



Explorer Tutorial

IBM Rational Tau[®]

Explorer Tutorial



This edition applies to IBM® Rational® Tau® version 4.3 and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2000, 2009.

Notices

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

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

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

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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

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

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

Intellectual Property Dept. for Rational Software|
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142
U.S.A.

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

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

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

Copyright license

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

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

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.html.

Third-party Trademarks

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Introduction

The purpose of this tutorial is to make you familiar with the essential validation functionality in Rational Tau. With validation we mean exploring the state space of an UML system with powerful methods and tools that will find virtually any kind of possible run-time errors that may be difficult to find with regular simulation and debugging techniques.

We have on purpose selected a simple example that should be easy to understand. It is assumed that you have a basic knowledge about UML.

As you read, you should perform the exercises on your computer system as they are described.

In this tutorial we will use a model already created for you, `umlVerificationCoffeeMachine`. This example provides the user with a cup of coffee or a cup of hot water (for tea), given that the customer has inserted the required amount of money. The coffee machine can handle coins of the values 5 and 10, where 5 is the price for a cup of tea while a cup of coffee costs 10. There is another tutorial on how to create this model and verify it using Rational Tau Verifier. The tutorial can be found at `<installation directory>\Locale\en\coffmach.pdf` (you do not need to read `coffmach.pdf` to be able to go through this tutorial).

Note: Note: A detailed description on all the commands used in this tutorial can be found in Rational Tau help under [Validating UML Models : The Tau Explorer : Model Explorer Reference](#)

Generating and starting the Explorer

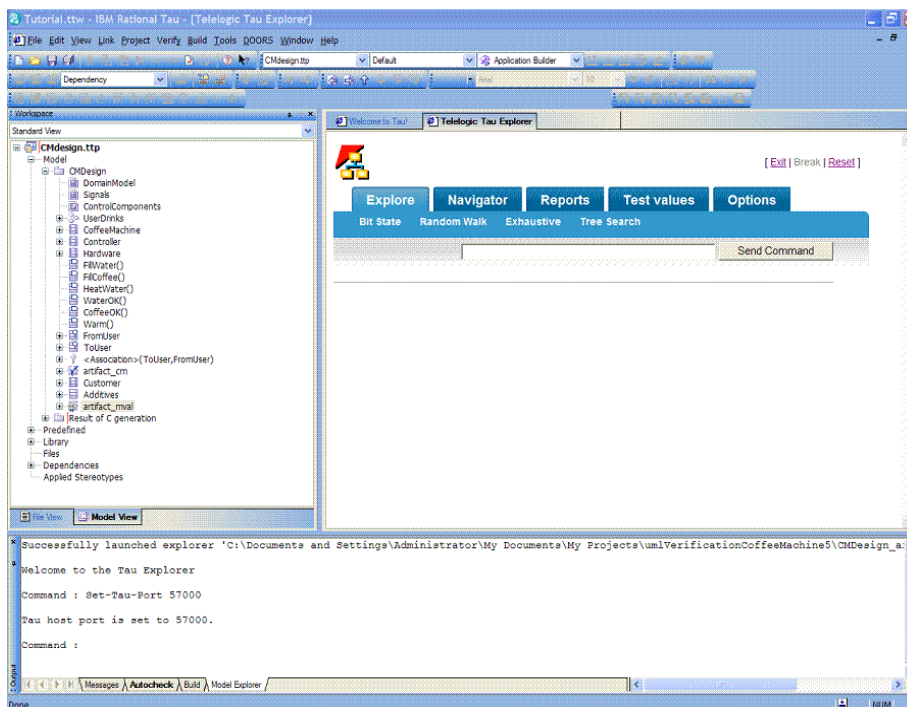
The following section will explain how to generate the Explorer artifact and how to start it.

1. Launch Rational Tau
2. Open CaffeMachine model sample by choosing *File->New-> Samples->umlVerificationCoffeeMachine->OK*
3. Enable the ExplorerExplorer addin by choosing **Tools->Customize->Addins->ModelExplorer->Close** (if it is already selected select cancel).
4. In the Model View under the CMDesign folder you will find the active class CoffeeMachine, select it by clicking on it. This is the model element that is the build root in the model.
5. Right click on “CoffeeMachine” and select **Model Explorer->New Artifact**.

