide.
2. On the **Configuration** page of the SmartGuide, type a name and location for the configuration file. Click **Next>>**.
3. On the **Target Type** page, select the type of object you want this configuration to produce (for example, an executable or shared library) and click the **Add Target** button. This launches the Target SmartGuide.
4. On the Target Name page, select a file name and location for the target. You can also choose a group of options to suit the type of target (for example, select IOC from the Program Type table to automatically add options suitable for using IBM Open Class™ in your application, or select the Debug radio button to add options for debugging).
5. Click **Next**.
6. On the Source Files page, select the file or files you want to use as sources, and click **Add>>**. You can also set the file types.
7. Click **Next**.
8. On the Classes page, you can define classes for the target. If you do not want to define classes, click **OK**.
9. You are returned to the Target Type page, where you can add another target to the same configuration, or finish this configuration.
 - To add another target, click **Add another target**. Repeat steps 4 to 8 above.
 - To complete the configuration, click **Finish**. You will be prompted to open the subproject in the IDE and build it, if the codestore file does not already exist.

To add the new independent subproject (configuration) to a project, the project must be open in the IDE.

1. Select the independent subproject in the Overview page of the Project section.
2. Right-click on the  object, and select **Add to Project File**.

RELATED CONCEPTS

Projects and Subprojects

Change a Build Setup

The build setup for your project is defined at the highest level by a project file. The project file lists subprojects, and controls the build order for all the subprojects. Within a subproject, the configuration file is used to control the building of source files and targets, and to set options.

The IDE offers two ways to change the build setup at either level:

- Visually, using a set of views in the Project and Configuration sections of the IDE to manipulate the objects you want to change, such as source files, groups, options or targets
- Directly editing, using a Source view of the project file or configuration file, or using any text editor

Manipulating Objects in Views to Change a Build Setup

You can add subproject directives, use directives, and variables to the project file using the views in the Project section of the workbook, on the Project File page. The most commonly used actions are grouped in the pop-up menus for individual objects. Select an object and right-click to see the actions available.

Editing the Project File or Configuration File to Change a Build Setup

You can also change the project file (which controls the entire project) or configuration file (which controls a subproject) by editing them directly in a Source view. Other views are not updated until the next build is performed, or until you refresh the views using the pop-up menu on the Project or Configuration workbook tab.

RELATED CONCEPTS

Configuration Files
Setting Build Options

RELATED TASKS

“Add a Subproject to a Project” on page 12
“Add or Remove a Target in the IDE” on page 17
“Add Source Files to a Project” on page 19

RELATED REFERENCES

Build Options
Configuration File Syntax
Project File Syntax

View a Build Setup for a Subproject

The Configuration section () is where you view the configuration file that describes which source files are used and what output is produced when the subproject is built.

Checking the Expected Outputs of a Subproject

The Targets view on the Targets page of the Configuration section shows all the targets in the subproject. Each target object () can be expanded to see which sources are used to produce the target.

Checking the Sources Used to Build a Subproject

The **Source and Groups** page in the Configuration section shows all the source directives () used in the subproject, including directives that define a group of sources (). The list shown in this view represents all the source files that are listed within the { and } of all the target directives in this configuration.

Checking the Options Set for a Subproject

Use the Options Groups page in the Configuration section to see options that have been explicitly set for the subproject. Options with default values that have not been changed are not displayed here.

Locating Sources, Targets, or Options Quickly in the Configuration File

The Source and Groups page and the Targets page both have Source views. When you select a directive or source specifier object, the Source view updates to display the location of the object’s definition. You can edit it directly here.

To view the source of the specifier (that is, to edit the contents of a source file that is listed in the configuration file):

1. In the Source and Groups view, locate the source file you want to see and select its Source Specifier ().
2. In the Source view below, click on the down arrow that is at the far right of the view, at the end of the filename text field.
3. A drop-down list is displayed. Select the file name of your source file from the list.
4. The Source view now displays the contents of the source file you chose.

RELATED CONCEPTS

Configuration Files
How Configuration Files Are Processed

RELATED TASKS

“Change a Build Setup” on page 15

RELATED REFERENCES

Configuration File Syntax

Add or Remove a Target in the IDE

A target is the output file of a build. An executable or shared library is an example of a target.

A target is defined by a target directive in a configuration file, and is represented in the IDE by a  . A target directive usually contains one or more source or group directives that specify the source files used to build the target.

