# IBM Rational Rhapsody and Visual Studio 2008

*Workflow Integration*

Ashirbad Dash

August 26, 2010

# Introduction

Models and code have been used side by side for quite some time now. The wide spread use of UML (Unified Modeling Language) and its advantages has led to the creation and use of a number of UML tools that provide code generation capabilities.

While most of the UML tools provide efficient code management mechanisms, these mechanisms are mostly centered on one aspect of an IDE (Integrated Development Environment), i.e. source code editor. IBM Rational Rhapsody also provides a rich set of features for source code management; however, most UML tools (including Rhapsody) lack or do not intend to support other important functionalities of an IDE like (compiler/interpreter, build automation tools, debugger, etc…).

Hence, an efficient integration between a UML tool and an IDE is an essential aspect of efficient application development. This whitepaper discusses one such integration between two such widely used tools in their respective domains; IBM Rational Rhapsody and Microsoft Visual Studio 2008.

The white paper intends to provide you with a comprehensive guide to successfully using the integration between IBM Rational Rhapsody and Microsoft Visual Studio 2008.

While the white paper illustrates and discusses certain features of both Rational Rhapsody and Visual Studio 2008, it does not intend to go into the specifics of Rational Rhapsody or Visual Studio 2008.

By the end you will be able to setup and use the integration and its various features that will enable you to be more efficient in the way code and model is handled together.

# Overview of the Integration

From version 7.5, Rational Rhapsody provides a tighter integration with Microsoft Visual Studio 2008 simplifying and automating the integration between the two.

A Rational Rhapsody toolbar is added into Visual Studio that enables navigating to the Rational Rhapsody model directly from the code in Visual Studio.

A Visual Studio project can now be created, that is connected to the Rational Rhapsody configuration and the integration updates 'include paths' and 'define symbols' directly into the Visual Studio project.

The code in Visual Studio and Rational Rhapsody model remain synchronized with one another through dynamic model code associatively. For example, changing the name of a class within Visual Studio is recognized within the Rational Rhapsody model.

If files are added in Rational Rhapsody, then they appear in Visual Studio. Existing Visual Studio projects can be exported to Rational Rhapsody and brought into the Rational Rhapsody model through reverse engineering.

Further, with Rational Rhapsody 7.5.2, an updated toolbar in Visual Studio makes it easier to work with multiple projects and components in Visual Studio and Rational Rhapsody.

The new Synchronization button helps maintain synchronization when the model or code changes without dynamic synchronization enabled.

Additionally, it is now possible to export all projects in a Visual Studio solution to Rational Rhapsody, which creates components and packages within Rational Rhapsody synchronized with each Microsoft Visual Studio project in the solution.

The round-tripping of model and code information is improved by allowing synchronization to occur without the need to switch the active project in Visual Studio or an active configuration in Rational Rhapsody. The changes are synchronized without that added step.

# Setting up the Integration

Setting up the integration is fairly simple. The integration can be installed by selecting the 'Microsoft Visual Studio 2008 Workflow Integration' Add-on during the installation of Rational Rhapsody or while performing a modify installation. Modifying the installation can be done either from the 'Control Panel' or using the 'msi' installer.

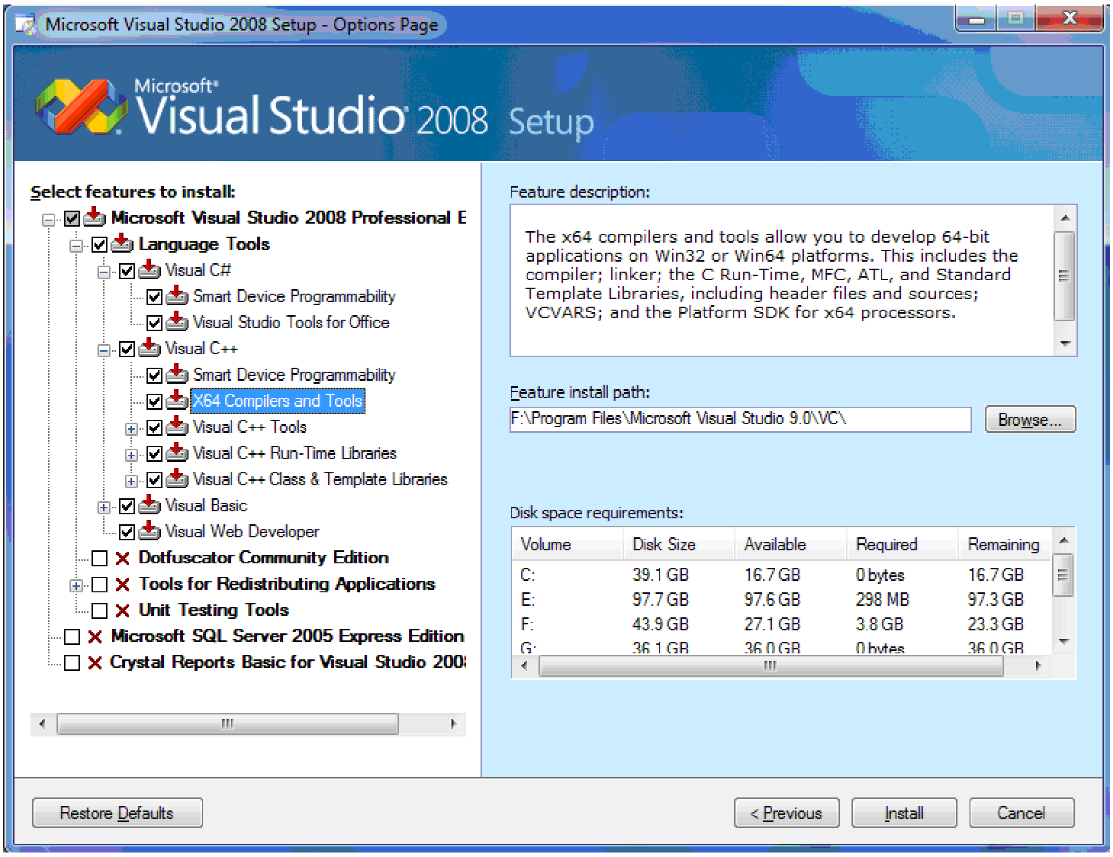
## *Download and Install Visual Studio 2008*

90 Day Trial link:

http://www.microsoft.com/downloads/details.aspx?FamilyId=83C3A1EC-ED72-4A79-8961-25635DB0192B&displaylang=en

During the installation, select the language of your choice (Rational Rhapsody 7.5.2 supports C, C++ and C#).

If 64 bit applications are to be developed, ensure the 'X64 Compiler and Tools' feature is selected.



Microsoft Visual Studio 2008 Setup - Options Page

Microsoft® Visual Studio 2008 Setup

Select features to install:

- ☑ Microsoft Visual Studio 2008 Professional E
  - ☑ Language Tools
    - ☑ Visual C#
      - ☑ Smart Device Programmability
      - ☑ Visual Studio Tools for Office
    - ☑ Visual C++
      - ☑ Smart Device Programmability
      - ☑ X64 Compilers and Tools
      - ☑ Visual C++ Tools
      - ☑ Visual C++ Run-Time Libraries
      - ☑ Visual C++ Class & Template Libraries
    - ☑ Visual Basic
    - ☑ Visual Web Developer
  - □ ✗ Dotfuscator Community Edition
  - □ ✗ Tools for Redistributing Applications
  - □ ✗ Unit Testing Tools
- □ ✗ Microsoft SQL Server 2005 Express Edition
- □ ✗ Crystal Reports Basic for Visual Studio 200!

Feature description:

The x64 compilers and tools allow you to develop 64-bit applications on Win32 or Win64 platforms. This includes the compiler; linker; the C Run-Time, MFC, ATL, and Standard Template Libraries, including header files and sources; VCVARS; and the Platform SDK for x64 processors.

Feature install path:
F:\Program Files\Microsoft Visual Studio 9.0\VC\

Disk space requirements:

| Volume | Disk Size | Available | Required | Remaining |
|---|---|---|---|---|
| C: | 39.1 GB | 16.7 GB | 0 bytes | 16.7 GB |
| E: | 97.7 GB | 97.6 GB | 298 MB | 97.3 GB |
| F: | 43.9 GB | 27.1 GB | 3.8 GB | 23.3 GB |
| G: | 36.1 GB | 36.0 GB | 0 bytes | 36.0 GB |

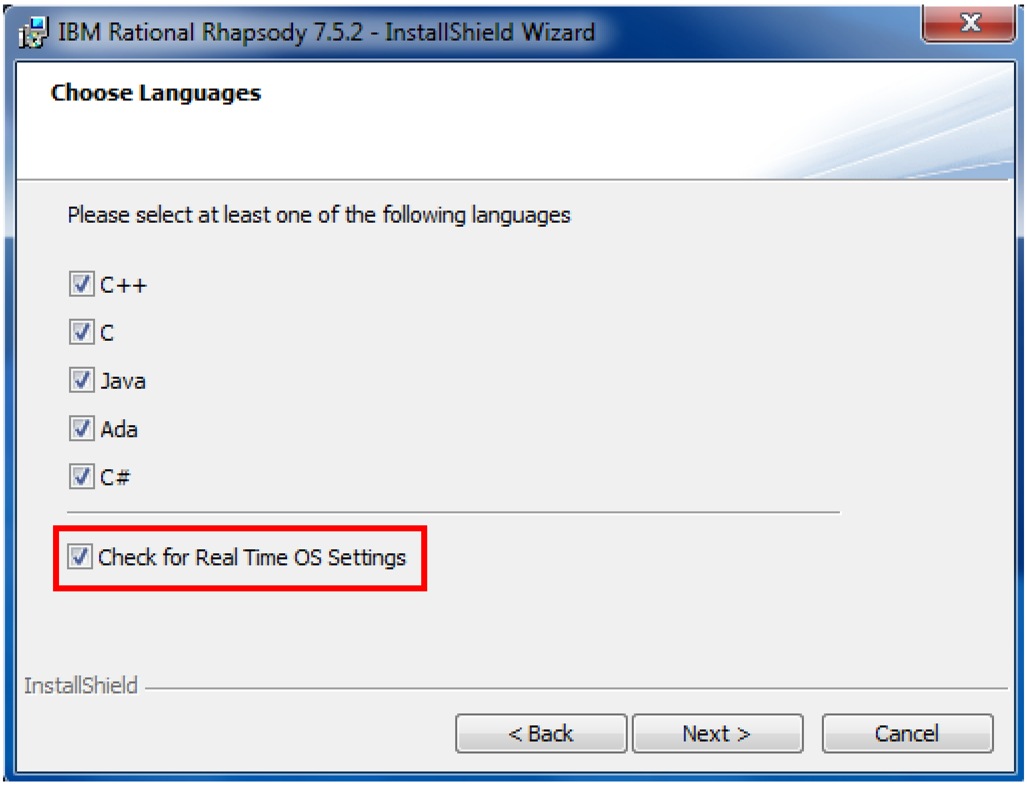
## *Download and install Rational Rhapsody*

30 Day Trial link:

http://www.ibm.com/developerworks/downloads/r/rhapsodydeveloper/index.html

This document uses IBM Rational Rhapsody 7.5.2.

During the installation, on the 'Choose Languages' window, check the option for "Check for Real Time OS settings".