In the model view you can find “Artifact0000”, this is the Explorer artifact you just created. You should also be able to see “artifact_mval” there, which is a Explorer build artifact already included in the model”.

6. Select “artifact_mval” and press Delete (this artifact is not needed now that you created your own).
7. Save the project.
8. Right-click on the “Artifact0000” and select launch. (If you get a message saying “the model is up to date do you still want to build it”, press No).

It will take a couple of seconds for the Explorer to launch. A new Explorer user interface window will be created, “Rational Tau Explorer”. See the Figure 1 below.



Note: More information about the Explorer User Interface can be found in Rational Tau help under: [Validating UML Models : The Tau Explorer : Validating an Application : The Explorer User Interface](#)

Initial tests

There are four different algorithms that can be used to go through and validate a system. Bit State, Random Walk, Exhaustive and Tree Search. The main differences between them are the strategies used to explore the state space and their memory requirements. Bit State and Random Walk are well suited for large systems, while Exhaustive and Tree Search are better suited for smaller systems. For example in the Exhaustive-Exploration method entire generated state space is stored in primary memory. This is only recommended for UML systems with small state spaces. Random walk is the fastest simple algorithm that can be used for very large UML systems.

Note: Rules used to validate a system are same in all four algorithms.

We will start by using Random walk.

1. Press Random Walk button. Because the umlCoffeeMachine is a relatively small model the validation will only take a fraction of a second.
2. In the Model Explorer output window you can find:

No of reports: 0

This means that there are no errors found in the part of the model that was tested.

Symbol coverage : 21.05

This means that only ~21% of the model has been covered; so most parts of the system have never been reached during the exploration. The reason for this is that the test values are inappropriate.

So now we will try to fix this.

1. In the Explorer user interface window select “Test values” view.

On the bottom of the window you will find default values used for the validation. Those are Integers 0, 55 and -55; and Booleans true and false.

1. Remove test values 55 and -55 (do this by pressing “Delete”).
2. In “Add Value” fields under type insert “Integer” and under “Value” insert 5, press “Add”.
3. Do the same for value 10.

Note: The coffee machine model can only handle coins of the values 5 and 10, where 5 is the price for a cup of tea while a cup of coffee costs 10.

Note: All of the things that can be done using UI window can also be done by using the command line. (More about this later)

4. Execute Random-Walk again

In the output window we can see that the coverage now has changed to 98.25%.

Note: More information about defining signals from the environment can be found in Rational Tau documentation at: [Validating UML Models : The Tau Explorer : Guidelines for Model Validation : Defining Signals from the Environment](#)

It is also possible to see which parts of the system are never reached.

5. In the command line type in:

```
Save-Coverage-Table coverage
```

This will save the coverage table in a file named “coverage”; the file is placed in the local directory.

6. To print the file run following in the command line:

```
Print-File coverage
```

7. This will print the file in the Rational Tau output window.

The numbers will show us how many times any parts of the system have been visited. If we scroll up among other things we will find:

```
Operator @NewAdditives0
0 : Start #SDTREF(U2,"u2:C:\...
-----
0 START : #SDTREF(U2,"u2:C:\...
0 TASK : #SDTREF(U2,"u2:C:\...
0 RETURN : #SDTREF(U2,"u2:C:\...
```

NewAdditives is an automatically generated constructor for the Additives type. Since the model never creates a new instance of Additives, it only references a statically generated instance, the constructor is never executed and we can't reach 100% coverage.

Analyzing Reports

To see how a bug can be analyzed we need to change the coffee machine model.

