

IBM[®] Rational[®] Rhapsody[®] Gateway Add On



Code Coupling Notes

Rhapsody[®]

**IBM[®] Rational[®] Rhapsody[®]
Gateway Add On**

Code Coupling Notes



License Agreement

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of the copyright owner.

The information in this publication is subject to change without notice, and Dassault Systèmes and its affiliates assume no responsibility for any errors which may appear herein. No warranties, either expressed or implied, are made regarding Rhapsody software and its fitness for any particular purpose.

Trademarks

Reqtify is a registered trademark of Dassault Systèmes or its affiliates in the US and/or other countries.

Rhapsody Gateway, IBM, the IBM logo, DOORS and Rhapsody are trademarks or registered trademarks of IBM Corporation.

All other product or company names mentioned herein may be trademarks or registered trademarks of their respective owners.

© Copyright 2001-2011 Dassault Systèmes. All rights reserved.

Contents

Contents	5
Introduction	7
Code Type	9
Selecting the Code Directory	9
Specific Parameters of the Code Type	9
Type Customization	10
Example Analysis Results	11
Capturing Traceability Information	12
Code C Type	13
Selecting the Code Directory	13
Specific Parameters of the Code C Type	13
Type Customization	14
Example Analysis Results	15
Capturing Traceability Information	16
Code C++ Type	19
Selecting the Code Directory	19
Specific Parameters of the Code C++ Type	19
Type Customization	20
Example Analysis Results	20
Capturing Traceability Information	21
Code C# Type	25
Selecting the Code Directory	25
Specific Parameters of the Code C# Type	25
Type Customization	26
Example Analysis Results	26
Capturing Traceability Information	27
ADA Code Type	31
Selecting the Code Directory	31
Specific Parameters of the Ada Code Type	31
Type Customization	32
Example Analysis Results	32
Capturing Traceability Information	33
Large Code Type of Analysis	37

Contents

Selecting the Large Code Directory.....	37
Type Customization	38
Configuring a Large Code Document	39
Capturing Traceability Information	40
Navigating to Code Files.....	41

Introduction


This technical note describes how Rhapsody Gateway analyzes code files using the **Code**, **Code C**, **Code C++**, **Ada Code** and **Large Code** types.

- ◆ The **Code** type is used to browse directories and files. It performs a default text analysis of the files.
- ◆ The **Code C** type is used to browse C directories and files. It performs a syntactic analysis of the files, and highlights C functions.
- ◆ The **Code C++** type is used to browse C++ directories and files. It performs a syntactic analysis of the files, and highlights C++ functions.
- ◆ The **Ada Code** type is used to browse Ada directories and files. It performs a syntactic analysis of the files, and highlights Ada packages and subprograms.
- ◆ The **Large Code** converter is dedicated to analysis of large number of files. The user defines additional conditions on file content to indicate if the file has to be analyzed or not.

Some standard operations, such as navigating in a code file from the Rhapsody Gateway tool, require a text editor to be installed.

Code Type

Selecting the Code Directory

In the Project Editor, insert a document, select a type of analysis based on **Code**, click on the **File or Directory** field then on the **Browse**  button.

Rhapsody Gateway displays a selection window. Select a Directory and click **OK**.

Rhapsody Gateway will analyze all code files found in the selected directory. By default the Code type also analyzes all code files in subdirectories of the selected directory.

Specific Parameters of the Code Type

Rhapsody Gateway can automatically check file dates and suggests to update the analysis results. Depending on the number of files and the network configuration, this verification can take a few seconds. The user may prefer the manual update instead.

Use the **Prompt when files change** variable to indicate if you prefer the automatic verification of file dates (activate the option) or manual update (deactivate the option).

Details								Modification Files		Covers	
Name		Type of Analysis	File or Directory	Ignor...	Inter...	Bloc...	Variable	Value			
Code Files		Code	Code\VC Sample	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Prompt when files change	<input type="checkbox"/>			

Type Customization

Define your own type by duplicating the default Code type of analysis.

By default the Code type analyzes files with the .c, .h, and .cpp extensions.

Use the **Filters** field of the **Advanced Options** pane to specify the list of file extensions. The extensions in the list must be separated by a comma or a semicolon character. You can use any extension, not only the ones corresponding to C or C++ files.

In addition, use the **Include sub-directories** checkbox to specify whether the type analyzes files from the sub-directories of the directory selected from the Project Editor.

The screenshot shows a dialog box for customizing a code type. It has a 'Name' field with 'MyCode' and an 'Icon' button. Below is a 'Display' field. There are two tabs: 'Analysis' and 'Advanced Options', with 'Advanced Options' selected. Under 'Advanced Options', there is a checked checkbox for 'Merge homonymous sections'. Below this are three text fields: 'Filters' (containing '*.c;*.h;*.cpp'), 'Excluded files', and 'Constant file'. There is another checked checkbox for 'Include sub-directories'. Below that are three more text fields: 'Expression to delete an element', 'Sub expression to delete an element', and 'Expressions to exclude text'. At the bottom is a 'Check CRC' checkbox which is unchecked. Each text field has a small icon button to its right.