IBM Rational Rhapsody 7.5.2 - InstallShield Wizard

Choose Languages

Please select at least one of the following languages

☑ C++

☑ C

☑ Java

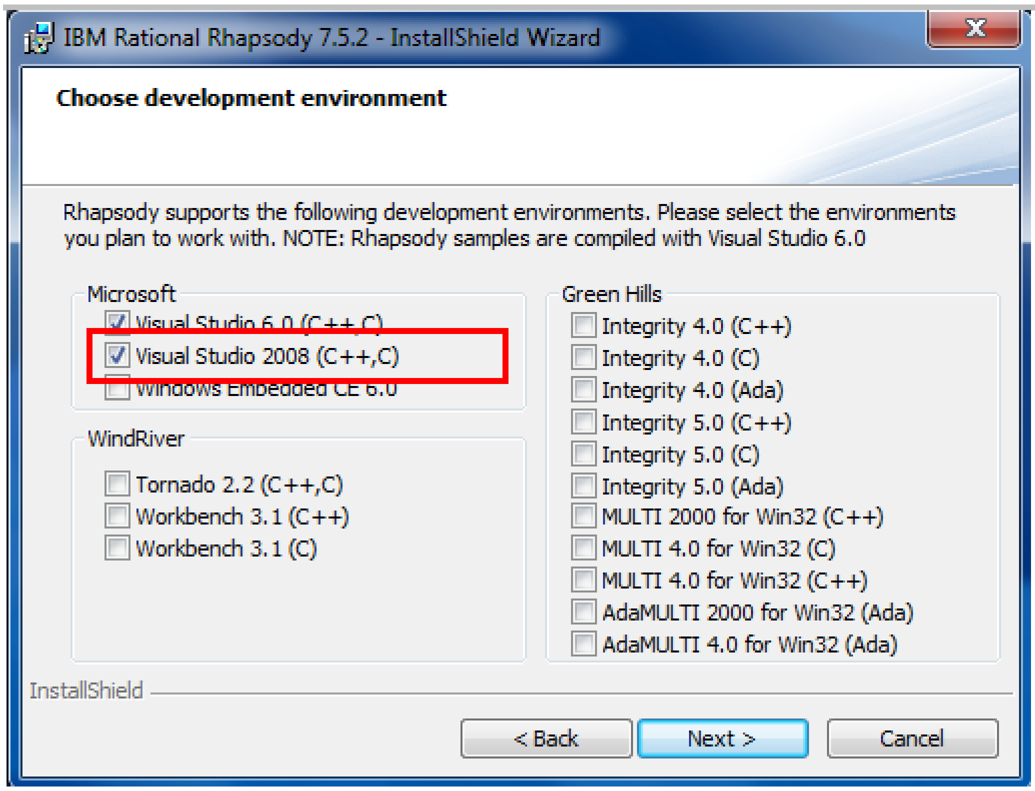
☑ Ada

☑ C#

☑ Check for Real Time OS Settings

On the 'Choose development environment' window, check the option for 'Visual Studio 2008'.

The 'Microsoft Visual Studio 2008 Workflow Integration' Add-on will not be available for selection if Visual Studio 2008 is not selected as a compiler (development environment) on the "Choose development environment" window as shown below.



IBM Rational Rhapsody 7.5.2 - InstallShield Wizard

**Choose development environment**

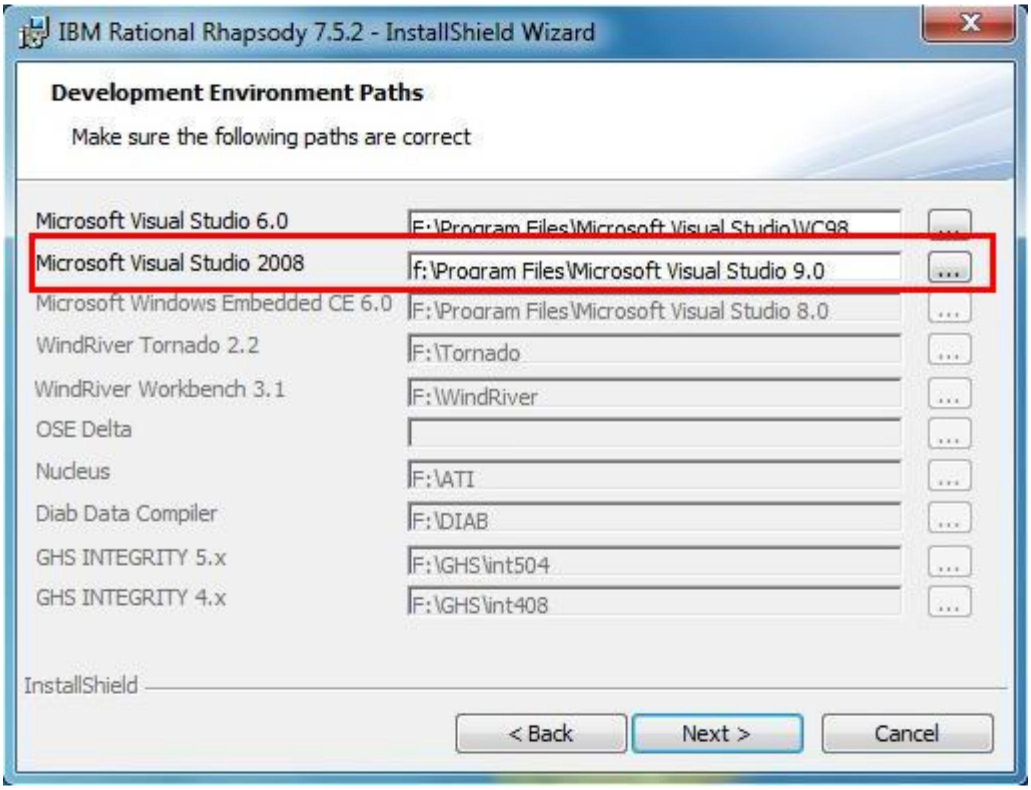
Rhapsody supports the following development environments. Please select the environments you plan to work with. NOTE: Rhapsody samples are compiled with Visual Studio 6.0

Microsoft
- ☑ Visual Studio 6.0 (C++,C)
- ☑ Visual Studio 2008 (C++,C)
- □ Windows Embedded CE 6.0

WindRiver
- □ Tornado 2.2 (C++,C)
- □ Workbench 3.1 (C++)
- □ Workbench 3.1 (C)

Green Hills
- □ Integrity 4.0 (C++)
- □ Integrity 4.0 (C)
- □ Integrity 4.0 (Ada)
- □ Integrity 5.0 (C++)
- □ Integrity 5.0 (C)
- □ Integrity 5.0 (Ada)
- □ MULTI 2000 for Win32 (C++)
- □ MULTI 4.0 for Win32 (C)
- □ MULTI 4.0 for Win32 (C++)
- □ AdaMULTI 2000 for Win32 (Ada)
- □ AdaMULTI 4.0 for Win32 (Ada)

Browse and select the path to the visual studio installation.



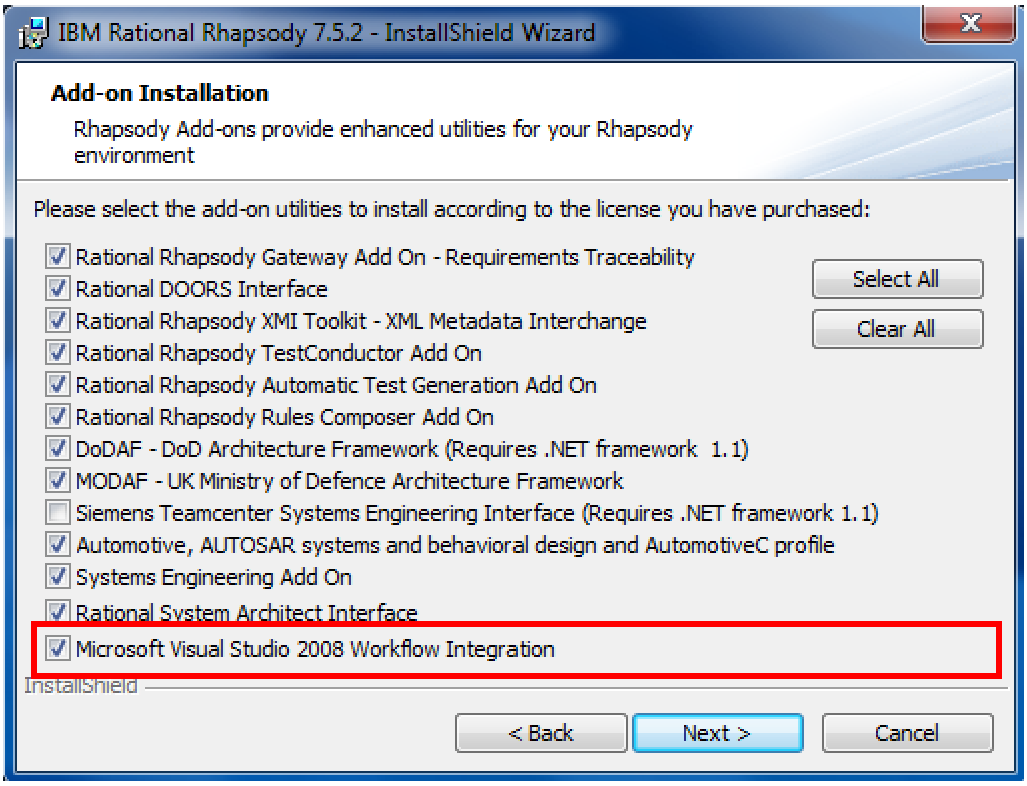
IBM Rational Rhapsody 7.5.2 - InstallShield Wizard

**Development Environment Paths**

Make sure the following paths are correct

| | |
|---|---|
| Microsoft Visual Studio 6.0 | F:\Program Files\Microsoft Visual Studio\VC98 |
| Microsoft Visual Studio 2008 | f:\Program Files\Microsoft Visual Studio 9.0 |
| Microsoft Windows Embedded CE 6.0 | F:\Program Files\Microsoft Visual Studio 8.0 |
| WindRiver Tornado 2.2 | F:\Tornado |
| WindRiver Workbench 3.1 | F:\WindRiver |
| OSE Delta | |
| Nucleus | F:\ATI |
| Diab Data Compiler | F:\DIAB |
| GHS INTEGRITY 5.x | F:\GHS\int504 |
| GHS INTEGRITY 4.x | F:\GHS\int408 |

Finally at the 'Add-on Installation' window select the 'Microsoft Visual Studio Workflow Integration' Add-on.



**Add-on Installation**

Rhapsody Add-ons provide enhanced utilities for your Rhapsody environment

Please select the add-on utilities to install according to the license you have purchased:

- ☑ Rational Rhapsody Gateway Add On - Requirements Traceability
- ☑ Rational DOORS Interface
- ☑ Rational Rhapsody XMI Toolkit - XML Metadata Interchange
- ☑ Rational Rhapsody TestConductor Add On
- ☑ Rational Rhapsody Automatic Test Generation Add On
- ☑ Rational Rhapsody Rules Composer Add On
- ☑ DoDAF - DoD Architecture Framework (Requires .NET framework 1.1)
- ☑ MODAF - UK Ministry of Defence Architecture Framework
- □ Siemens Teamcenter Systems Engineering Interface (Requires .NET framework 1.1)
- ☑ Automotive, AUTOSAR systems and behavioral design and AutomotiveC profile
- ☑ Systems Engineering Add On
- ☑ Rational System Architect Interface
- ☑ Microsoft Visual Studio 2008 Workflow Integration

By default this Add-on is automatically selected for installation, if the Visual Studio 2008 settings are gathered correctly.

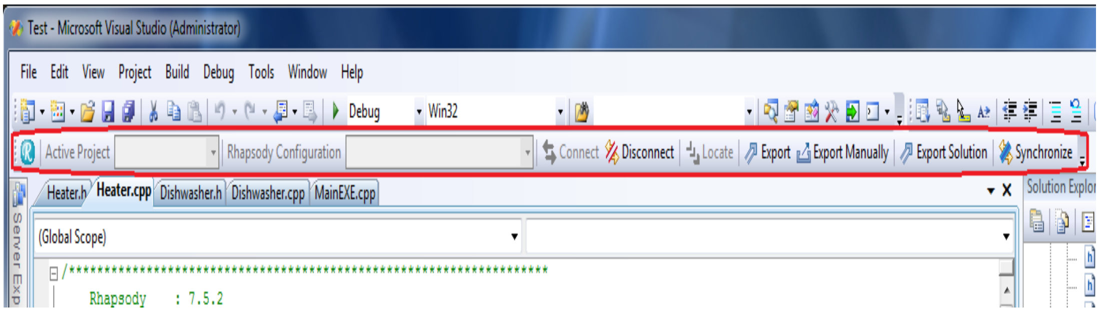
Once the installation is complete, the Rhapsody – Visual Studio workflow integration is ready to use.

# Integration Features

This section describes the various integration features and functionalities.