Adding a target to a configuration

1. Select the Targets page in the Configuration section.
2. Right-click in the background of the Targets view. Select **Add directive** from the pop-up menu.
3. From the cascading menu, select **Target**. (You can also select **Target through SmartGuides** to launch the Target SmartGuide.)
4. In the **Add target** dialog, type a name for the target.
 - If you do not give the file name an extension, the name you type is assumed to be a variable, unless you enclose it in quotation marks (“ ”).
 - If the extension you give does not indicate the type of file you want to build (for example, if you want the target to be a static library, but do not add `.a` to the file name), you should select a target type from the pull-down **Type** list. This also provides more information to the Targets view later, and may make it easier to distinguish targets in a large project.
5. Click **Apply**.

Removing a target from a configuration

1. Select the Targets page in the Configuration section.
2. Select a target object () from the Targets view.

3. Right-click on the target directive, and select **Remove** from the pop-up menu. The selected target directive is deleted from the configuration file, and the views in the Configuration section are refreshed. If source and group directives are defined in the target, they are also removed from the configuration file.

Targets can also be removed through any view in the Configuration section that displays the target object. To change or move the source files used to build the target, see the links below.

In the IDE, the Targets view on the Targets page lists all of the target directives in a configuration file.

RELATED CONCEPTS

Configuration Files
Targets

RELATED TASKS

Change a Target Name or Type
Change a Target's Source Files
"Move a Source File within a Subproject"
"Change a Build Setup" on page 15
View All Targets in a Configuration

RELATED REFERENCES

target Configuration File Directive

Move a Source File within a Subproject

To move a source file from one source directive to another, or to a new source directive in a target directive:

1. In the **Configuration** section, select the **Source and Groups** page.
2. In the Source and Groups view on this page, select the filename (the *source specifier*, ) that you want to move.
3. Right-click on the file. A menu is displayed.
4. Select **Move to Source** to move the file to a source or group directive, or select **Move to Target** to move the file to a new source directive in a target directive. **Move to Target** will only appear if the source specifier is not already listed in a target directive.
5. The **Move to Source** or the **Move to Target** dialog box is displayed.
6. In the dialog box, in the **Source** or **Target** list, select the directive to which you want to move this file.
7. To undo the selections you have made, click **Restore**. Click **Cancel** to close the dialog box without moving the source file.
8. Click **Apply** to move the source file, and to save the change to the configuration file. The other views in the Configuration section are refreshed. To update the views in other sections, you must build the project.

RELATED TASKS

Change a Target's Source Files
Search a View Using Live Find

Add Source Files to a Project

To add existing or new source files to your project, you must add them to a configuration.

A source file is added to the configuration when a source or group directive is added to the configuration file (.icc file), and a source specifier names the source file within this directive.

A source directive defines one or more source files that are used in a build. A group directive names a group of source files. A group name is used like a variable in source and target directives to represent a set of source files. For a source file to be part of a build, it must either be specified in a source directive, or be in a group that is specified in a source directive.

Source files can be added to a project or removed from a project in two ways:

- By adding source directives to a configuration file directly
- By using views in the IDE (page 19)

Adding Source Directives to a Configuration

Source directives usually appear within a target directive. Although any source file listed in a configuration file will be compiled, source files listed outside of a target directive will not be built into any target (executable or library file).

To add a source directive, add a line like the following to your configuration file, between the { and } braces of a target directive or another source directive:

```
source filename
```

For example, the following target directive contains a directive to include the source file `example.cpp` when the target `app.exe` is built:

```
target type(exe) "app"  
{  
    source "example.cpp"  
}
```

Adding Existing Source Files Through the IDE

To add existing source files to your project, use the Targets page or the Source and Groups page in the Codestore section.

1. First, you must decide which directive you want to add the source file to.

Locate that directive (, , or ) in one of the views in the pages listed above. If you do not intend to add it to a directive, you can add it to the top of the configuration file (the source will be listed in the configuration, but not built).

2. Right-click on the directive, and select **Add directive into** or **Add directive after**, or **Add Source Specifier** from the pop-up menu. If you want to add the source to the top of the configuration file, right-click on the view background instead, and select **Add directive**.
3. From the cascading menu (if there is one), select **Source**.
4. In the dialog box (**Add Source**, **Add Source after**, **Add Source Specifier** or **Add Source into**), type the name of the source file in the **Source** field, or locate

it in your file system by clicking the **Browse** button. The file extension is used to determine the file type. You do not have to make any changes to the file type unless you want to override the type implied by the extension, or unless the file has no extension.