Use the **Excluded files** field to specify the files to be ignored.

The other field are general ones, documented in the *Customization Guide*.

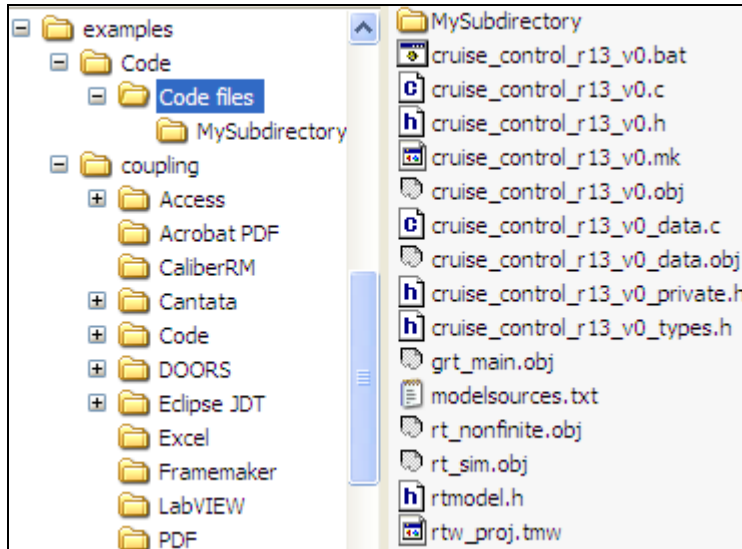
Note

You can restrict which files the type analyzes. In order to specify portions of the base name that the file must match. For instance, to analyze C files with names starting with `build_`, enter `*build_*.c` in the **Filter** field. Do not specify `build_*.c` because Rhapsody Gateway will consider the full path.

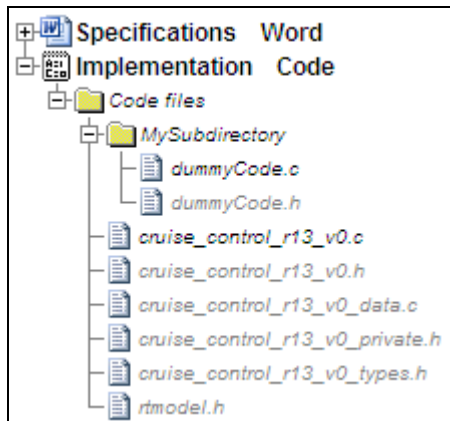
Example Analysis Results

The following example illustrates how the Code type will process files from a selected directory and its subdirectories.

Files on the disk



Analysis results displayed in the Rhapsody Gateway project workspace



Capturing Traceability Information

You can add traceability information directly to a code file by inserting text into the file. The text must adhere to the syntax defined by the **reference** element of the type. By default the reference element of the Code type recognizes C/C++ style comments within files as shown below.


The default formalism:

```
// Implements <requirementID>
```

Code C Type

The **Code C** type is used for C code analysis. In addition to the features presented for the Code type, the Code C type performs syntax analysis and lists the functions within files and their contents, allowing you to specify more accurate traceability within source files.

Selecting the Code Directory

In the Project Editor, insert a document, select a type of analysis based on **C Code**, click on the **File or Directory** field then on the **Browse**  button.



Rhapsody Gateway displays a selection window. Select a Directory and click **OK**.

Rhapsody Gateway will analyze all code files found in the selected directory. By default the **Code C** type also analyzes all code files in subdirectories of the selected directory.

Specific Parameters of the Code C Type

Rhapsody Gateway can automatically check file dates and suggests to update the analysis results. Depending on the number of files and the network configuration, this verification can take a few seconds. The user may prefer the manual update instead.

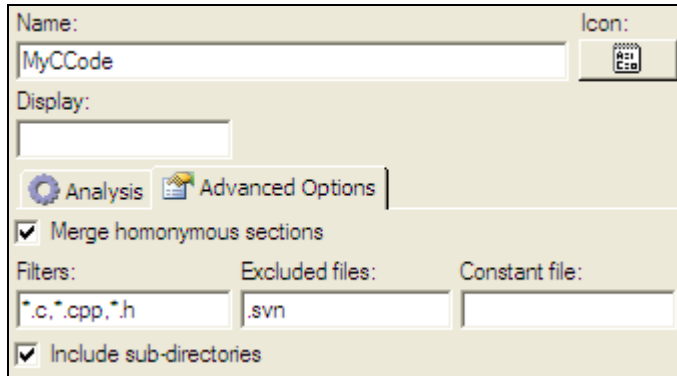
Use the **Prompt when files change** variable to indicate if you prefer the automatic verification of file dates (activate the option) or manual update (deactivate the option).