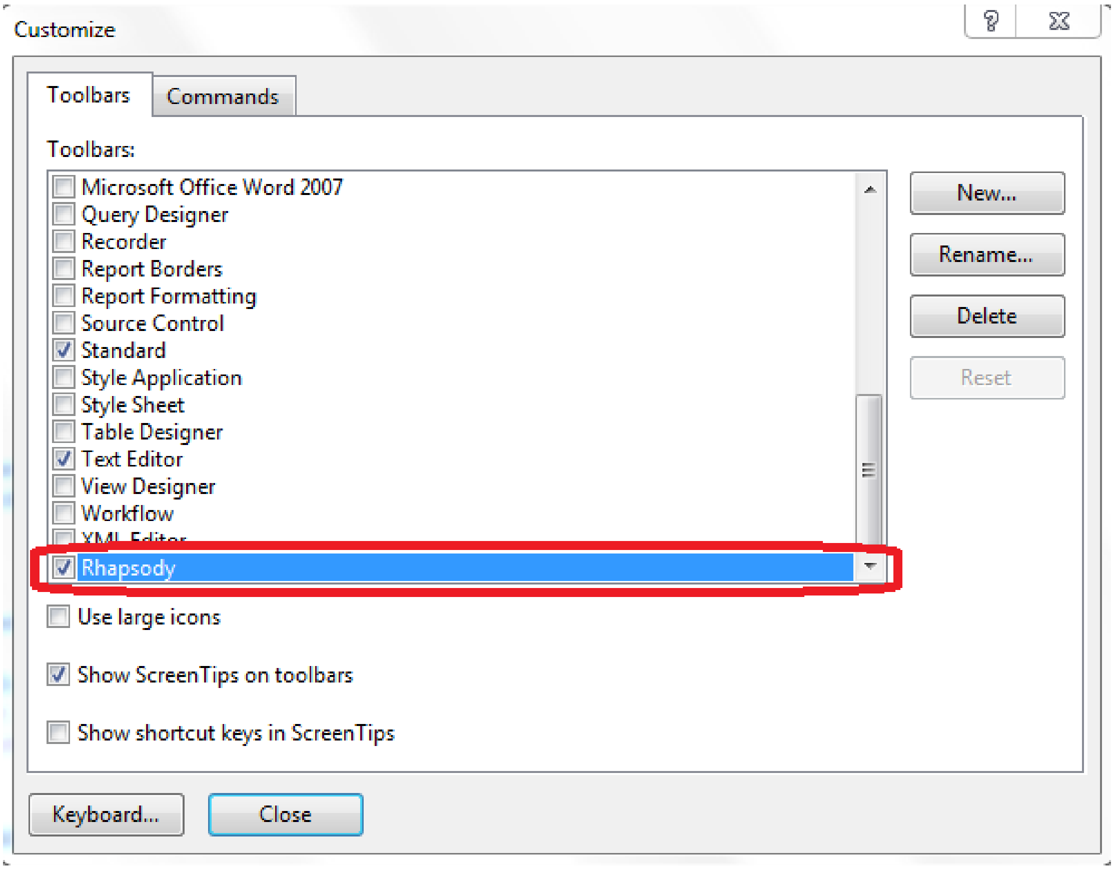
## *Rhapsody Toolbar*

The integration creates a toolbar for Rational Rhapsody in the Visual Studio 2008 IDE.



```
/*********************************************************************
    Rhapsody    : 7.5.2
```

The toolbar can be removed/added by toggling the corresponding Toolbar check box in the customize window.



Toolbars:

- □ Microsoft Office Word 2007
- □ Query Designer
- □ Recorder
- □ Report Borders
- □ Report Formatting
- □ Source Control
- ☑ Standard
- □ Style Application
- □ Style Sheet
- □ Table Designer
- ☑ Text Editor
- □ View Designer
- □ Workflow
- □ XML Editor
- ☑ Rhapsody

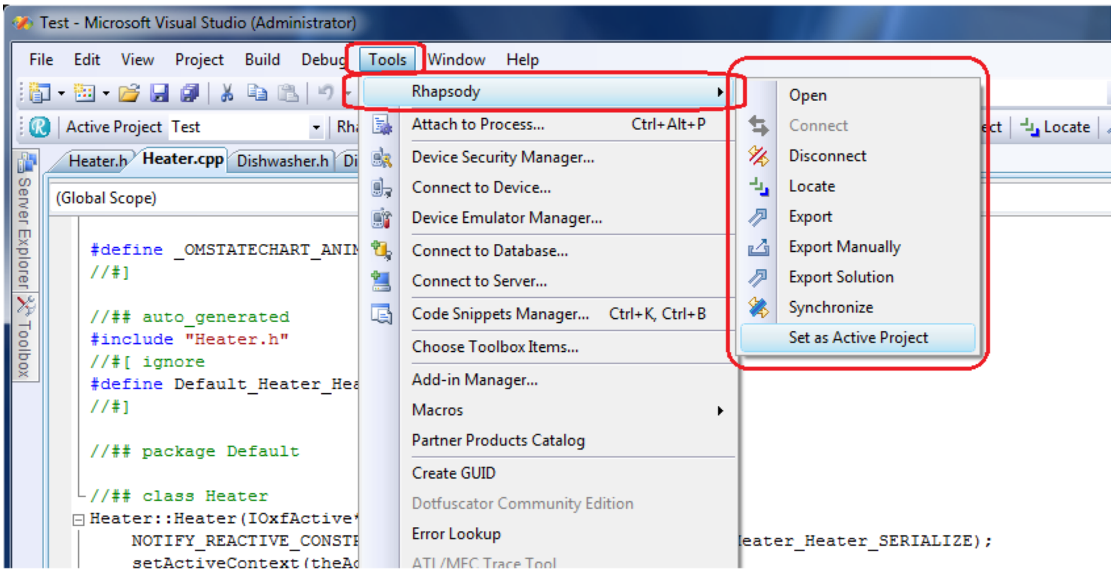
□ Use large icons

☑ Show ScreenTips on toolbars

□ Show shortcut keys in ScreenTips

## Rhapsody Menu

The integration also creates a set of Rational Rhapsody menus under 'Tools' -> 'Rhapsody' submenu in the Visual Studio 2008 IDE.

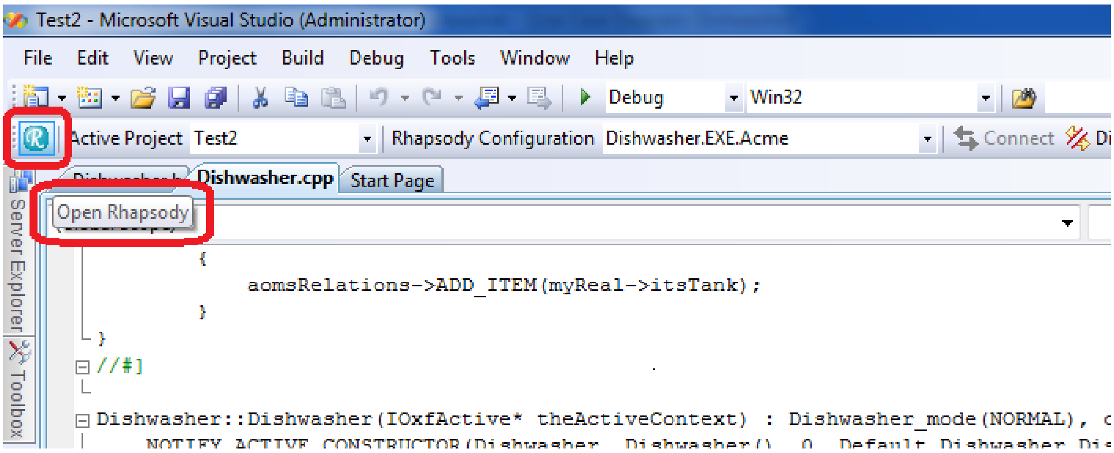


```
#define _OMSTATECHART_ANIM
//#]

//## auto_generated
#include "Heater.h"
//#[ ignore
#define Default_Heater_Hea
//#]

//## package Default

//## class Heater
Heater::Heater(IOxfActive*
    NOTIFY_REACTIVE_CONSTR
    setActiveContext(theAc
```

```
eater_Heater_SERIALIZE);
```

## Open Rhapsody

This feature starts and establishes communication with Rational Rhapsody. If Rational Rhapsody is already running, communication is established with that instance of the program.

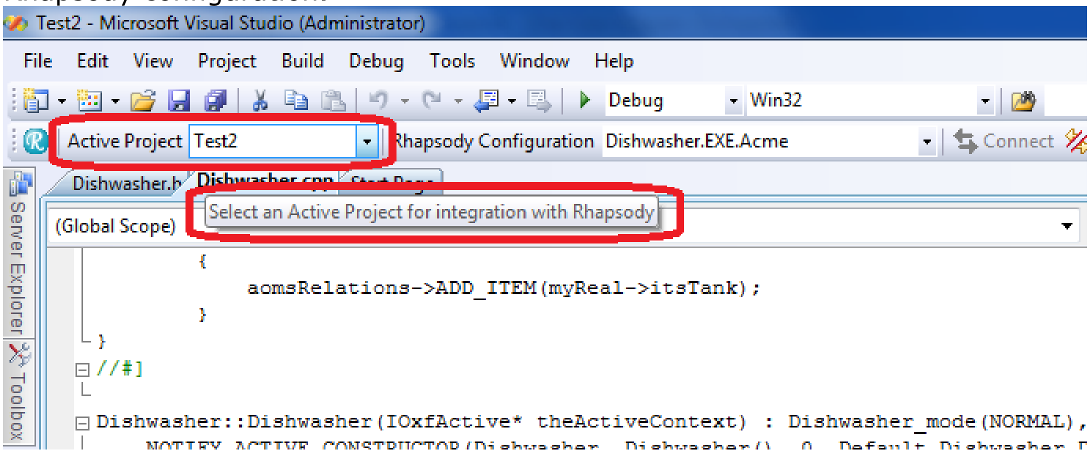

Test2 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Tools Window Help
Debug
Win32
Active Project Test2
Rhapsody Configuration Dishwasher.EXE.Acme
Connect
Di
Dishwasher.h
Dishwasher.cpp
Start Page
Open Rhapsody
Server Explorer
Toolbox

```
            {
                aomsRelations->ADD_ITEM(myReal->itsTank);
            }
}
//#]

Dishwasher::Dishwasher(IOxfActive* theActiveContext) : Dishwasher_mode(NORMAL), c
    NOTIFY_ACTIVE_CONSTRUCTOR(Dishwasher, Dishwasher(), 0, Default_Dishwasher_Dis
```

The icon, rather the entire toolbar is disabled if no solution is currently open in Microsoft Visual Studio 2008.

## Active Project

This feature allows selection of an active project for integration with Rational Rhapsody configuration.


Test2 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Tools Window Help
Debug
Win32
Connect
Dishwasher.h
Dishwasher.cpp
Start Page
(Global Scope)
Server Explorer
Toolbox

Active Project Test2

Rhapsody Configuration Dishwasher.EXE.Acme

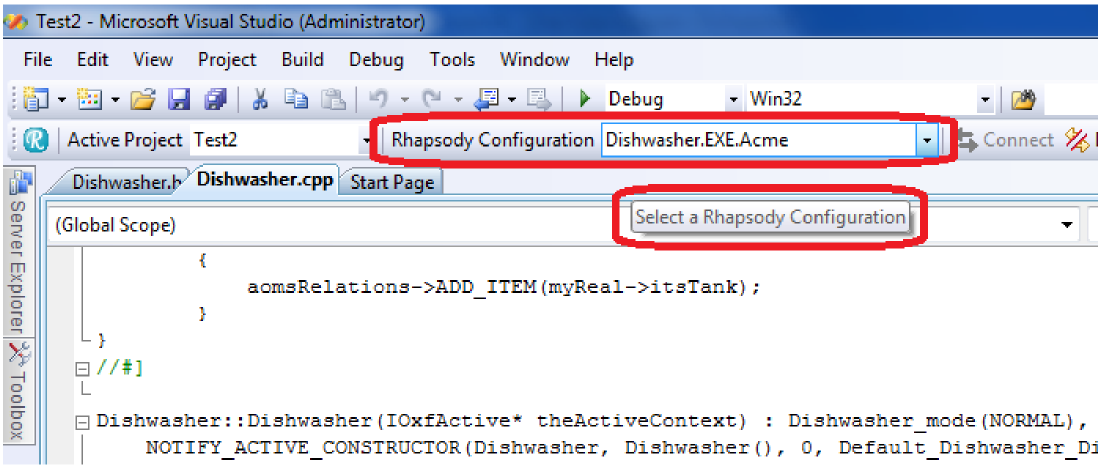
Select an Active Project for integration with Rhapsody

```
            {
                aomsRelations->ADD_ITEM(myReal->itsTank);
            }
    }
//#]

Dishwasher::Dishwasher(IOxfActive* theActiveContext) : Dishwasher_mode(NORMAL),
    NOTIFY_ACTIVE_CONSTRUCTOR(Dishwasher, Dishwasher(), 0, Default_Dishwasher_D
```

This functionality can be used to select the project that is to be imported into Rational Rhapsody.

