

IBM® Rational® Rhapsody® Automatic Test Generation Add On



Frequently Asked Questions

Rhapsody®

**IBM® Rational® Rhapsody®
Automatic Test Generation
Add On**

Frequently Asked Questions

Release 3.6



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, BTC Embedded Systems AG.

The information in this publication is subject to change without notice, and BTC Embedded Systems AG assumes 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

IBM® Rational® Rhapsody®, IBM® Rational® Rhapsody® Automatic Test Generation Add On, and IBM® Rational® Rhapsody® TestConductor Add On are registered trademarks of IBM Corporation.

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

© Copyright 2000-2009 BTC Embedded Systems AG. All rights reserved.

Contents

1	Preface.....	6
1.1	Contacting IBM® Rational® Software Support.....	6
2	Overview.....	7
3	Frequently Asked Questions.....	7
3.1	What... “is the minimal configuration (processor, memory, ...) to use Rhapsody with ATG ?	7
3.2	What If... “RhapsodyATG Browser Section Is Empty?”	7
3.3	What If... “Stereotype atgComponent is not visible in the Feature Dialog of Components?”	7
3.4	What If... “ATG Complains About Component Settings?”	7
3.5	What If... “ATG Test Case Generation Complains That It Can Not Compile?”	7
3.6	What is... “the meaning of ATG > File > Settings?”	8
3.6.1	Enable Warnings for non-animated Model elements	8
3.6.2	Regard only stimuli when exporting to SD	8
3.6.3	Remap stimuli from Env when exporting to SD	8
3.6.4	Use TestContext instead of Env when exporting to SD	8
3.6.5	Add return values when exporting to SD	9
3.6.6	Skip startup messages when exporting to SD	9
3.6.7	Use object names in arguments when exporting to SD.....	9
3.6.8	Compute test cases with lower thread priority	9
3.6.9	Use original framework.....	9
3.6.10	Discard test cases with more than N messages	9
3.7	What If... “ATG Test Case Generation Complains About Missing Files”	9
3.8	What If... “ATG Does Not Generate Any Test Cases?”	10
3.9	What If... “ATG Does Not Cover All Test Goals?”	10
3.10	What If... “Not All Relevant Messages Are Shown In the Test Cases?”	10
3.11	What If... “I Want to Have Not Only the First Instance as Interface Object?”	10
3.12	What... “Is the Meaning of ATG > General > Use ATG Compilation Environment?”	10
3.13	What... “If I added an Element to the Model, but It Is Not Visible in ATG?”	11
3.14	What... “Is The Relation Between ATG Testing Component and ATG Test Generation Configuration?”	11
3.15	What... “Is the Meaning of the ATG Check-Box <i>Use Model Interface Specification</i> ?”	11
3.16	“Does ATG exploit both Statechart diagrams (attached to classes) and activity diagrams (attached to operations, for describing methods)?”	11
3.17	“Does ATG require an event-driven, reactive style of architecture? Or can it be used on e.g. systems without statecharts?”	12
3.18	What... “If a method is not modeled, but its C++ code is provided?”	12
3.19	What... “kind of coverage can be achieved on non modeled methods?	12
3.20	What... “needs to be modeled in UML, and what can be provided directly as C++ code?	12
3.21	“Are the activity states of the Activity Diagrams attached to operations also included in the class instance test goals?	12
3.22	When... “specifying events as test goals: is the goal the generation or the consumption (reception) of events?	12
3.23	What... “is the value of executing test cases with TestConductor?	12
3.24	“Is it possible to use ATG on a sub-part of a model that has dependencies on for instance operations that have not been modeled yet, and need to be stubbed?	13
3.25	“What is the meaning of Test Generation Strategy Breadth-First and Beam...?”	13
3.26	“What is the meaning of <i>Handling of Timeouts</i> Standard and Timestep...?”	13
3.27	“Why shows the ATG browser covered operations of a class although I have selected specific instances to be covered...?”	13
3.28	“Which syntax is supported to specify values of arguments in the ATG interface tab...?”	14
3.29	“Why using OM_RETURN() and OM_RETURN_VOID()...?”	14
3.30	“What about differences in the Rhapsody SD animation and test generation with ATG...?”	14

1 Preface

Welcome to the Frequently Asked Questions (FAQ) document for IBM® Rational® Rhapsody® Automatic Test Generation Add On (Rhapsody ATG). Rhapsody ATG is a test case generation tool using standard Unified Modeling Language™ (UML™) design notations. Using ATG, you can automatically generate test suites and perform test execution for your applications developed with the Rhapsody in C++ design tool at any stage in your development cycle.