5. Click the **Add** button (next to the **Browse** button) after entering each file name, even if you are only adding one source file.
6. Click **Apply** to finish adding sources and exit the dialog.

Adding New Files

1. Select **Open or Create File** from the **Project Workbook** menu.
2. In the **Open or Create File** dialog, type a name for the file. If any subprojects are open in memory, you will also have the option of adding the file to the configuration for a subproject. To do this, check the box labelled **Add to subproject**.
3. Click **Open**.

RELATED CONCEPTS

Configuration Files
Sources

RELATED TASKS

Group Source Files in a Configuration
"Move a Source File within a Subproject" on page 18
"Remove Source Files from a Project"

RELATED REFERENCES

source Configuration File Directive

Remove Source Files from a Project

To remove source files from your project, you must remove them from the configuration in which they are used.

In any Configuration section view where the file is listed as a graphical object (), such as the Source and Groups view on the Source and Groups page, do the following to remove a source file:

1. Select the source file, and right-click over it.
2. Select **Remove** from the pop-up menu. The source file is deleted from the configuration file.

The views in the Configuration section are refreshed immediately, but views in the Codestore section are not updated until you rebuild the project. The source file remains in your file system.

To remove more than one source file at once, hold down the **Ctrl** button, and click on each of the source files you want to remove. When you select **Remove** from the pop-up menu, all the files you have highlighted will be removed from the configuration.

RELATED CONCEPTS

"Workbook" on page 1
Configuration Files

RELATED TASKS

“Add Source Files to a Project” on page 19
Search an IDE View Using Live Find

Set Build Options for Source Files or Targets

Options can be applied to individual source files and targets within a configuration.

1. In the Configuration () section, select the Options page.
2. On the Options page, in the Sources and Targets view, select a source file or target.
3. In the Change Options view, set the options for that source file or target.
4. If you need to start over while you are setting options, click **Reset** to restore the values in the configuration file.
5. Click **Apply** to save the changes to the configuration file. An option directive containing the build options you set is added to the configuration file. The option directive will apply only to the source file or target you selected.

RELATED CONCEPTS

How Configuration Files Are Processed
Setting Build Options

RELATED TASKS

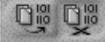
“Set Build Options for a Subproject” on page 14
Change an Option Directive

RELATED REFERENCES

Build Options
Configuration File Syntax

Build

There are several ways to start a build in the IDE:

- Click the Build button  on the Workbook toolbar.
- Select **Build** from the **Project Workbook** menu, and select one of the objects from the cascading menu.
- Press **Ctrl+Shift+B** to build the object currently identified in the information area at the bottom of the IDE window.
- Click the **Build** button in the Overview page of the Project section to build the object that is currently selected in the view.

The differences between building a complete project, building a subproject, and building only a codestore are described in *Incremental C++ Build*, in the Related Concepts below.

The information area at the bottom of the IDE window displays the object that is being built:



You can change the object to be built with the selector arrow next to the object name. If you are not sure which object will be built, hold the mouse pointer over the Build toolbar button. The hover help shows the name of the object.

All of your source file changes are saved automatically before the build starts. It does not matter which page is displayed when you start the build.

For performance reasons, the codestore is not saved to disk on every build by default. You can adjust how often the codestore is automatically saved on the Settings page in the Workbook section.

Progress messages and the results of your build are displayed in the information area at the bottom of the IDE window. Build messages are displayed in a Messages view.

Rebuilding After Moving Source Files

If you attempt to rebuild a subproject after deleting a source file or included source file (e.g. a .cpp or .hpp file) from its original path, the build will fail. The build also fails if you move a source file or included source file from its original location to a path not searched elsewhere in the project. If you move a source file from its original path to a path searched for any other source file in the project, the file will be found and the subproject will build successfully.