## Rhapsody Configuration

The Rhapsody Configuration list in the Rational Rhapsody toolbar can be used to select the Rational Rhapsody configuration to which the Visual Studio project would be linked.


Test2 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Tools Window Help
Debug
Win32
Connect
Dishwasher.h
Dishwasher.cpp
Start Page
(Global Scope)
Server Explorer
Toolbox

Active Project Test2

Rhapsody Configuration Dishwasher.EXE.Acme

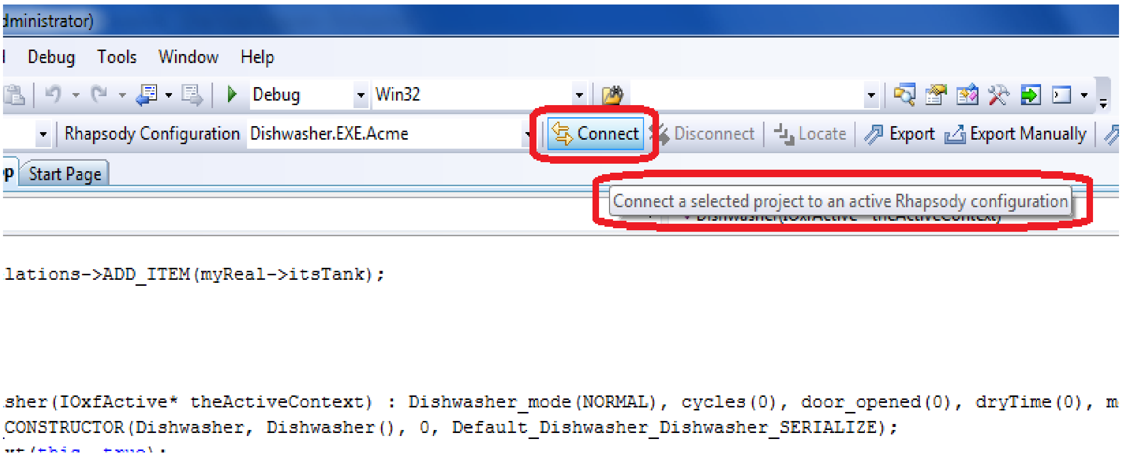
Select a Rhapsody Configuration

```
            {
                aomsRelations->ADD_ITEM(myReal->itsTank);
            }
    }
//#]

Dishwasher::Dishwasher(IOxfActive* theActiveContext) : Dishwasher_mode(NORMAL),
    NOTIFY_ACTIVE_CONSTRUCTOR(Dishwasher, Dishwasher(), 0, Default_Dishwasher_Di
```

This functionality could be used to select an existing configuration or create a new one.

## Connect

This feature is used to connect a selected project to an active Rhapsody configuration.



Connect a selected project to an active Rhapsody configuration

```
lations->ADD_ITEM(myReal->itsTank);


sher(IOxfActive* theActiveContext) : Dishwasher_mode(NORMAL), cycles(0), door_opened(0), dryTime(0), m
CONSTRUCTOR(Dishwasher, Dishwasher(), 0, Default_Dishwasher_Dishwasher_SERIALIZE);
```

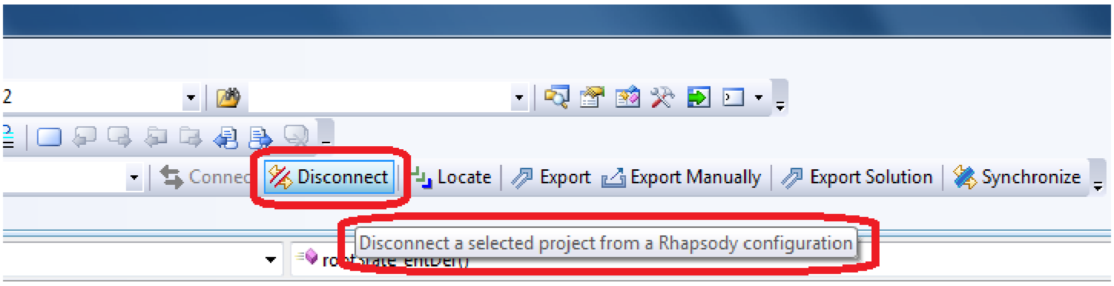
The feature would be used to generate the code directly into the Visual Studio Project directory.

If a Rational Rhapsody configuration has not been selected, an informational message is displayed, allowing users to connect to a new one.

Adding Rational Rhapsody generated files to the Visual Studio project is optional.

## Disconnect

This feature is used to disconnect a selected project from an active Rhapsody configuration.

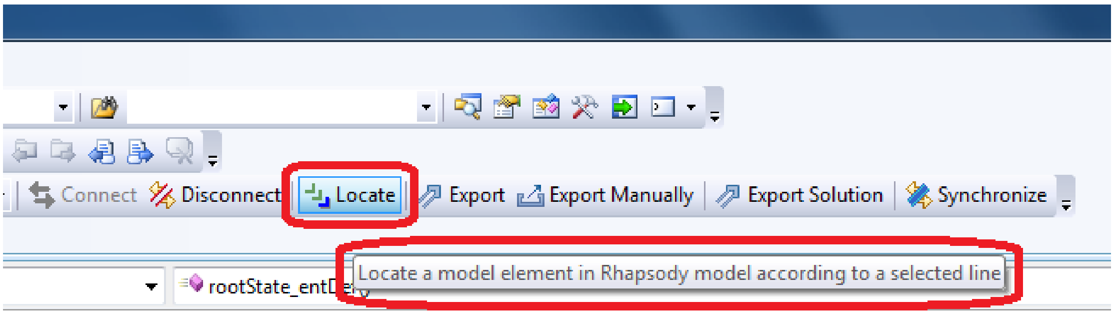


Disconnect a selected project from a Rhapsody configuration

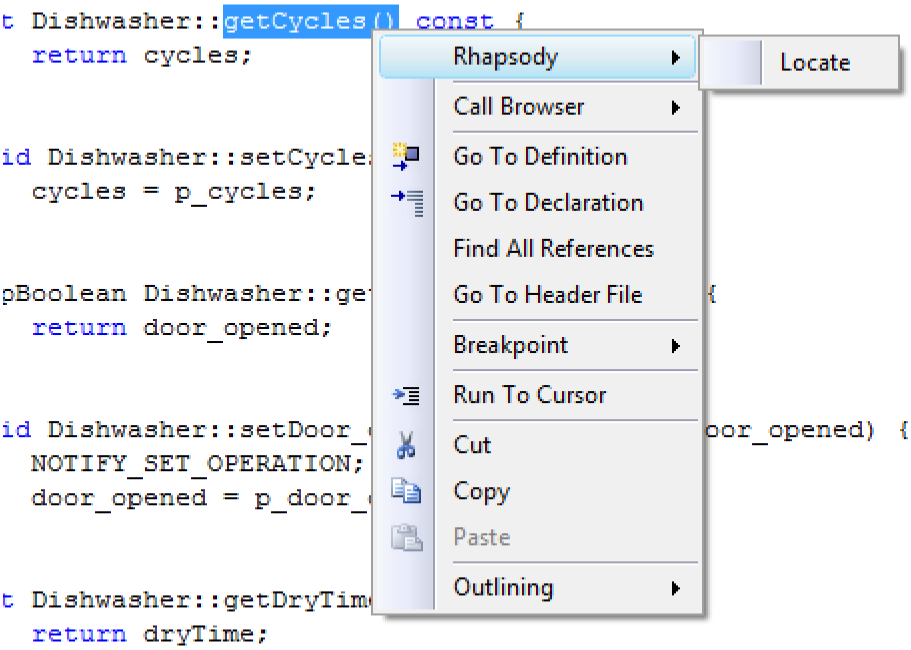
You have the choice to remove Rhapsody generated files and build settings when disconnecting a Visual Studio Project from a Rhapsody configuration.

## *Locate*

This feature is used to locate a model element in Rhapsody model according to a selected line in the active code file.



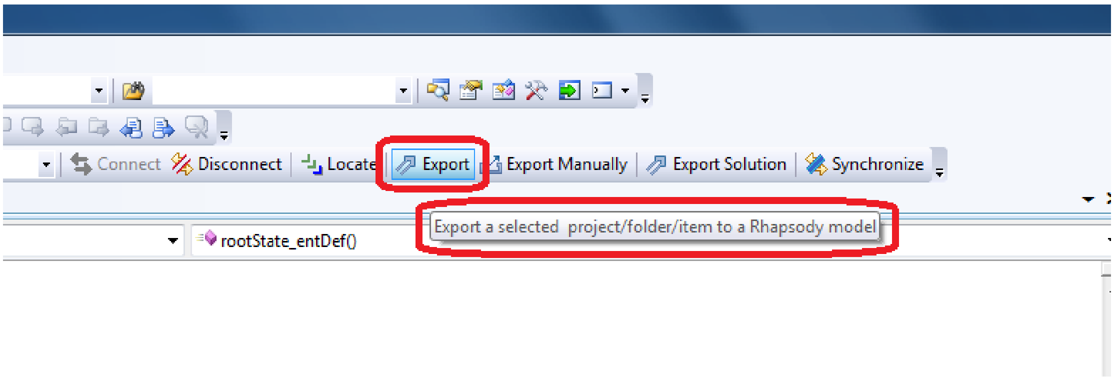
This feature can also be accessed by a right click at any place in the code file.



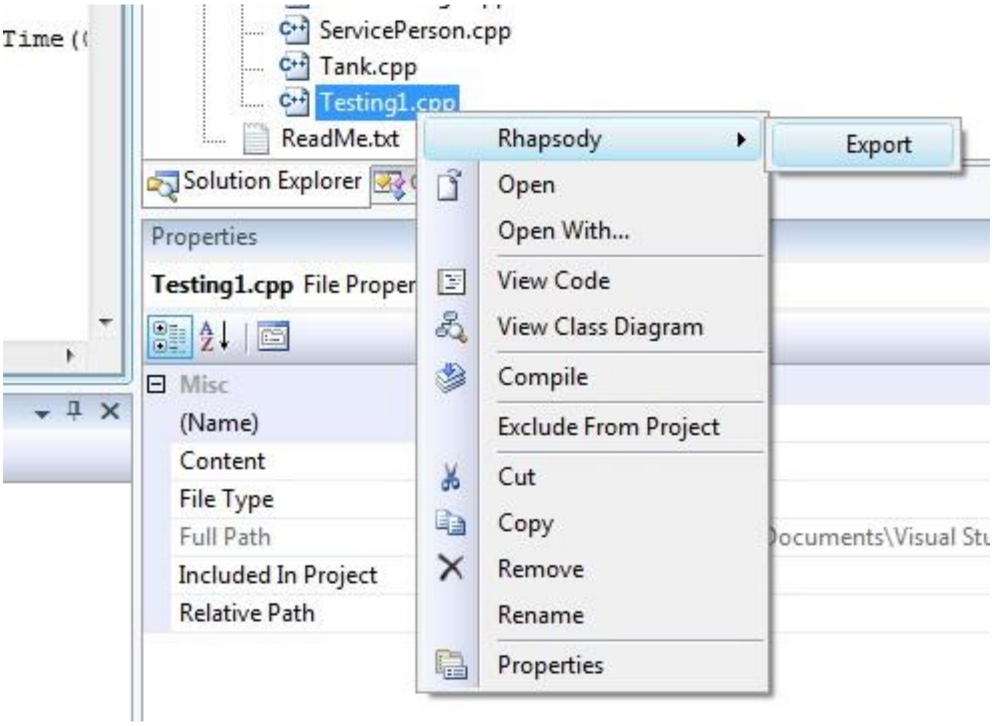
```
t Dishwasher::getCycles() const {
  return cycles;

id Dishwasher::setCycle
  cycles = p_cycles;

pBoolean Dishwasher::ge                                    {
  return door_opened;

id Dishwasher::setDoor_                                oor_opened) {
  NOTIFY_SET_OPERATION;
  door_opened = p_door_

t Dishwasher::getDryTim
  return dryTime;
```

## Export

This feature allows you to export a selected Visual Studio element (project / folder / item) to a Rational Rhapsody model.