Details Modification Files Covers								
	Name	Type of Analysis	File or Directory	Ignor...	Inter...	Bloc...	Variable	Value
	Code Files	 Code C	Code\VC Sample	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Prompt when files change	<input type="checkbox"/>

Use the **Includes function body** variable to insert function bodies in the intermediate file, in order to use their contents for example in report, snapshots, etc.

Type Customization

Define your own type by duplicating the default **Code C** type of analysis.



The screenshot shows a dialog box for customizing the Code C type. It has a 'Name' field with 'MyCCode' and an 'Icon' field with a small icon. Below these is a 'Display' field. There are two tabs: 'Analysis' (selected) and 'Advanced Options'. Under the 'Analysis' tab, there is a checked checkbox for 'Merge homonymous sections'. Below this are three fields: 'Filters' (containing '*.c;*.cpp;*.h'), 'Excluded files' (containing '*.svn'), and 'Constant file' (empty). At the bottom, there is a checked checkbox for 'Include sub-directories'.

By default the **Code C** type analyzes files with the `.c`, `.h`, and `.cpp` extensions.

Use the **Filters** field of the **Advanced Options** pane to specify the list of file extensions. The extensions in the list must be separated by a comma or a semicolon character.

In addition, use the **Include sub-directories** checkbox to specify whether the type analyzes files from the sub-directories of the directory selected from the Project Editor.

Use the **Excluded files** field to specify the files to be ignored.

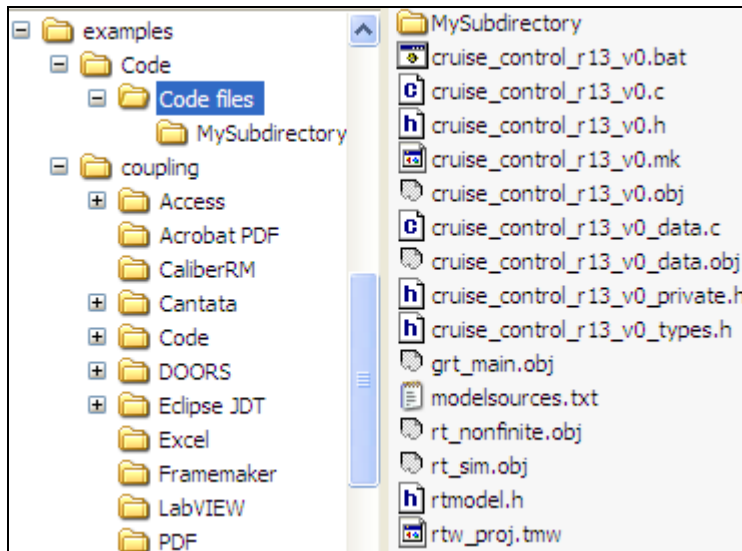
Note

You can restrict which files the type analyzes. In order to specify portions of the base name that the file must match. For instance, to analyze C files with names starting with `build_`, enter `*build_*.c` in the **Filter** field list. Do not specify `build_*.c`, because Rhapsody Gateway will consider the full path.

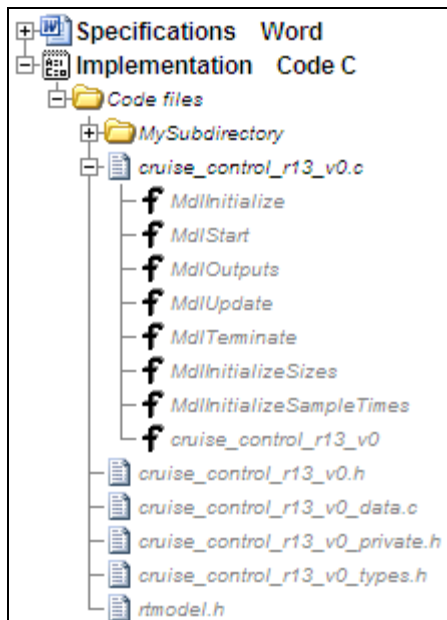
Example Analysis Results

The following example illustrates how the **Code C** type will process the files from a selected directory and its subdirectories.

Files on the disk



Analysis results displayed in the Rhapsody Gateway project workspace



Capturing Traceability Information

You can add traceability information directly to a code file by inserting text into the file. The **Code C** type processes C style comments within a file and generates elements in an XML intermediate file that the **Code C** type analyzes.

The default formalisms are:

```
// Implements <requirementID>
and
/* Implements <requirementID> */
```

The **Code C** type analyzes the comments associated with a file and the functions in the file. The following examples illustrate how the Code C type determines whether to associate traceability information with the file or with a function in the file.

Example: The comment is associated with the function it precedes if no blank line exists between the function and the comment.

```
int function1(void)
{
    return 1;
}

//Implements REQ_GLV_20
int function2(void)
{
    return 1;
}
```

Result: The reference is associated with function2.

Example: The comment is associated with the file if a blank line exists between the function and the comment.

```
int function1(void)
{
    return 1;
}

//Implements REQ_GLV_20

int function2(void)
{
    return 1;
}
```

Result: The reference is associated with the file instead of function2.