The typical UML development process (such as the Rapid Object-Oriented Process for Embedded Systems (ROPES)) is iterative, starting with an early, fairly abstract version and progressing to more and more concrete prototypes. To test a System Under Test, use ATG in your development process to do unit testing, integration testing, or regression testing.

Rhapsody ATG is complemented by Rhapsody® TestConductor. TestConductor automatically generates test monitors and test drivers from Rhapsody sequence diagrams (SDs). During automated test execution, the generated monitors determine whether the executed model satisfies the selected SDs. ATG generates test cases that can be exported to UML sequence diagrams in order to execute test cases with TestConductor.

1.1 Contacting IBM® Rational® Software Support

IBM Rational Software Support provides you with technical assistance. The IBM Rational Software Support Home page for Rational products can be found at <http://www.ibm.com/software/rational/support/>.

For contact information and guidelines or reference materials that you need for support, read the [IBM Software Support Handbook](#).

For Rational software product news, events, and other information, visit the [IBM Rational Software Web site](#).

Voice support is available to all current contract holders by dialing a telephone number in your country (where available). For specific country phone numbers, go to <http://www.ibm.com/planetwide>.

Before you contact IBM Rational Software Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:

What software versions were you running when the problem occurred?

Do you have logs, traces, or messages that are related to the problem?

Can you reproduce the problem? If so, what steps do you take to reproduce it?

Is there a workaround for the problem? If so, be prepared to describe the workaround.

2 Overview

Rhapsody ATG analyses a Rhapsody UML model and generates test cases. For test case generation, ATG parses and analyses the Rhapsody generated C++ code. This document contains a set of Frequently Asked Questions that shall help to work with RhapsodyATG. The “What If...?” questions below are ordered along the main workflow of RhapsodyATG.

3 Frequently Asked Questions...

3.1 What... “is the minimal configuration (processor, memory, ...) to use Rhapsody with ATG ?”

ATG is a processing intensive product. Recommended is to use machines with 1 GB main memory or more, and 1 Ghz processor speed or more.

3.2 What If... “RhapsodyATG Browser Section Is Empty?”

Check that you stereotyped at least one Rhapsody component as *atgComponent*. Launch ATG and press **File > Sync ATG Data with UML Model**

3.3 What If... “Stereotype atgComponent is not visible in the Feature Dialog of Components?”

Check that ATGProfile is available in the model (Package *Profiles*). If this is not the case press **Rhapsody > File > Add To Model...** and add **<Rhapsody-Installation>\Share\Profiles\ATG\ATGProfile.sbs**. The profile provides a set of stereotypes like *atgComponent*.

3.4 What If... “ATG Complains About Component Settings?”

ATG uses Animation to define test goals, and Simulated Time Model as well as Flat Statechart Implementation to generate test cases. This is only needed for test case generation. Test execution can be done with different settings.

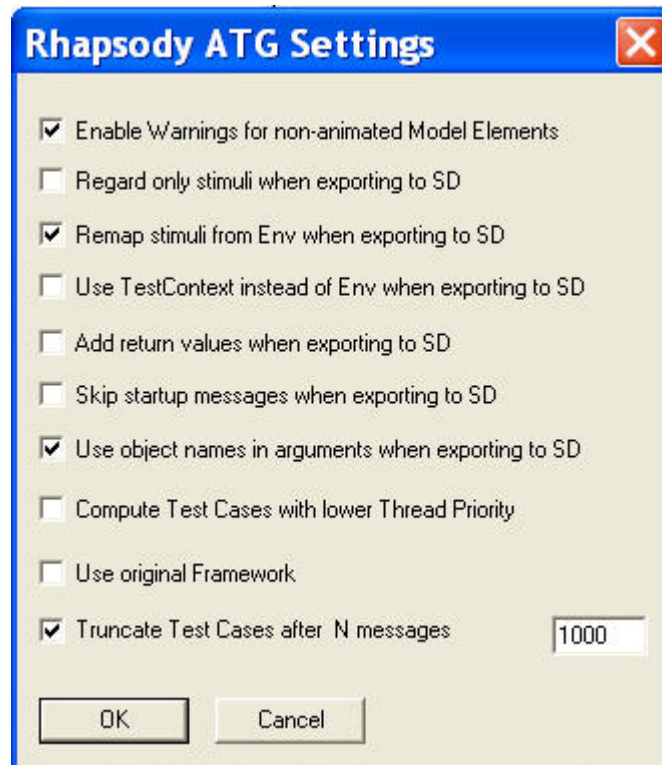
For more information see RhapsodyATG Tutorial section **Testing Components**.