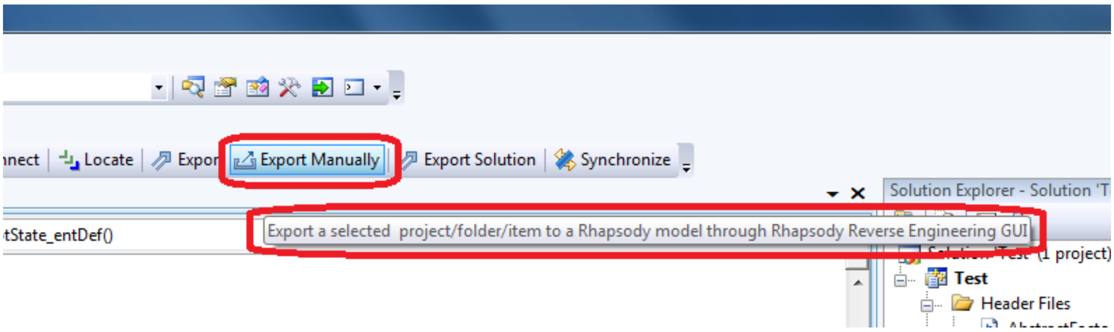
Specific code files can be exported directly to the Rational Rhapsody project by selecting the same in the Solution Explorer of the Visual Studio IDE and using the right click menu 'Rhapsody' → 'Export'.



The feature can also be invoked using the right click menu for selected Visual Studio elements.

## Export Manually

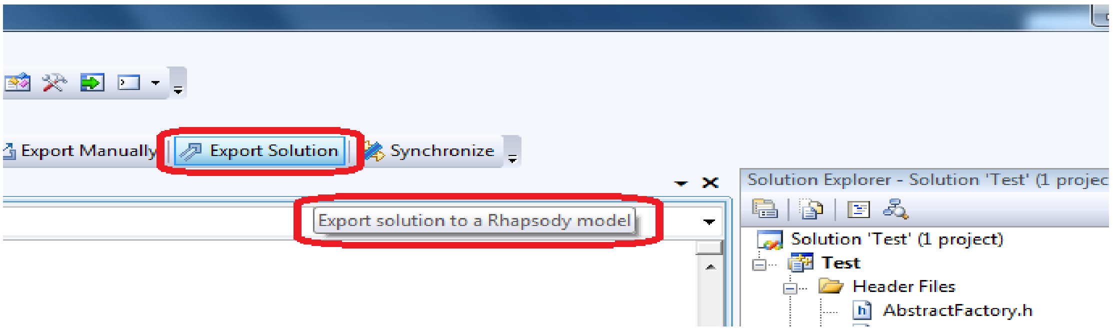
This feature invokes the Reverse Engineering Tool and allows you to manually administer the export process of a selected Visual Studio element (project / folder / item).



Export a selected project/folder/item to a Rhapsody model through Rhapsody Reverse Engineering GUI

## Export Solution

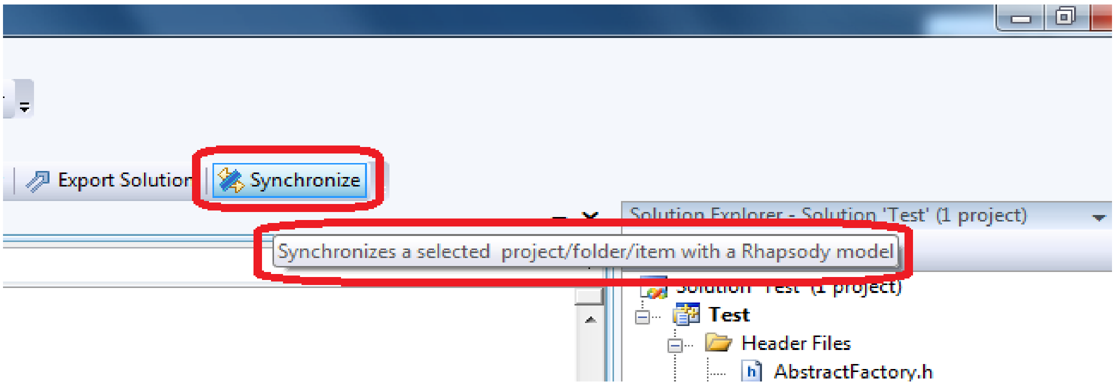
This feature is used to export a complete Visual Studio solution to a Rhapsody model.



Export solution to a Rhapsody model

## Synchronize

This feature allows you to synchronize a selected Visual Studio element (project / folder / item) with a Rhapsody model.



Export Solution
Synchronize
Solution Explorer - Solution 'Test' (1 project)
Solution 'Test' (1 project)
Test
Header Files
AbstractFactory.h

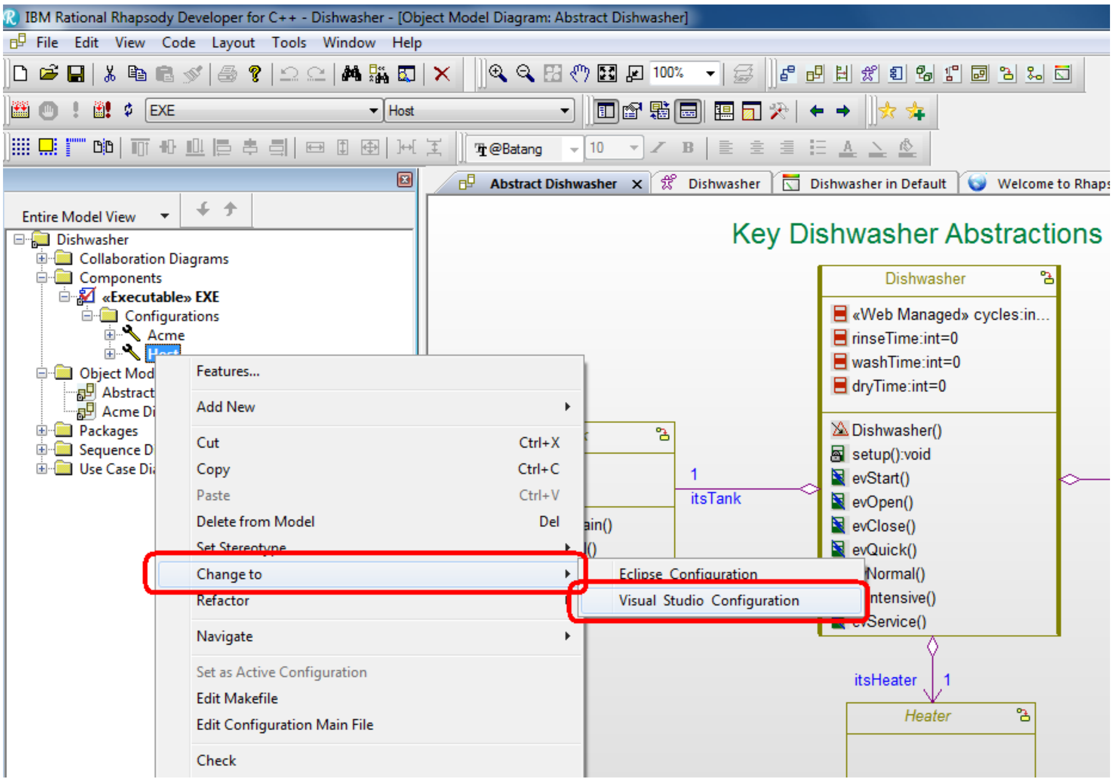
Synchronizes a selected project/folder/item with a Rhapsody model

# Integration Workflow

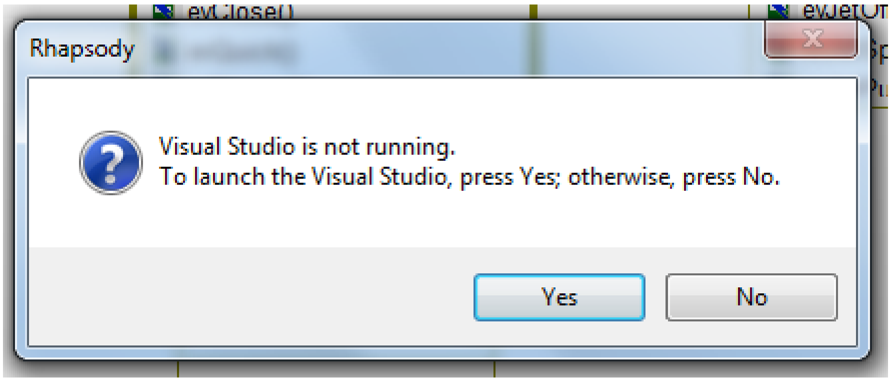
## *Creating Visual Studio project from Rational Rhapsody model*

Procedure:

1. Open the model in Rational Rhapsody.
2. In the Rational Rhapsody browser, select an existing configuration or create one.
    a. If a new configuration is being created, create a Visual Studio configuration.
    b. If an existing configuration is being used, use the pop-up menu to change the configuration to a Visual Studio configuration.

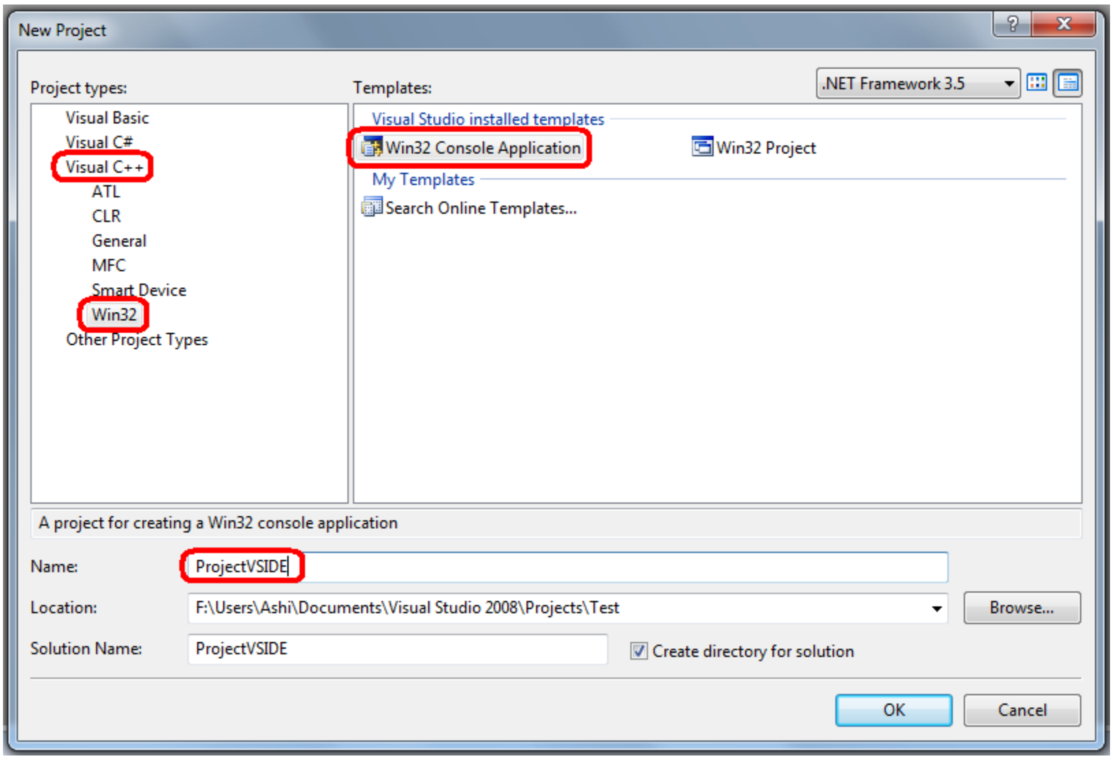


3. If Visual Studio is not open a confirmatory message is displayed.



Visual Studio is not running.
To launch the Visual Studio, press Yes; otherwise, press No.

4. The focus shifts to Visual Studio and the New Project window is displayed.



Project types: Visual Basic, Visual C#, Visual C++ (ATL, CLR, General, MFC, Smart Device, Win32), Other Project Types

Templates: Win32 Console Application, Win32 Project, Search Online Templates...