-
1. Go to the “Making Beverage” state-machine diagram in Controller:: initialize and delete the signal sending symbol “HeatWater()” (the one followed by the “CoffeeOk()”); do this by clicking on it and pressing Delete.
 2. In the upper right corner in the Explorer window there is a button “Exit”. Press “Exit”.
 3. Right-click on the Explorer artifact and launch it again.
 4. Add the test values 5 and 10.
 5. Select “Explore” and then “Random walk”

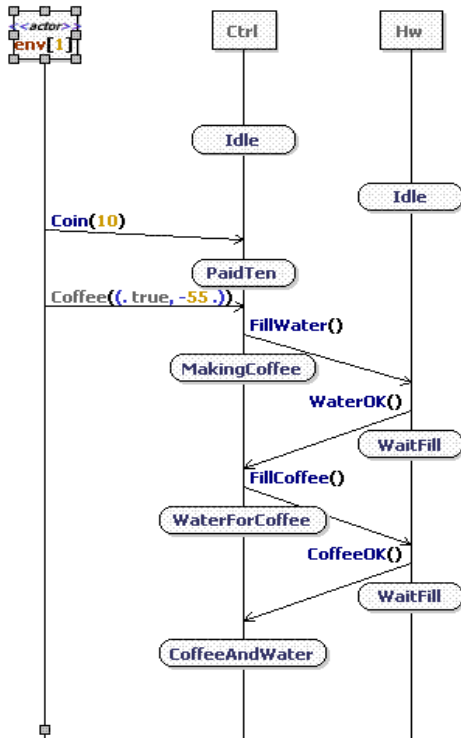
If we take a look at output window we can see one of the lines showing:

```
No of reports: 1
```

This means that the Explorer found something wrong in the model and created a report. In the Explorer select “Reports” view. There we find two links: “Deadlock Depth: 10” and “Show Trace”

Note: “Show Trace” is a new feature in Rational Tau 4.1”

Both links will show the same thing, but the first one will show it in text format and the other one in a trace diagram. Many find trace diagram easier to read, see figure 2 below.



We can see that both “Ctrl” and “Hw” are waiting for signals. If we take a look at the model we will see that “Hw” is waiting for “HeatWater()” and “Ctrl” is waiting for “Warm()”.

Depth

To explore the maximum depth concept we need to change the model again.

The coffee machine is modeled in a way that it will continue to run forever. At the end of every sequence we return to the idle state and wait for the user to insert a coin again. For each iteration we increment the Integer variables NbrOfCoffe and NbrOfTea by one. This is why any specified depth will be reached by the Explorer. So if there was no limit for the maximum depth the Explorer would also be running forever or until we stopped it or an error in the design is found.

To see how maximum depth can be reached we have to terminate the sequence in some part of the model.

Let us assume that we have designed the model so that the number of coffee orders can never exceed 20 due to some HW limitations. Now we want to test this in the Explorer also. We can do this in two ways, by defining a new rule or adding an assert function. We will start by adding an assert function.

1. In statemachine diagram “MakingBeverage” in Controller::initialize add an action symbol and insert the following:

```
[[if (#(NbrOfCoffee) > 20)
    xAssertError("nbr of Coffee larger than 20");
]]
```

2. The sending signal symbol deleted in step one needs to be drawn again so that the model looks like it did before the change. See figure 3 below.
3. Now change the diagram to fit the diagram in the Figure 3 below (delete the two Idle symbols and have one Idle symbol after the newly created action symbol):