RELATED CONCEPTS

Configuration Files
Codestore
Incremental C++ Build

RELATED TASKS

View Build Messages
“Control Build Messages” on page 24
Build from the Command Line

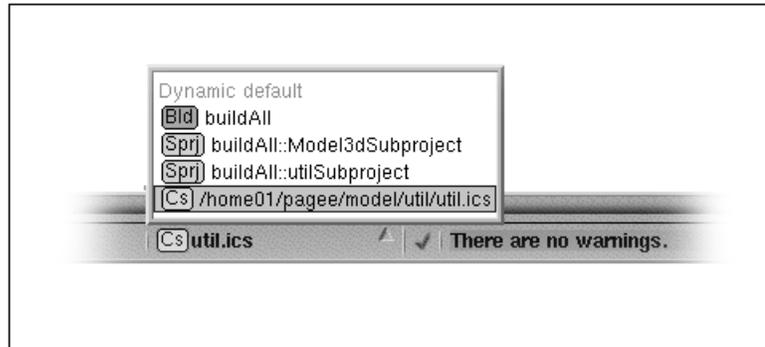
RELATED REFERENCES

“Dynamic Default Build Setting”

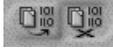
Dynamic Default Build Setting

The information area at the bottom of the IDE window tells you which object is being built, or will be built. It also allows you to select a different object for a build.

If you prefer to automatically select any object you have currently selected in the active pane, you can set the information area to *dynamic default*. Simply select `Dynamic default` from the pull-up menu in the information area.



When dynamic default is selected, the object selected in the Overview page will be

built the next time you click the Build toolbar button () or the Build button in the pane. When dynamic default is selected, the last object selected in the view currently in focus will be built. For example, if you selected a class object, then the codestore containing that class will be built.

RELATED TASKS

“Build” on page 21

View a Subproject After Processing

After a subproject has been built, you can verify how conditional processing was done, or check how variables were evaluated.

To do this, use the Interpreted Configuration view on the Advanced page in the Configuration section.

This page has two default views: the Details view on the left, and the Interpreted Configuration view on the right.

The Details view is a parsed view of the configuration file, highlighting all directives and allowing you to collapse and expand each directive that contains nested directives, so that you can view as little or as much detail as you want.

On the right side of the window, the Interpreted Configuration view resembles the Details view, but instead of showing all the directives as they appear in the configuration file, it shows how each directive was evaluated. If, for example, you used an if directive in the configuration file, which was evaluated as “False” in the build, this directive and its block are greyed out in the Interpreted Configuration view.

You can quickly see how changes you make to the configuration will be handled, without starting a build. After making your changes, click **Interpret** in the Interpreted Configuration view.

RELATED TASKS

View Preprocessed Source

“View a Build Setup for a Project” on page 11

“View a Build Setup for a Subproject” on page 16

Control Build Messages

You control the output of build messages by specifying build options in the configuration file. The IDE provides actions to disable a message or to treat an information or warning message as an error message. These actions add the appropriate build options to the configuration file. In a Messages view, right-click on a message object to display the pop-up menu containing these actions.

To disable a message, select **Disable Message** from the pop-up menu of the message. The report message option is added to the configuration file as a project option.

To treat an information or warning message as an error message, select **Treat Message as Error** from the pop-up menu of the message. The report treataserror option is added to the configuration file as a project option.

To reset the output behavior of a build message, use the Subproject Options page in the Configuration section. This page also contains other options for controlling build messages. If you want to control build messages for a specific file, use the Change Options view on the Options page.