A project for creating a Win32 console application

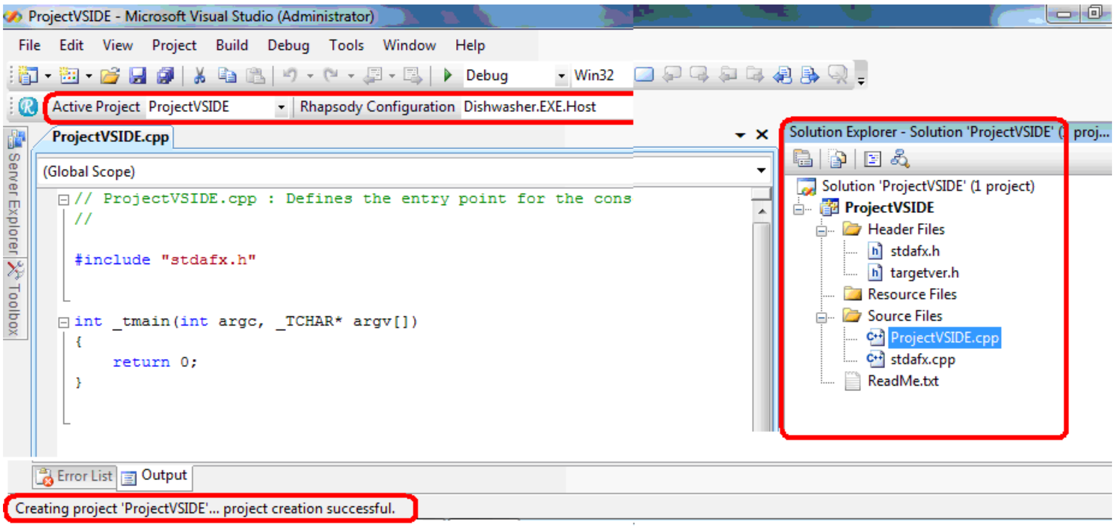
Name: ProjectVSIDE

Location: F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test

Solution Name: ProjectVSIDE

☑ Create directory for solution

5. Using the New Project window, create a Visual Studio project. After the new project is created, the project name is displayed as the Active Project in the Rational Rhapsody toolbar in Visual Studio. The Rational Rhapsody configuration selected is displayed as the Rhapsody Configuration in the toolbar. The new Visual Studio project is now linked to the Rational Rhapsody model. All changes to the model are reflected in the project code, and all changes to the project code are round-tripped into the model.



Active Project ProjectVSIDE

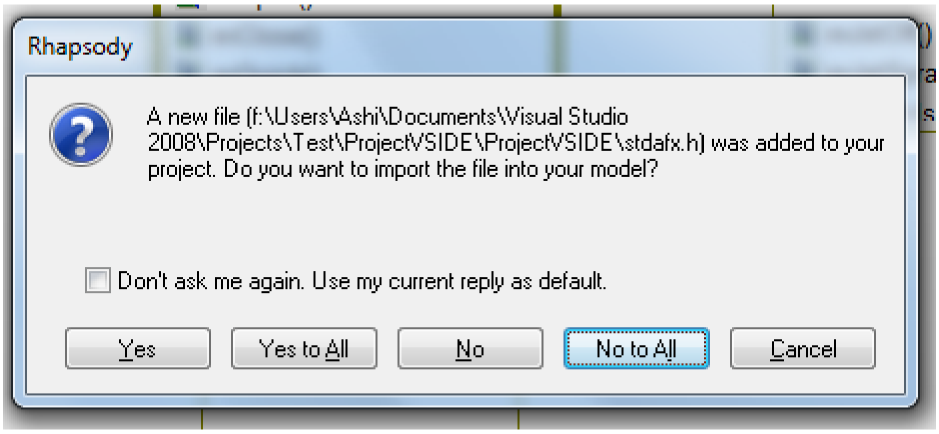
Rhapsody Configuration Dishwasher.EXE.Host

```
// ProjectVSIDE.cpp : Defines the entry point for the cons
//

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}
```

```
Solution 'ProjectVSIDE' (1 project)
  ProjectVSIDE
    Header Files
      stdafx.h
      targetver.h
    Resource Files
    Source Files
      ProjectVSIDE.cpp
      stdafx.cpp
    ReadMe.txt
```

Creating project 'ProjectVSIDE'... project creation successful.

6. Upon switching to the Rational Rhapsody window, Visual Studio creates a few files such as ‘stdafx.cpp’, ‘stdafx.h’ and ‘targetver.h’, which Rational Rhapsody wants to add to the model. Unless these files are needed, select ‘No to All’ at the below message.

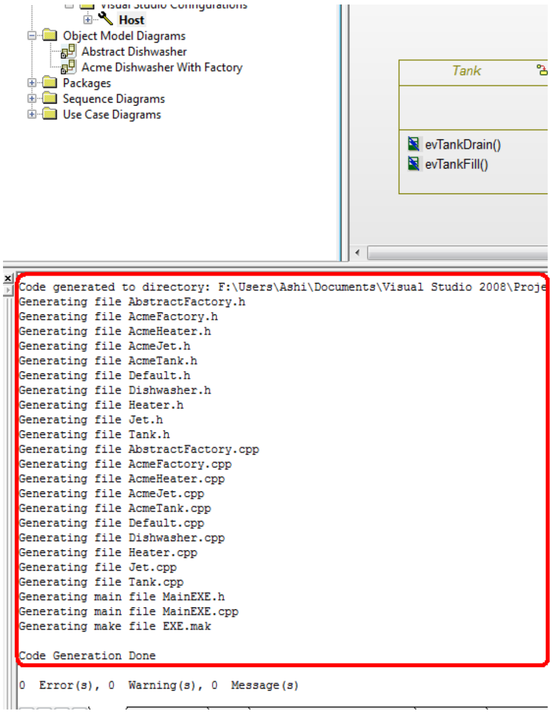


A new file (f:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\stdafx.h) was added to your project. Do you want to import the file into your model?

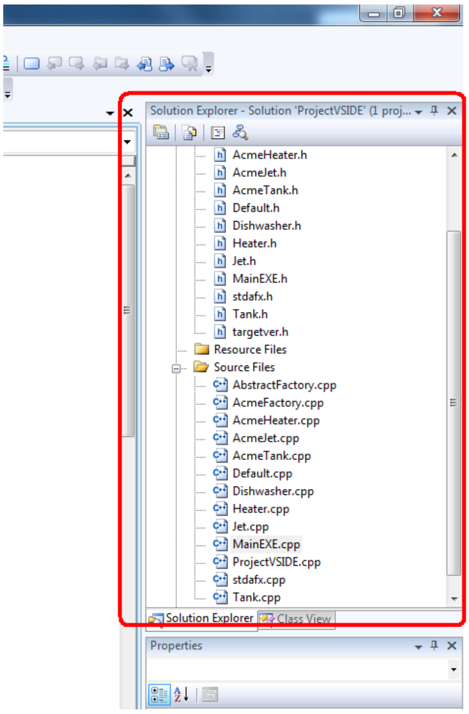
□ Don't ask me again. Use my current reply as default.

7. To create the code files, select Code -> Generate in Rational Rhapsody. Once the code has been generated, they are reflected in the Visual Studio project.

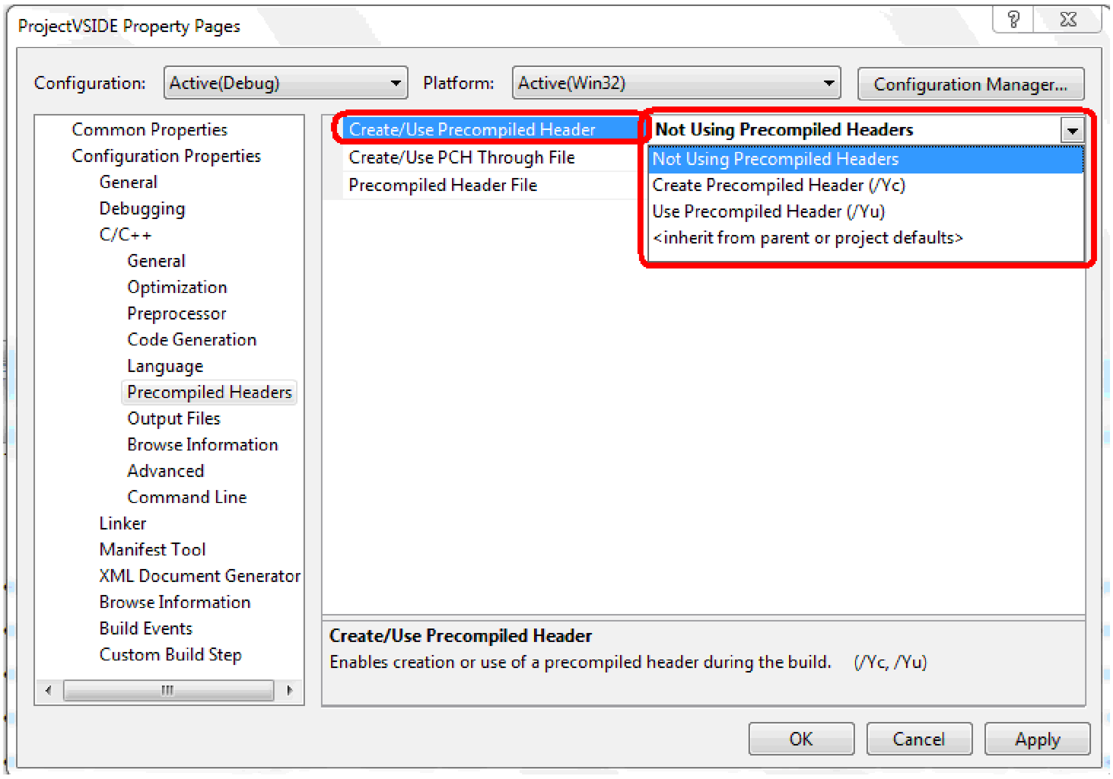
   Observe the code being generated into the newly created Visual Studio project’s directory and also reflected in the Visual Studio IDE.



```
Code generated to directory: F:\Users\Ashi\Documents\Visual Studio 2008\Proje
Generating file AbstractFactory.h
Generating file AcmeFactory.h
Generating file AcmeHeater.h
Generating file AcmeJet.h
Generating file AcmeTank.h
Generating file Default.h
Generating file Dishwasher.h
Generating file Heater.h
Generating file Jet.h
Generating file Tank.h
Generating file AbstractFactory.cpp
Generating file AcmeFactory.cpp
Generating file AcmeHeater.cpp
Generating file AcmeJet.cpp
Generating file AcmeTank.cpp
Generating file Default.cpp
Generating file Dishwasher.cpp
Generating file Heater.cpp
Generating file Jet.cpp
Generating file Tank.cpp
Generating main file MainEXE.h
Generating main file MainEXE.cpp
Generating make file EXE.mak

Code Generation Done

0 Error(s), 0 Warning(s), 0 Message(s)
```



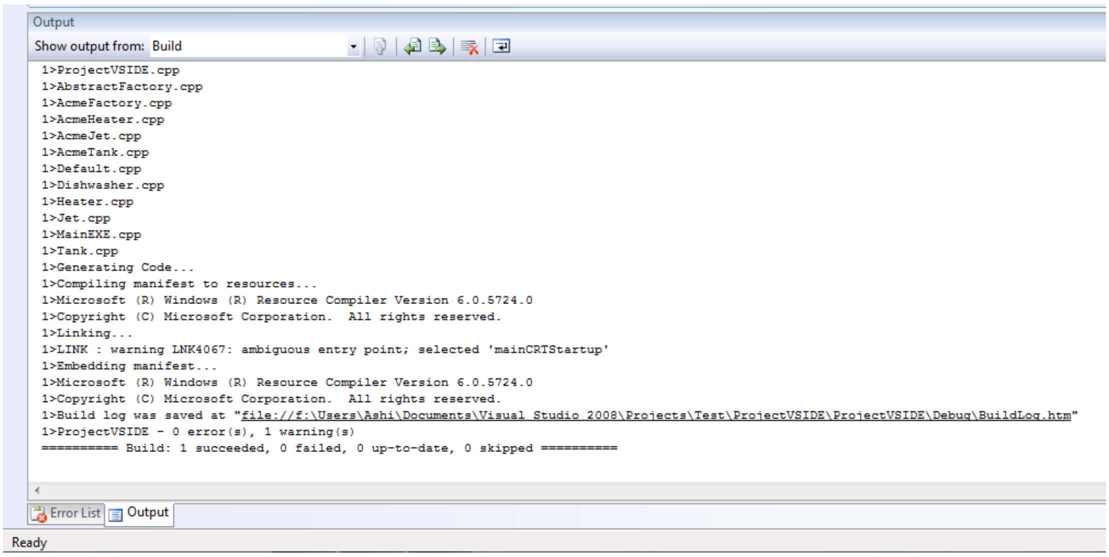
8. In the project's property pages, select 'Not Using Precompiled Headers' if precompiled headers like 'stdafx.h' are not being used.