3.5 What If... “ATG Test Case Generation Complains That It Can Not Compile?”

If you can not generate any test cases check that you can compile and build the Component in Rhapsody. You must be able to generate an executable component in order to apply ATG

3.6 What is... “the meaning of ATG > File > Settings?”

Users can control and influence the test case generation process and the export of the test cases with some options the settings dialog.



3.6.1 Enable Warnings for non-animated Model elements

ATG will generate a warning if test cases shall be generated for non-animated model elements. The reason is that ATG might not be able to generate test cases due to the missing animation information.

3.6.2 Regard only stimuli when exporting to SD

ATG will export only the input messages when exporting test cases to Rhapsody.

3.6.3 Remap stimuli from Env when exporting to SD

ATG will map messages that are generated as inputs by ATG to appropriate TestComponents if possible.

3.6.4 Use TestContext instead of Env when exporting to SD

When exporting test cases, inputs that are not mapped to other TestComponents are mapped to the TestContext instance line instead of the ENV instance line.

3.6.5 Add return values when exporting to SD

ATG computes expected return values for operation calls, too. The values can be exported to SDs as well. This option allows the user to control if return values are shown in SDs.

3.6.6 Skip startup messages when exporting to SD

This option allows to suppress messages which are recorded during the initialization phase of a tested system. Sometimes such messages lead to stuck test execution with TestConductor due to some inconsistencies between the Rhapsody animation layer and the ATG test case generation.

3.6.7 Use object names in arguments when exporting to SD

ATG generates test cases. Users can export test cases to SDs in Rhapsody. Messages can have arguments of different types. In the case that a message argument is a class type, the user can choose if exported SDs use animation names in message arguments (e.g. 'Telephone[0]'). The benefit is that TestConductor can use this animation information for test execution. If the export option is switched off, then ATG will replace the animation name with an asterisk symbol, which is interpreted by TestConductor as 'don't care' during test execution later on.

3.6.8 Compute test cases with lower thread priority

This option allows users to control the task priority assigned to the ATG test case generation process during the phase of test case generation.

3.6.9 Use original framework

If enabled, ATG uses the original Rhapsody oxf framework instead of its own oxf framework implementation in order to generate test cases. If this option is enabled, ATG might be able to generate test cases for models that use many oxf framework functions which might not be possible with the ATG oxf framework implementation.

3.6.10 Discard test cases with more than N messages

Sometimes the generated test cases are very long and the writing of the test cases takes long time. With this option users can control that test cases are not generated if they have a certain length N.

3.7 What If... "ATG Test Case Generation Complains About Missing Files"

ATG analyzes the settings of the used Rhapsody component/configuration in order to retrieve relevant information about additional sources and about additional include paths as added by the user to the model. This information is retrieved by ATG, but not visualized in the ATG UI. If you still cannot generate any test cases then check if you must add additional Include Paths. Check also if you must add additional source files. Both settings could be made in the "Test Definition Options" dialog for a Testing Component. This dialog is opened by highlighting a Testing Component in the ATG browser and selecting "**File->Test Definition Options**" from the ATG menu or "**Test Definition Options**" from the context-menu.

3.8 What If... "ATG Does Not Generate Any Test Cases?"

- Check that your application contains Initial Instances.
- Check that you specified a *Provided Interface* in ATG such that ATG can generate events and call operations on the system under test.