Example: The comment is associated with the function which it is located.

```
int function2(void)
{
    //Implements REQ_GLV_20
    return 1;
}
```

Result: The reference is associated with function2.

If you use a C style comment block (/*...*/) to associate a reference with a function, the comment block must be located immediately before or in the function declaration.

Example: Several comment blocks, with one comment block inserted between the comment block with the reference and function2.

```
/* ----- */
/* Implements REQ_GLV_20 */
/* ----- */
int function2(void)
{
    return 1;
}
```

Result: The reference is associated with function2.

If you place a blank line or another kind of comment between the function and the comment block that contains the reference, the type will associate the reference with the file instead of the function as shown below.

Example: Several comment blocks, with one comment block inserted between the comment block with the reference and function2.


```
/* ----- */
/* Implements REQ_GLV_20 */
//-----
int function2(void)
{
    return 1;
}
```

Result: The reference is associated with the file instead of function2.

Code C++ Type

The **Code C++** type is used for C++ code analysis. In addition to the features presented for the Code type, the **Code C++** type performs syntax analysis and lists the functions within files and their contents, allowing you to specify more accurate traceability within source files.

Selecting the Code Directory

In the Project Editor, insert a document, select a type of analysis based on **C++ Code**, click on the **File or Directory** field then on the **Browse**  button.



Rhapsody Gateway displays a selection window. Select a Directory and click **OK**.

Rhapsody Gateway will analyze all code files found in the selected directory. By default the **Code C++** type also analyzes all code files in subdirectories of the selected directory.

Specific Parameters of the Code C++ Type

Rhapsody Gateway can automatically check file dates and suggests to update the analysis results. Depending on the number of files and the network configuration, this verification can take a few seconds. The user may prefer the manual update instead.

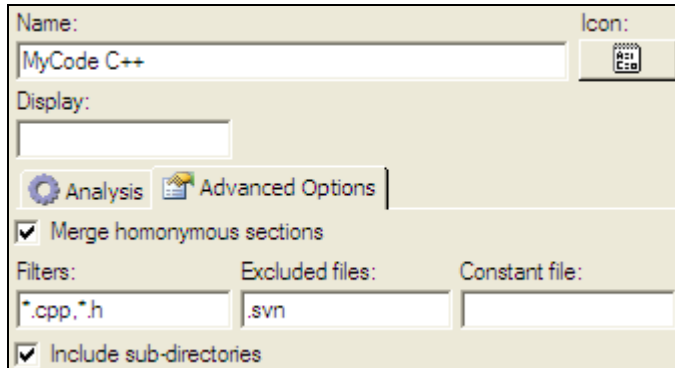
Use the **Prompt when files change** variable to indicate if you prefer the automatic verification of file dates (activate the option) or manual update (deactivate the option).

Details Modification Files Covers							
Name	Type of Analysis	File or Directory	Ignor...	Inter...	Bloc...	Variable	Value
 Code Files	 Code C++	Sources	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Prompt when files change	<input type="checkbox"/>

Use the **Includes function body** variable to insert function bodies in the intermediate file, in order to use their contents for example in report, snapshots, etc.

Type Customization

You can define your own type by duplicating the default **Code C++** type of analysis.



By default, the **Code C++** type analyzes files with the `.cpp` and `.h` extensions.

Use the **Filters** field of the **Advanced Options** pane to specify the list of file extensions. The extensions in the list must be separated by a comma or a semicolon character.

In addition, use the **Include sub-directories** checkbox to specify whether the type analyzes files from the sub-directories of the directory selected from the Project Editor.

Use the **Excluded files** field to specify the files you want to ignore.

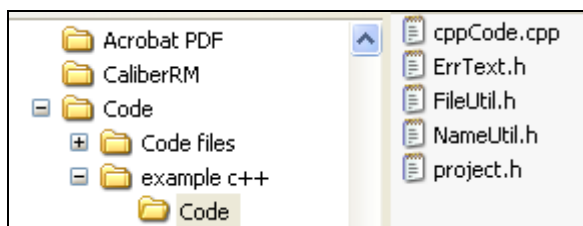
Note

You can restrict which files the type analyzes, in order to specify portions of the base name that the file must match. For example, if you want to only analyze C++ files with names that start with `build_`, enter `*build_*.cpp` in the **Filter** field list. Do not specify `build_*.cpp` only, because Rhapsody Gateway will consider the full path.

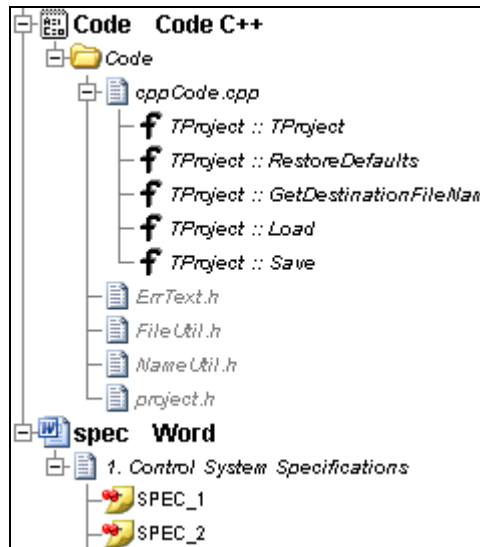
Example Analysis Results

The following example illustrates how the **Code C++** type will process the files from a selected directory and its subdirectories.