Create/Use Precompiled Header: Not Using Precompiled Headers
Create/Use PCH Through File
Precompiled Header File

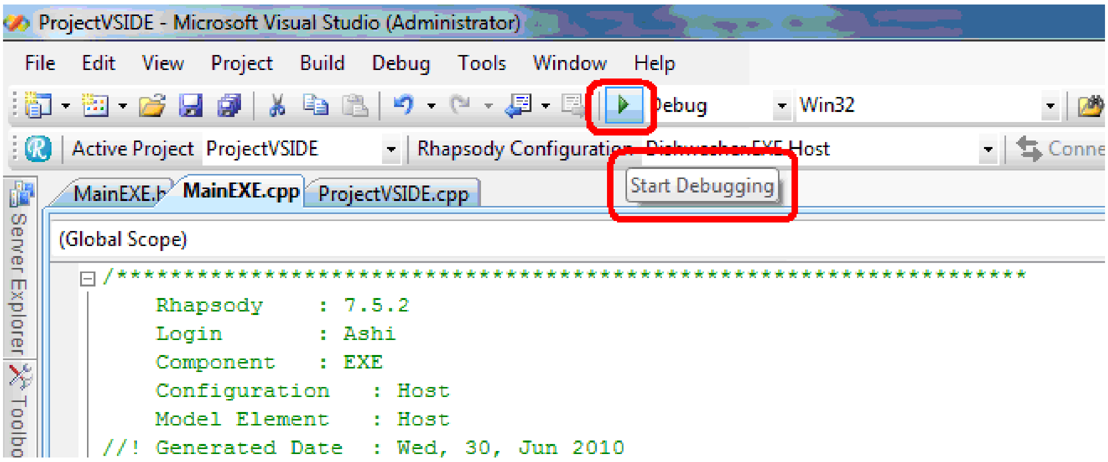
**Create/Use Precompiled Header**
Enables creation or use of a precompiled header during the build. (/Yc, /Yu)

9. Click on GMR in the Rational Rhapsody IDE and observe the build being performed in the Visual Studio IDE.



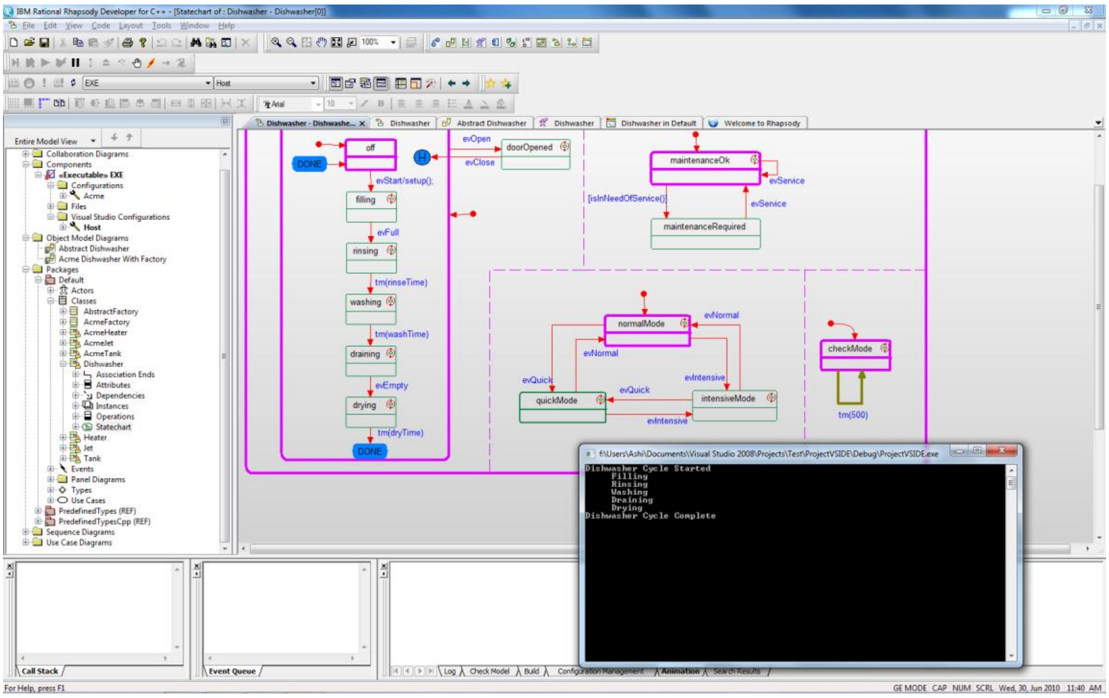
```
1>ProjectVSIDE.cpp
1>AbstractFactory.cpp
1>AcmeFactory.cpp
1>AcmeHeater.cpp
1>AcmeJet.cpp
1>AcmeTank.cpp
1>Default.cpp
1>Dishwasher.cpp
1>Heater.cpp
1>Jet.cpp
1>MainEXE.cpp
1>Tank.cpp
1>Generating Code...
1>Compiling manifest to resources...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0
1>Copyright (C) Microsoft Corporation.  All rights reserved.
1>Linking...
1>LINK : warning LNK4067: ambiguous entry point; selected 'mainCRTStartup'
1>Embedding manifest...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0
1>Copyright (C) Microsoft Corporation.  All rights reserved.
1>Build log was saved at "file://f:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Debug\BuildLog.htm"
1>ProjectVSIDE - 0 error(s), 1 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

10. Now in the Visual Studio IDE click on the 'Start Debug' button.



```
/*********************************************************************
    Rhapsody     : 7.5.2
    Login        : Ashi
    Component    : EXE
    Configuration    : Host
    Model Element    : Host
//! Generated Date   : Wed, 30, Jun 2010
```

11. If needed, control the animation using the Rational Rhapsody GUI window.



f:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\Debug\ProjectVSIDE.exe

```
Dishwasher Cycle Started
     Filling
     Rinsing
     Washing
     Draining
     Drying
Dishwasher Cycle Complete
```

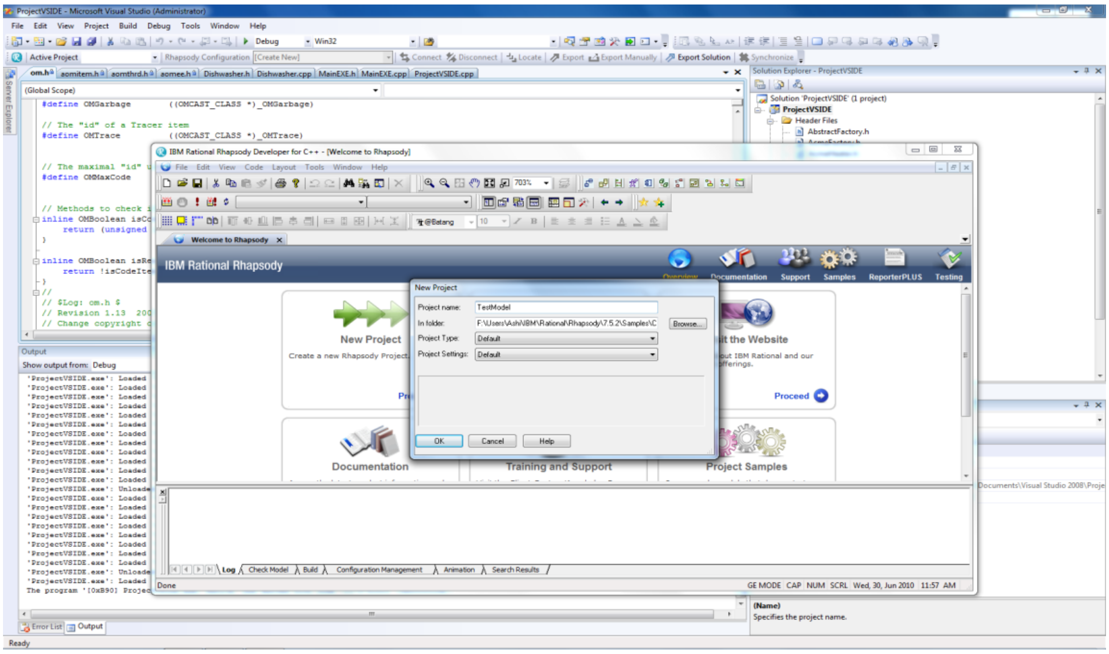
## *Creating Rational Rhapsody model from Visual Studio project*

Procedure:

1. Open the solution in Visual Studio.

2. Click the Rational Rhapsody icon in the Rational Rhapsody toolbar in Visual Studio.

3. Once focus is given to Rational Rhapsody, use the menu to create a Rational Rhapsody project.



New Project

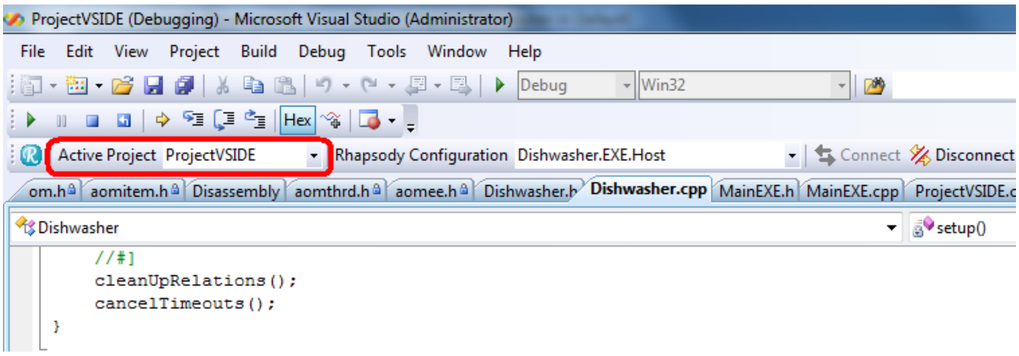
Project name: TestModel

In folder: F:\Users\Ash\IBM\Rational\Rhapsody\7.5.2\Samples\C

Project Type: Default

Project Settings: Default

4. In Visual Studio, use the Active Project list in the Rational Rhapsody toolbar to select the project that should be imported into Rational Rhapsody.

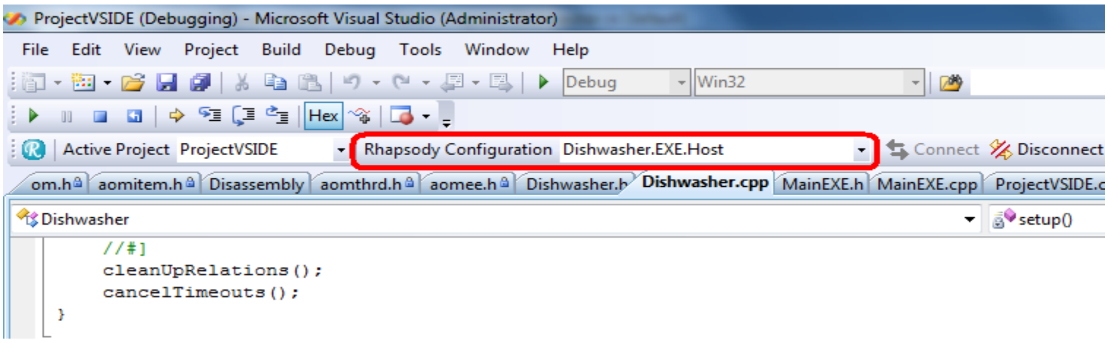


Active Project ProjectVSIDE

Rhapsody Configuration Dishwasher.EXE.Host