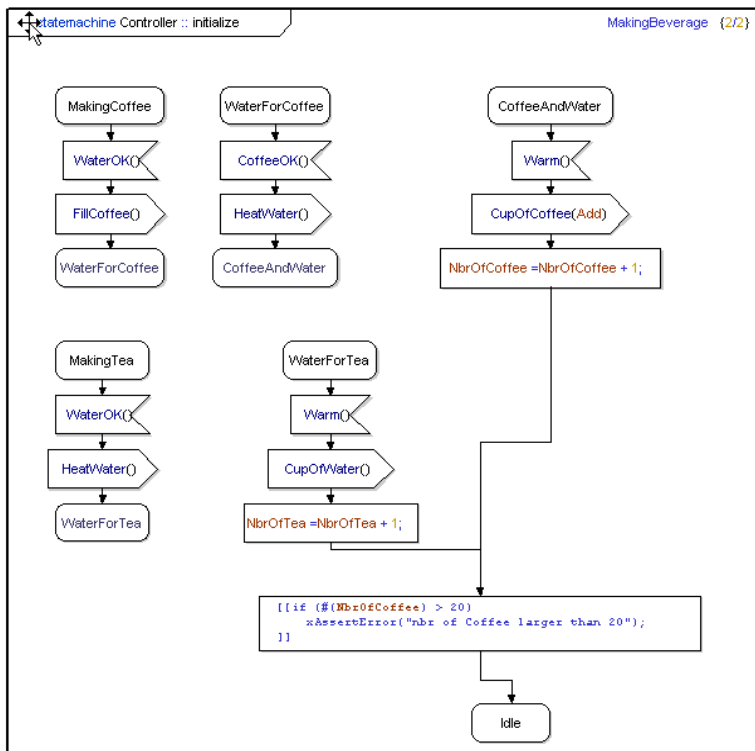


Figure 3 Assert Function

4. Re-launch the Explorer again and **add the test values five and ten.**
5. Run Random-Walk.

We find that the test report number is zero. In our case we know that we have a faulty design so the Explorer should have produced a report for this. The reason why we did not receive a report is that the default depth defined is 100 and this is not enough to reach the case when number of coffees exceeds 20.

1. Run this command in command line:

```
Define-Random-Walk-Depth 900
```

2. Run Random Walk again.

Now Tau generates one report “**Assertion is false: nbr of Coffee larger than 20 Depth: 306¹**”

So as we see depending on our model complexity and parts we need to test we will need to adjust and fine-tune many of the default settings and depth is one of them. It is out of scope for this tutorial to explain all of the settings that can be adjusted. All the information can be found in Rational Tau documentation under *Validating UML Models : The Tau Explorer : Model Explorer Reference*

Note: More information on how to use assertions can be found in Rational Tau Help under: *Validating UML Models: The Tau Explorer : Guidelines for Model Validation : Using Assertions*

We will do the same thing by defining a new rule instead of using assert function.

1. First delete the action symbol containing the assert function and re-launch the Explorer. In command line add:

```
define-rule (Ctrl:1->NbrOfCoffee>20);
```

“Ctrl:1” is an instance of Controller created by the Explorer. We can find the names for the instances by going to the Navigator view and looking at the transition names, or issue the command “List-Active-Class -“ (the dash, ‘-’, is a wildcard).

1.If you have deleted test values 0 and 55 you will get depth 388; also if you are not getting this error please check that you have added test values 5 and 10.

-
2. Add the test values 5 and 10, and change the depth to 900 again. Run Random-Walk.

A report will be created, just as predicted.

Note: Note: More information on how to use user defined rules can be found in Rational Tau Help under: [Validating UML Models : The Tau Explorer : Guidelines for Model Validation : Using User-Defined Rules](#)

Changing the starting position

In all our tests so far the Explorer would start from the initial start state in the model and go through the whole model. But if we only want to test some parts of the model, than this is not the optimal solution. If we have a large model that is taking a lot of time to go through.

What we need to do is to limit the testing to only some parts of the model and change the starting point.

1. Select the “Navigator” view and start navigating through the states by selecting up or down transitions.
2. Enter this command:

```
Define-Root Current
```
3. Run random walk again.
4. Select the “Navigate” view again

What we see is that the fist state is the same as the one we choose to start from just a while ago. In our small model this will not have any significant impact but in large models this is a very useful function.

To set the root back to the original start we use:

```
Define-Root Original
```

Together with defining new rules and placing assert functions in the model the user can control which parts of the system will be tested.

Conclusion

We have looked at some basic functions the Explorer has to offer. As mentioned at the start of this tutorial the purpose is to make you familiar with the essential validation functionality in Rational Tau. We have looked at some basic functionalities, like how to work with test values and how to analyze reports generated by the tool. Now you should have an idea on how to use the Explorer for validating your own UML model. This tutorial is only an introduction. There is a more detailed description on how and why you can use the Explorer on your own model in Rational Tau help.