Files on the disk



Analysis results displayed in the Rhapsody Gateway project workspace



Capturing Traceability Information

You can add traceability information directly to a code file by inserting text into the file. The **Code C++** type processes C++ style comments within a file and generates elements in an XML intermediate file that the **Code C++** type analyzes.

The default formalisms are:

```
// Implements <requirementID>  
and  
/* Implements <requirementID> */
```

The **Code C++** type analyzes the comments associated with a file and the functions in the file. The following examples illustrate how the **Code C++** type determines whether to associate traceability information with the file or with a function in the file.

Example: The comment is associated with the function it precedes, if no blank line exists between the function and the comment.

```
int TProject::Function1(void)
{
    return 1;
}
//Implements REQ_1
int TProject::Function2(void)
{
    return 1;
}
```

Result: The reference is associated with TProject::Function2.

Example: The comment is associated with the file if a blank line exists between the function and the comment.

```
int TProject::Function1(void)
{
    return 1;
}
//Implements REQ_1

int TProject::Function2(void)
{
    return 1;
}
```

Result: The reference is associated with the file instead of TProject::Function2.

Example: The comment is associated with the function which it is located.

```
int TProject::Function2(void)
{
    //Implements REQ_1
    return 1;
}
```

Result: The reference is associated with TProject::Function2.

If you use a C++ style comment block (`/*...*/`) to associate a reference with a function, the comment block must be located immediately before or in the function declaration.

Example: Several comment blocks, with one comment block inserted between the comment block with the reference and TProject::Function2.

```
/* ----- */
/* Implements REQ_1 */
/* ----- */
int TProject::Function2(void)
{
    return 1;
}
```

Result: The reference is associated with TProject::Function2.

If you place a blank line or another kind of comment between the function and the comment block that contains the reference, the type will associate the reference with the file instead of the function as shown below.

Example: Several comment blocks, with one comment block inserted between the comment block with the reference and TProject::Function2.


```
/* ----- */
/* Implements REQ_GLV_20 */
//-----
int Tproject::Function2(void)
{
    return 1;
}
```

Result: The reference is associated with the file instead of TProject::Function2.

Code C# Type

The **Code C#** type is used for C# code analysis. In addition to the features presented for the Code type, the **Code C#** type performs syntax analysis and lists the functions within files and their contents, allowing you to specify more accurate traceability within source files.

Selecting the Code Directory

In the Project Editor, insert a document, select a type of analysis based on **C# Code**, click on the **File or Directory** field then on the **Browse**  button.

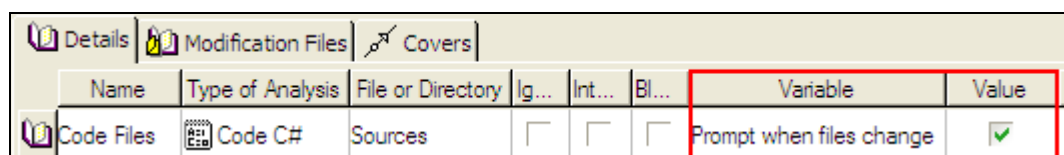
Rhapsody Gateway displays a selection window. Select a Directory and click **OK**.



Rhapsody Gateway will analyze all code files found in the selected directory. By default the **Code C#** type also analyzes all code files in subdirectories of the selected directory.

Specific Parameters of the Code C# Type

Rhapsody Gateway can automatically check file dates and suggests to update the analysis results. Depending on the number of files and the network configuration, this verification can take a few seconds. The user may prefer the manual update instead.

Use the **Prompt when files change** variable to indicate if you prefer the automatic verification of file dates (activate the option) or manual update (deactivate the option).



Details Modification Files Covers							
Name	Type of Analysis	File or Directory	Ig...	Int...	Bl...	Variable	Value
 Code Files	 Code C#	Sources	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Prompt when files change	<input checked="" type="checkbox"/>

Use the **Includes function body** variable to insert function bodies in the intermediate file, in order to use their contents for example in report, snapshots, etc.

Type Customization

You can define your own type by duplicating the default **Code C#** type of analysis.

The screenshot shows a configuration window for a new analysis type named 'My Code C#'. It has tabs for 'Analysis' and 'Advanced Options', with 'Advanced Options' currently active. Under 'Advanced Options', the 'Merge homonymous sections' checkbox is checked. The 'Filters' field is set to '*.cs', 'Excluded files' is set to '*.svn', and 'Include sub-directories' is checked. The 'Constant file' field is empty.

By default, the **Code C#** type analyzes files with the `.cs` extension.