```
        //#]
        cleanUpRelations();
        cancelTimeouts();
}
```

5. In Visual Studio, use the Rhapsody Configuration list in the Rational Rhapsody toolbar to select the Rational Rhapsody configuration to which the Visual Studio project should be linked. Select an existing configuration or select the option of creating a new one.



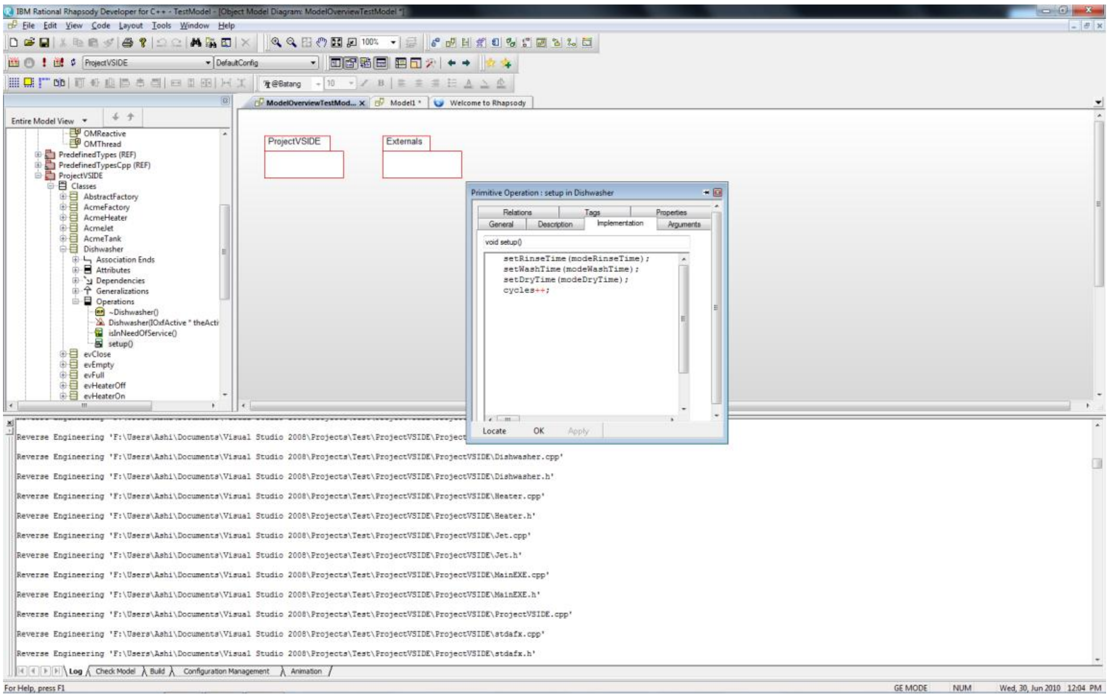
Active Project ProjectVSIDE

Rhapsody Configuration Dishwasher.EXE.Host

```
    //#]
    cleanUpRelations();
    cancelTimeouts();
}
```

6. Choose one of the two export options included in the Rational Rhapsody toolbar in Visual Studio in order to export the entire project or only selected classes. If the manual export option is selected, Rational Rhapsody Reverse Engineering window will be displayed, allowing you to select the relevant files.

7. Before the files are imported into the model, Rational Rhapsody will check if the files need to be imported, or should the Visual Studio project be further linked to the Rational Rhapsody model. If the second option is selected, all changes to the model will be reflected in the project code, and all changes to the project code will be round-tripped into the model.



void setup()

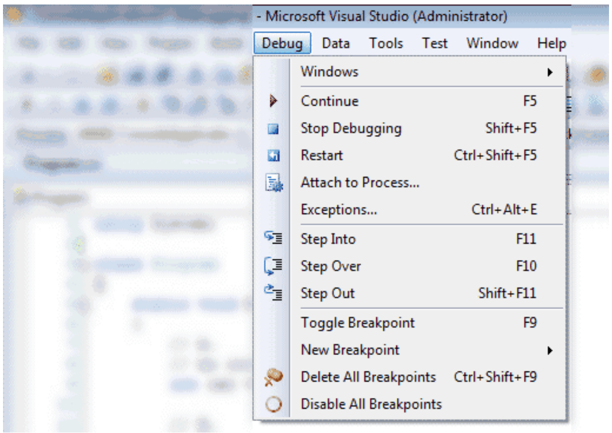
```
    setRinseTime(modeRinseTime);
    setWashTime(modeWashTime);
    setDryTime(modeDryTime);
    cycles++;
```

```
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\Project
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Dishwasher.cpp'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Dishwasher.h'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Heater.cpp'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Heater.h'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Jet.cpp'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\Jet.h'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\MainEXE.cpp'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\MainEXE.h'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\ProjectVSIDE.cpp'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\stdafx.cpp'
Reverse Engineering 'F:\Users\Ashi\Documents\Visual Studio 2008\Projects\Test\ProjectVSIDE\ProjectVSIDE\stdafx.h'
```

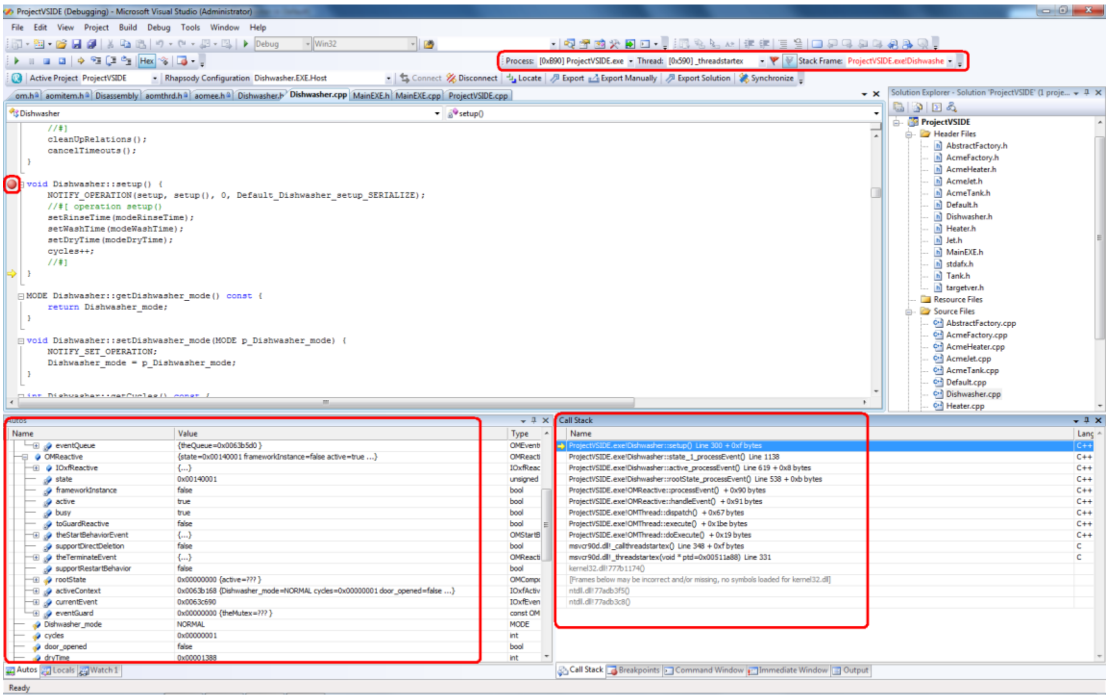
# Debugging with Visual Studio

An important advantage of the integration is that you can leverage the powerful debugging capabilities of the Visual Studio IDE.

Many times there is a need to see the values of the local variables in the program as it executes. This is useful when you want to ensure that the values are correct. Here you see an example of how Visual Studio 2008 integrated development environment can be used to debug your code.



The below screenshot shows a few of the useful debug features like Process, Thread, Stack Frame, Breakpoints, Call Stack, Autos, Locals, Watch, Command Window, Intermediate Window, Output.



```
        //#]
        cleanUpRelations();
        cancelTimeouts();
    }

void Dishwasher::setup() {
    NOTIFY_OPERATION(setup, setup(), 0, Default_Dishwasher_setup_SERIALIZE);
    //#[ operation setup()
    setRinseTime(modeRinseTime);
    setWashTime(modeWashTime);
    setDryTime(modeDryTime);
    cycles++;
    //#]
}

MODE Dishwasher::getDishwasher_mode() const {
    return Dishwasher_mode;
}

void Dishwasher::setDishwasher_mode(MODE p_Dishwasher_mode) {
    NOTIFY_SET_OPERATION;
    Dishwasher_mode = p_Dishwasher_mode;
}

int Dishwasher::getCycles() const {
```

Autos

| Name | Value | Type |
|---|---|---|
| eventQueue | {theQueue=0x0063b5d0 } | OMEventi |
| OMReactive | {state=0x00140001 frameworkInstance=false active=true ...} | OMReacti |
| IOxfReactive | {...} | IOxfReac |
| state | 0x00140001 | unsigned |
| frameworkInstance | false | bool |
| active | true | bool |
| busy | true | bool |
| toGuardReactive | false | bool |
| theStartBehaviorEvent | {...} | OMStartB |
| supportDirectDeletion | false | bool |
| theTerminateEvent | {...} | OMReacti |
| supportRestartBehavior | false | bool |
| rootState | 0x00000000 {active=??? } | OMCompo |
| activeContext | 0x0063b168 {Dishwasher_mode=NORMAL cycles=0x00000001 door_opened=false ...} | IOxfActiv |
| currentEvent | 0x0063c690 | IOxfEven |
| eventGuard | 0x00000000 {theMutex=??? } | const OM |
| Dishwasher_mode | NORMAL | MODE |
| cycles | 0x00000001 | int |
| door_opened | false | bool |
| dryTime | 0x00001388 | int |

Call Stack

| Name | Lang |
|---|---|
| ProjectVSIDE.exe!Dishwasher::setup() Line 300 + 0xf bytes | C++ |
| ProjectVSIDE.exe!Dishwasher::state_1_processEvent() Line 1138 | C++ |
| ProjectVSIDE.exe!Dishwasher::active_processEvent() Line 619 + 0x8 bytes | C++ |
| ProjectVSIDE.exe!Dishwasher::rootState_processEvent() Line 538 + 0xb bytes | C++ |
| ProjectVSIDE.exe!OMReactive::processEvent() + 0x90 bytes | C++ |
| ProjectVSIDE.exe!OMReactive::handleEvent() + 0x91 bytes | C++ |
| ProjectVSIDE.exe!OMThread::dispatch() + 0x67 bytes | C++ |
| ProjectVSIDE.exe!OMThread::execute() + 0x1be bytes | C++ |
| ProjectVSIDE.exe!OMThread::doExecute() + 0x19 bytes | C++ |
| msvcr90d.dll!_callthreadstartex() Line 348 + 0xf bytes | C |
| msvcr90d.dll!_threadstartex(void * ptd=0x00511a88) Line 331 | C |
| kernel32.dll!777b1174() | |
| [Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll] | |
| ntdll.dll!77adb3f5() | |
| ntdll.dll!77adb3c8() | |

# Conclusion

Both Rational Rhapsody and Microsoft Visual Studio 2008 Professional are independent tools and are widely used as per their strengths and capabilities in their respective domains. Despite being independent, each of them has quite a few features/functionalities that could be leveraged by the other to improve its functionality/capability. Hence, using this integration could be of great value to you especially if you have been (or are planning to) work with both the tools, taking the overall strength of both the tools to much higher levels.

## Notes and Warnings

- The integration is not intended for Visual Studio 2008 Express Edition.
- The integration creates a 'Visual Studio' folder in the Rational Rhapsody installation path which contains the 'RhapsodyAddIn' Visual Studio Add-in definition file. This file is responsible for the Integration functionalities.
- The integration also creates a Rational Rhapsody plug-in in the Visual Studio 2008 IDE. You can enable/disable this plug-in in the Visual Studio IDE, however this will affect the working of the integration.
- Ideally the feature 'Open Rhapsody' is intended to work for a valid Rational Rhapsody project pre-exists for the currently open Visual Studio project.
- The 'stdafx.h' is the file for pre-compiled header support. If you do not want to add it to each generated file (as in, if there is no need for it), you will have to disable it from the Visual Studio project's properties. By default this feature is enabled for all Visual Studio projects (C++).

## References

The following were used in references or as other sources of information:

- http://msdn.microsoft.com/hi-in/vstudio/default.aspx
- http://www.ibm.com/developerworks/rational/products/rhapsody/
- http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
- http://en.wikipedia.org/wiki/Integrated_development_environment