Further Reading

Validating UML Models: The Tau Explorer : Validating an Application: Explains some principals on how the Explorer works. Here you can also find more information on User Interface and how to generate and start the Explorer

Validating UML Models: The Tau Explorer : Guidelines for Model Validation:

1. *Validating a UML System:* Some basic explanations on why and how to use the Explorer for Validating a UML System
2. *Validating Large Systems:* Shows how to use some useful techniques when designing and validating large UML systems
3. *Defining Signals from the Environment:* Describes how to work with test values
4. *Validating Systems with External C Code:* Description on how to directly use the Explorer on a system that uses external C code; and explains the restrictions.
5. *Using User-Defined Rules:* How to define your own rules
6. *Using Assertions:* How to use assertions

Validating UML Models: The Tau Explorer : Model Explorer Reference: This section provides an alphabetical listing of all available commands in the Explorer. You should use this section continually as you are working with the Explorer.

Technical Support and Documentation

Contacting IBM Rational Software Support

If the self-help resources have not provided a resolution to your problem, you can contact IBM® Rational® Software Support for assistance in resolving product issues.

Prerequisites

To submit your problem to IBM Rational Software Support, you must have an active Passport Advantage® software maintenance agreement. Passport Advantage is the IBM comprehensive software licensing and software maintenance (product upgrades and technical support) offering. You can enroll online in Passport Advantage from <http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html>

- ◆ To learn more about Passport Advantage, visit the Passport Advantage FAQs at http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html.
- ◆ For further assistance, contact your IBM representative.

To submit your problem online (from the IBM Web site) to IBM Rational Software Support, you must additionally:

- ◆ Be a registered user on the IBM Rational Software Support Web site. For details about registering, go to <http://www.ibm.com/software/support/>.
- ◆ Be listed as an authorized caller in the service request tool.

Submitting problems

To submit your problem to IBM Rational Software Support:

1. Determine the business impact of your problem. When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting.

Use the following table to determine the severity level.

Severity	Description
1	The problem has a <i>critical</i> business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
2	This problem has a <i>significant</i> business impact: The program is usable, but it is severely limited.
3	The problem has <i>some</i> business impact: The program is usable, but less significant features (not critical to operations) are unavailable.
4	The problem has <i>minimal</i> business impact: The problem causes little impact on operations or a reasonable circumvention to the problem was implemented.

2. Describe your problem and gather background information. When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Rational Software Support specialists 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?

To determine the exact product name and version, use the option applicable to you:

- ◆ Start the IBM Installation Manager and select **File > View Installed Packages**. Expand a package group and select a package to see the package name and version number.
- ◆ Start your product, and click **Help > About** to see the offering name and version number.
- ◆ What is your operating system and version number (including any service packs or patches)?
- ◆ Do you have logs, traces, and messages that are related to the problem symptoms?
- ◆ Can you recreate the problem? If so, what steps do you perform to recreate the problem?

-
- ◆ Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, or other system components?
 - ◆ Are you currently using a workaround for the problem? If so, be prepared to describe the workaround when you report the problem.
3. Submit your problem to IBM Rational Software Support. You can submit your problem to IBM Rational Software Support in the following ways:
- ◆ **Online:** Go to the IBM Rational Software Support Web site at <https://www.ibm.com/software/rational/support/> and in the Rational support task navigator, click **Open Service Request**. Select the electronic problem reporting tool, and open a Problem Management Record (PMR), describing the problem accurately in your own words.
 - ◆ For more information about opening a service request, go to <http://www.ibm.com/software/support/help.html>
 - ◆ You can also open an online service request using the IBM Support Assistant. For more information, go to <http://www.ibm.com/software/support/isa/faq.html>.
 - ◆ **By phone:** For the phone number to call in your country or region, go to the IBM directory of worldwide contacts at <http://www.ibm.com/planetwide/> and click the name of your country or geographic region.
 - ◆ **Through your IBM Representative:** If you cannot access IBM Rational Software Support online or by phone, contact your IBM Representative. If necessary, your IBM Representative can open a service request for you. You can find complete contact information for each country at <http://www.ibm.com/planetwide/>.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Rational Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Rational Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Rational Software Support Web site daily, so that other users who experience the same problem can benefit from the same resolution.