Use the **Filters** field of the **Advanced Options** pane to specify the list of file extensions. The extensions in the list must be separated by a comma or a semicolon character.

In addition, use the **Include sub-directories** checkbox to specify whether the type analyzes files from the sub-directories of the directory selected from the Project Editor.

Use the **Excluded files** field to specify the files you want to ignore.

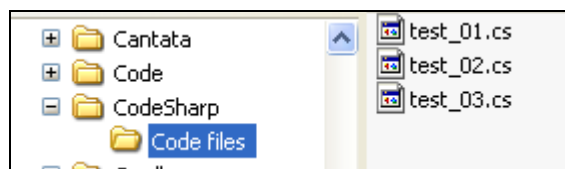
Note

You can restrict which files the type analyzes, in order to specify portions of the base name that the file must match. For example, if you want to only analyze C# files with names that start with `build_`, enter `*build_*.cs` in the **Filter** field list. Do not specify `build_*.cs`, because Rhapsody Gateway will consider the full path.

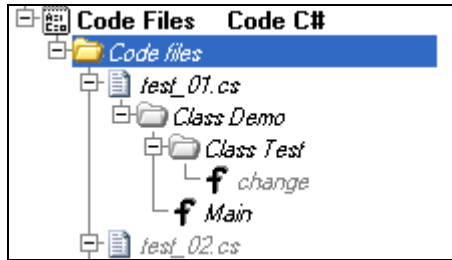
Example Analysis Results

The following example illustrates how the **Code C#** type will process the files from a selected directory and its subdirectories.

Files on the disk



Analysis results displayed in the Rhapsody Gateway project workspace



Capturing Traceability Information

You can add traceability information directly to a code file by inserting text into the file. The **Code C#** type processes C# style comments within a file and generates elements in an XML intermediate file that the **Code C#** type analyzes.

The default formalisms are:

```
// Implements <requirementID>  
and  
/* Implements <requirementID> */
```

The **Code C#** type analyzes the comments associated with a file and the functions in the file. The following examples illustrate how the **Code C#** type determines whether to associate traceability information with the file or with a function in the file.

Example: The comment is associated with the function it precedes, if no blank line exists between the function and the comment.

```
public class Test  
{  
    public void Function1 ()  
    {  
        return;  
    }  
    //Implements REQ_1  
    public void Function2 ()  
    {  
        return;  
    }  
}
```

Result: The reference is associated with Function2.

Example: The comment is associated with the file if a blank line exists between the function and the comment.

```
public class Test
{
    public void Function1 ()
    {
        return;
    }
    //Implements REQ_1

    public void Function2 ()
    {
        return;
    }
}
```

Result: The reference is associated with the file instead of Function2.

Example: The comment is associated with the function which it is located.

```
public void Function2 ()
{
    //Implements REQ_1
    return;
}
```

Result: The reference is associated with Function2.

If you use a C++ style comment block (/*...*/) to associate a reference with a function, the comment block must be located immediately before or in the function declaration.

Example: Several comment blocks, with one comment block inserted between the comment block with the reference and Function2.

```
/* ----- */
/* Implements REQ_1 */
/* ----- */
public void Function2 ()
{
    return;
}
```

Result: The reference is associated with Function2.

If you place a blank line or another kind of comment between the function and the comment block that contains the reference, the type will associate the reference with the file instead of the function as shown below.

Example: Several comment blocks, with one comment block inserted between the comment block with the reference and Function2.


```
/* ----- */
/* Implements REQ_GLV_20 */
//-----
public void Function2 ()
{
    return;
}
```

Result: The reference is associated with the file instead of Function2.

ADA Code Type

The **Ada Code** type is used for Ada code analysis. In addition to the features presented for the Code type, the Ada Code type performs syntax analysis and lists the sub-programs within files and their contents, allowing you to specify more accurate traceability within source files.

Selecting the Code Directory

In the Project Editor, insert a document, select a type of analysis based on **Ada Code**, click on the **File or Directory** field then on the **Browse**  button.

Rhapsody Gateway displays a selection window. Select a Directory and click **OK**.

Rhapsody Gateway will analyze all Ada code files found in the selected directory. By default the **Ada Code** type also analyzes all ada and adb code files in subdirectories of the selected directory.



Specific Parameters of the Ada Code Type

Rhapsody Gateway can automatically check file dates and suggests to update the analysis results. Depending on the number of files and the network configuration, this verification can take a few seconds. The user may prefer the manual update instead.

Use the **Test files date** variable to indicate if you prefer the automatic verification of file dates (activate the option) or manual update (deactivate the option).