RELATED CONCEPTS

Messages Produced by the Build Process
Setting Build Options

RELATED TASKS

View Build Messages

RELATED REFERENCES

Miscellaneous Build Options

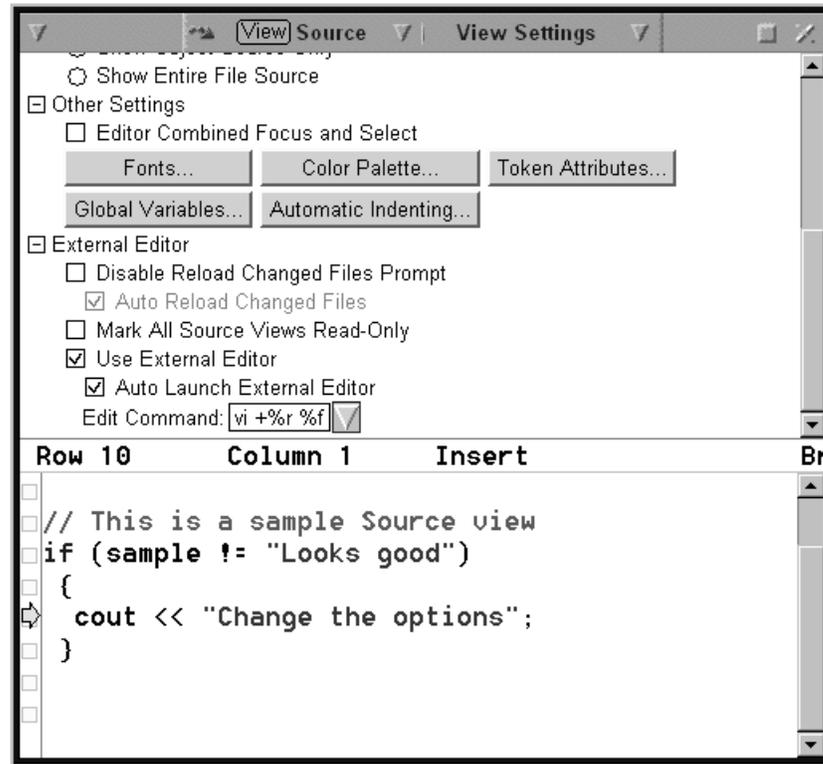
Use an External Editor with the IDE

If you prefer to work with a standalone editor, rather than using the Source views in the IDE, you can set the IDE to update its views from your editor.

Select any **Source** view in the IDE.

Right-click on the pane title bar, on the view name. Select **View Settings** from the pop-up menu (near the bottom).

In the upper half of the View Settings view, scroll down to the the **External Editor** setting (at the bottom of the list).



Check the **Use External Editor** checkbox. Optionally, you can also choose to launch the external editor whenever a page containing a Source view is displayed or a Source view is updated.

In the **Edit Command** field, you can either choose one of the two preset external editor commands, or enter your own. If you choose one of the preset commands, you can also change the parameters, if you want.

For example, the command for launching vi in the illustration above is:

```
vi +%r %f
```

This launches vi and positions it to the row number indicated in the current Source view (%r), in the same file (%f) as that shown in the Source view. You can edit these parameters.

RELATED CONCEPTS

Editing in the IDE

RELATED TASKS

Change the Behavior of the Default Editor Used in the IDE

Contacting IBM

We're delighted to offer a solid cornerstone for your application development: IBM's comprehensive support services. Whether you are an occasional user with a simple question, a power user with many complex technical questions, or someone who requires application design assistance and consulting, IBM Support can meet your needs.

Comments on This Help

Please let us know about any errors or omissions in this online help or in the hardcopy Getting Started book, or our PDF documents. Send your e-mail to: torrcf@ca.ibm.com

Fee Support

Developers on the VisualAge C++ for AIX Service and Support team handle everything from simple how-to's to complex technical problems. Solutions may take the form of a brief explanation, a workaround, a fix to the current product, or a fix to the next release.

<http://www.ibm.com/support/> describes IBM Support Offerings on all platforms, worldwide.

<http://www.ibm.com/rs6000/support/> describes support offerings on the RS/6000® platform, in your country. It also indicates whether your country provides support electronically over the Internet in addition to telephone support.

<http://www.lotus.com/passport> describes the IBM and Lotus® Passport Advantage™ contracting option.

The IBM Software Support Handbook, accessible from <http://www.ibm.com/software/support>, also lists worldwide support contacts.

Phone numbers for information on Support Line offerings are:

- **United States:** 1-888-426-4343 (IBM Global Services), option 3 as of December 1999. Should this number change, IBM general information at 1-800-IBM-4YOU (1-800-426-4968) can route you to the appropriate group.
- **Canada:** 1-800-465-9600, option 3 as of December 1999. Should this number change, you can also contact IBM general information at 1-800-IBM-4YOU (1-800-426-4968).

Please call 1-800-266-8720 in the U.S. and Canada for information on Passport Advantage offerings.

- Elsewhere, please contact your local IBM office.

If you contact support, please have the following information available:

- The product name
- The product version
- The hardware configuration and software (product names and versions you are using)
- What happened and what you were doing when the problem occurred

- Whether you tried to solve the problem and how
- The exact wording of any messages displayed

Consulting Services

VisualAge and WebSphere™ Product Affinity Services Group is a core group of technical specialists from the IBM development labs that created the IBM VisualAge and WebSphere products. With access to a network of IBM product area experts, IBM and industry business partners, and some of the best resources in the industry, we can put the optimal team in place to meet the challenge of absorbing new technology. Our goal is to enable organizational success with VisualAge and WebSphere — ensuring that our products are used effectively within your development team.

For more information, visit <http://www.ibm.com/software/ad/vaws-services/> or contact the Product Affinity Services Team at:

AIM_SERVICES@us.ibm.com