For more information see RhapsodyATG Tutorial section **Test Generation Configurations**.

3.9 What If... "ATG Does Not Cover All Test Goals?"

- try more run time...
- check that the Provided Interface has been defined appropriately
- check that the argument ranges of selected messages in the Provided Interface are set correctly
- check if unreachable test goals might be dead code...

For more information see RhapsodyATG Tutorial section **Test Generation Configurations**.

3.10 What If... "Not All Relevant Messages Are Shown In the Test Cases?"

Check that you selected all relevant messages in the *Required Interface* section of ATG.

For more information see RhapsodyATG Tutorial section **Test Generation Configurations**.

3.11 What If... "I Want to Have Not Only the First Instance as Interface Object?"

In the *Provided Interface* section of ATG you can specify which of the instances are input objects. Default is always instance 0, e.g. *class_A[0]:class_A*. You specify either

- 0 // instance class_a[0] is input object
- 0-3 // instance class_a[0], class_a[3] are input objects
- 0,2,4 // 0 // instance class_a[0], class_a[2], class_a[4] are input objects

For more information see RhapsodyATG Tutorial section **Test Generation Configurations**.

3.12 What... "Is the Meaning of *ATG > General > Use ATG Compilation Environment?*"

If the check box is NOT set, then ATG uses the Microsoft compilation environment, either Visual Studio or .NET 2002.

If the check box is et, then ATG uses its own compilation environment. Just in case that test case generation with either Visual Studio or .NET 2002 does not work enable the check box and try again.

3.13 What... "If I added an Element to the Model, but It Is Not Visible in ATG?"

Press **ATG > File > Sync ATG Data with UML Model** in order to inform ATG about the model changes.

3.14 What... "Is The Relation Between ATG Testing Component and ATG Test Generation Configuration?"

To generate test vectors with ATG, you must create a Test Generation Configuration based on a selected Testing Component. While a Testing Component describes the scope, Test Generation Configurations specify some necessary details for actual test generation.

You can specify Provided Interface and Required Interface for Testing Components. When you create a new Test Generation Configuration it inherits the interface settings of the Testing Component. You can change and override the settings in the Interface Tab of the Test Generation Configuration. The actual settings of the Test Generation Configuration are used for test case generation.

For more information see RhapsodyATG Tutorial sections **Testing Components** and **Test Generation Configurations**.

3.15 What... "Is the Meaning of the ATG Check-Box *Use Model Interface Specification*?"

If you do NOT set **Use Model Interface Specification** all classes contained in the Testing Component will be listed in the Interface Tab of ATG.

If you do set **Use Model Interface Specification** only those classes contained in the Testing Component will be listed in the Interface Tab of ATG which are stereotyped in the model as <<providedInterface>> or <<requiredInterface>.

For more information see RhapsodyATG Tutorial section **Testing Components**.

3.16 "Does ATG exploit both Statechart diagrams (attached to classes) and activity diagrams (attached to operations, for describing methods)?"

Yes, ATG exploits both.

3.17 "Does ATG require an event-driven, reactive style of architecture? Or can it be used on e.g. systems without statecharts?"

It can also be used with non event-driven systems without statecharts.

3.18 What... "If a method is not modeled, but its C++ code is provided?"

If the method is part of the C++ code to be analyzed ATG will use such a method as if it was modeled.

3.19 What... "kind of coverage can be achieved on non modeled methods?"

All decisions and conditions in such a method will be MCDC covered.

3.20 What... "needs to be modeled in UML, and what can be provided directly as C++ code?"

User decides what to model in UML and what to provide directly as C++ code. ATG can cope with both situations.

3.21 "Are the activity states of the Activity Diagrams attached to operations also included in the class instance test goals?"

No. ATG does not use states of Activity Diagrams as test goals. Nevertheless, ATG also performs code coverage based test case generation. In this context ATG also covers all decisions and conditions of Activity Diagrams.

3.22 When... "specifying events as test goals: is the goal the generation or the consumption (reception) of events?"

Goal is the generation of events.