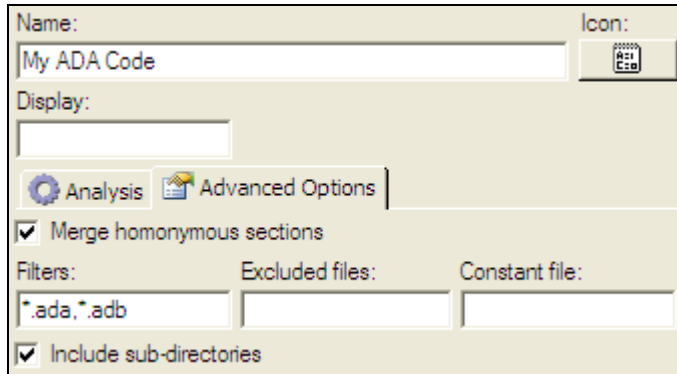
Details Modification Files Covers								
	Name	Type of Analysis	File or Directory	Ignor...	Inter...	Bloc...	Variable	Value
	AdaSource	 ADA Code	Sources	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Test files date	<input type="checkbox"/>

Use the **Includes subprogram body** variable to indicate if the bodies of the sub-programs have to be exported in the same time.

Details Modification Files Covers								
	Name	Type of Analysis	File or Directory	Ignor...	Inter...	Bloc...	Variable	Value
	AdaSource	 ADA Code	Sources	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	includes subprogram body	<input type="checkbox"/>

Type Customization

You can define your own type by duplicating the default **Ada Code** type of analysis.



By default the **Ada Code** type analyzes files with the `.ada` and `.adb` extensions.

Use the **Filters** field of the **Advanced Options** pane to specify the list of file extensions. The extensions in the list must be separated by a comma or a semicolon character.

In addition, use the **Include sub-directories** checkbox to specify whether the type analyzes files from the sub-directories of the directory selected from the Project Editor.

Use the **Excluded files** field to specify the files you want to ignore.

Note

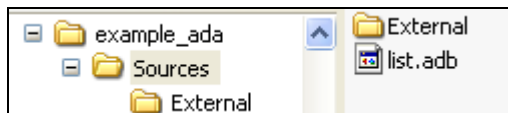
You can restrict which files the type analyzes, in order to specify portions of the base name that the file must match. For example, if you want to only analyze Ada files with names that start with `build_`, enter `*build_*.ada` in the Filter field list. Do not specify `build_*.ada` only, because Rhapsody Gateway will consider the full path.

Example Analysis Results

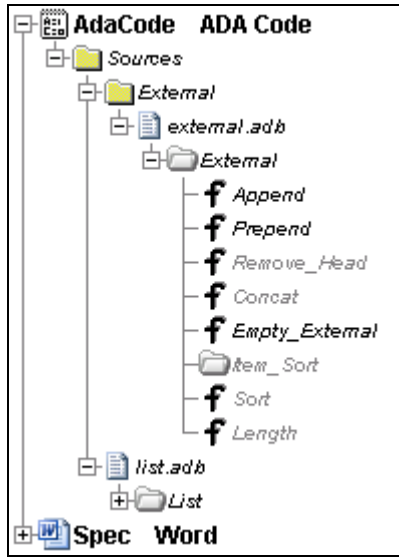
The following example illustrates how the **Ada Code** type will process the files from a selected directory and its subdirectories.

Files on the disk

Directories which contain code files:



Analysis results displayed in the Rhapsody Gateway project workspace



Capturing Traceability Information

You can add traceability information directly to a code file by inserting text into the file. The **Ada Code** type processes Ada style comments within a file and generates elements in an XML intermediate file that the **Ada Code** type analyzes.

The default formalism is:

```
-- Implements: <requirementID>
```

The **Ada Code** type analyzes the comments associated with a file, a package of a file and the sub-programs in the package. The following examples illustrate how the Ada Code type determines whether to associate traceability information with the file, the package or with a sub-program.

Example: The comment is associated with the sub-program it precedes.

```
procedure proc is
begin
    a:=10;
end proc;
-- Implements: REQ_1
function func return Integer is
begin
    return 1;
end func;
```

Result: The reference is associated with function func.

Example: A comment can be located in the sub-program declaration.

```
procedure proc is
begin
  -- Implements: REQ_1
  a:=10;
end proc;
```

Result: The reference is associated with procedure proc.

Example: If the comment is defined before the package declaration it will be associated with this package.

```
-- Implements: REQ_1
package body List is
...
End List;
```

Result: The reference is associated with the package List.

Example: The comment is added at the end of the file.

Result: The comment will be associated with the file itself.

Extract of an Ada code:

```
-----
-- Procedure : Empty_List :
-----
-- Implements: Spec_4
procedure Empty_List(Lst : in out List) is
begin
  Finalize(Lst.The_List);
end Empty_List;
```

Extract of corresponding intermediate XML file:

```


...
<PROCEDURE_BODY name="List/Empty_List(List)"
localname="Empty_List(List)">
  <COMMENT>-----
  -----</COMMENT>
  <COMMENT>Procedure : Empty_List
:</COMMENT>
  <COMMENT>-----
  -----</COMMENT>
  <COMMENT>Implements: Spec_4</COMMENT>
</PROCEDURE_BODY>
...

```


Large Code Type of Analysis

The **Large Code** type is designed to analyze large numbers of code files, potentially several thousand. The type is optimized and applies a filter to determine which files to analyze.

Selecting the Large Code Directory

In the Project Editor, insert a document, select a type of analysis based on **Large Code**, click on the **File or Directory** field then on the **Browse**  button.

Rhapsody Gateway displays a selection window. Select a Directory and click **OK**.

Rhapsody Gateway will analyze all code files found in the selected directory. By default the **Large Code** type also analyzes all code files in subdirectories of the selected directory.

Type Customization

You can define your own type by duplicating the default **Large Code** type of analysis.

By default the **Large Code** type analyzes files with the `.c`, `.h`, and `.cpp` extensions.

Use the **Filters** field of the **Advanced Options** pane to specify the list of file extensions. The extensions in the list must be separated by a comma or a semicolon character.

In addition, use the **Include sub-directories** checkbox to specify whether the type analyzes files from the sub-directories of the directory selected from the Project Editor.

Use the **Excluded files** field to specify the files you want to ignore.

The other fields are general ones, documented in the *Customization Guide*.

Note

You can restrict which files the type analyzes by specifying portions of the base name that the file must match. For example, if you want to only analyze C files with names that start with `build_`, enter `*build_*.c` in the **Filter** field list. Do not specify `build_*.c` only, because Rhapsody Gateway will consider the full path.

Configuring a Large Code Document

Rhapsody Gateway can automatically check file dates and propose to update the analysis results. This feature is available for **Code** and **Code C** types, as explained in the previous sections of this *Coupling Note*, but not for the **Large Code** type of analysis. The assumption is that the number of files to be analyzed is very large, and such a check would be too long. For a document based on a **Large Code** type, you need to manually re-analyze the files.

The only variable you have to define for a **Large Code** type is the **Filter Expression** variable.

Details Modification Files Covers							
Name	Type of Analysis	File or Directory	Ignor...	Inter...	Bloc...	Variable	Value
Code Files	Large Code	Code\C Sample	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Filter Expression	(// Implements .+)\$

The **Filter Expression** specifies a regular expression that the type uses to determine if traceability information exists within a document. The type performs the following analysis steps:

- The type scans the selected directory for the document to locate files that match the file extensions listed in the **Filters** field of the **Advanced Options** dialog box (by default, .c, .h, and .cpp files).

- The type analyzes the files using the **Filter Expression** to determine if the file contains traceability information. If no traceability information is found, the type ignores the file. If traceability information is found, the type analyzes the file and displays the file in the main window. The type retains only the sections of the files that contain traceability information.

- The type analyzes the retained traceability information.

Refer to the **Capturing Traceability Information** section below for more information.

Note

The **Filter Expression** that you define for the document in the Project Editor applies to only the files associated with the document. Other documents in the Project Editor can use the same **Large Code** type, and each document can specify a unique **Filter Expression** value.

Capturing Traceability Information

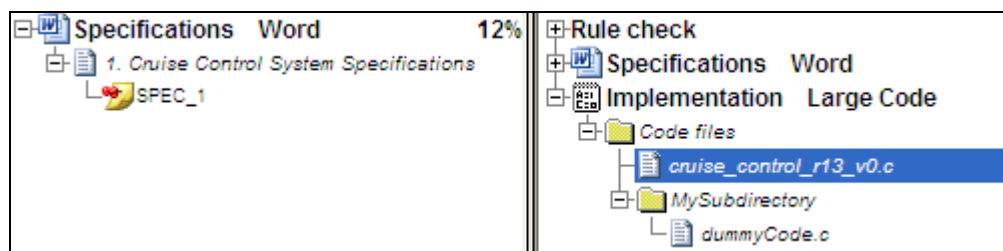
You can add traceability information directly to a code file by inserting text into the file. The text must adhere to the syntax defined by the **reference** element of the type. By default the reference element of the **Large Code** type recognizes C++ style comments within files as shown below.

```
// Implements <requirementID>
```

For the **Large Code** type to perform properly, the **Filter Expression** variable must locate text in files that adheres to the format defined by the **Reference** element for the type.

The following example illustrates how the type processes files:

File Content	<pre>//Implements SPEC_1 int function2(void) { return 1; }</pre>
Filter Expression (defined for a document) (// Implements.+)\$	Conversion will provide: <File path and name> <pre>//Implements SPEC_1</pre>
Reference (defined for a Type) // Implements\s+(.+) \$	Classical analysis of: <pre>//Implements SPEC_1</pre>
Analysis result	SPEC_1



The following is an example of an incorrect Filter expression and Reference:

Filter expression: (Implements) ➔ Result = Implements

Reference: // Implements\s+(.+) \$ ➔ No traceability information is detected!

Navigating to Code Files

By default when you perform a navigation operation on a file for a document that uses the **Code**, **Code C**, **Code C++**, **Ada Code** or **Large Code** type, Rhapsody Gateway opens the file within the application that is associated with the extension of the file in your system. You can also select a specific application from a list when you customize the type, by selecting the application in the **Edit tool** field of the Type Editor as shown below. The default edit tool selection is Code.