3.23 What... "is the value of executing test cases with TestConductor?"

There are several use cases:

- to check for side effects of "real conditions", such as memory overflows and threading implications
- to perform regression testing after model changes
- to find exceptions and other problems during application execution. ATG uses a virtual memory model and behaves much more robust than a physical processor in order to generate test vectors. Hence, it is important to run test cases on the linked application.

- To analyze the behavior when the application is linked with real user libraries.

3.24 "Is it possible to use ATG on a sub-part of a model that has dependencies on for instance operations that have not been modeled yet, and need to be stubbed?"

Yes. ATG performs auto stubbing. User can modify or extend stubbed functions.

3.25 "What is the meaning of Test *Generation Strategy* Breadth-First and Beam...?"

Breadth-first search and beam-search, respectively, are two different test case generation strategies of ATG. While breadth-first search always computes the shortest test case that reaches a certain test goal, beam-search computes in general longer traces. However, by restricting the number of possibilities that ATG considers when generating input values to at most "beam-width " possibilities, beam-search can find longer test cases that cannot be found with breadth-first search. Reason is that breadth-first search always considers all possibilities for generating input values, which might result in missing test cases due to run-time complexity problems.

3.26 "What is the meaning of *Handling of Timeouts* Standard and Timestep...?"

If you have a model with some transitions with small timeouts (e.g 10ms), and one transition with a large timeout (e.g. 10000ms), then ATG most likely won't find a test case covering the transition with the large timeout. The reason in this case is that whenever a small timeout elapses, ATG checks if it can reach any new coverage goal by injecting a message of the selected input interface. If you use the option "Timesteps" and enter a large number (e.g. 1000), then ATG will most likely create a test case covering the transition with the larger timeout (and also the transitions with the small timeouts). The drawback of a large Timestep is that there are fewer chances to cover transitions without a timeout.

3.27 "Why shows the ATG browser covered operations of a class although I have selected specific instances to be covered...?"

It might happen that a specific instance (e.g. instance '0') cannot cover a particular test goal, e.g. a primitive operation of a class. However, after test case generation the ATG browser might show that this primitive operation has been covered regardless of the fact that the selected instance '0' could not cover the operation. This can be the case due to other existing instances in the tested system which covers the operation. In this case the ATG browser shows this covered primitive operation directly under the class and not under a specific instance of this class. The other primitive operations, which are covered by the selected instance, are shown in the specific browser part of the instance 0.

3.28 "Which syntax is supported to specify values of arguments in the ATG interface tab...?"

ATG supports that users can specify constraints for input arguments. For instance, for a primitive operation *foo(double p)* users can specify that ATG shall call this operation with values out of the range of -100.0 to 100.0. The supported syntax is

- comma separated list: e.g. "-50.5, 0, 10.0, 100.0"
- range specification: e.g. "-50.5 – 100.0" or alternatively "-50.5 .. 100.0"
- values can be specified as integer values as real values with dot notation, or as identifiers for e.g. string values.
- Other syntactical notations are not supported
- This holds for all other types, e.g. float, real, etc.

3.29 "Why using OM_RETURN() and OM_RETURN_VOID()...?"

ATG can generate test cases which also show the return values of operations. In order to properly execute these test cases with TestConductor it is mandatory to replace the return-statements in the operation body with either OM_RETURN(<value>) or OM_RETURN_VOID(). Otherwise TestConductor cannot check the return values, because they are not animated when using standard return-statements. The effect is that test execution seems to get stuck. Please refer to the Rhapsody documentation when using OM_RETURN(<value>) or OM_RETURN_VOID().

3.30 "What about differences in the Rhapsody SD animation and test generation with ATG...?"

ATG generates test cases by the means of virtual execution of C++ programs and operations. ATG applies the standard C++ semantics. The Rhapsody SD animation uses an animation layer as provided by the Rhapsody framework. In some situations there are slight differences in the behavior of the animation layer compared with the behavior of ATG, which might be a problem when ATG generated test cases are executed with TestConductor. Since TestConductor uses the Rhapsody animation layer it might happen that test execution gets stuck. This in particular happens sometimes during the phases of instance creation when the constructors already call operations or send events. Another reason might be that the Rhapsody animation layer applies a slightly different naming scheme for instances compared with ATG and TestConductor. This might also lead to stuck test case execution.