TestManager 使用手册

翻译
测试部： 姚伟

日期：2003-7-1

前言（Preface）

Rational TestManager是一个开放的可扩展的构架，他统一了所有的工具、成品（**artifacts**）和数据，而数据是由测试工作产生并与测试工作（`effort`）关联的。在这个唯一的“保护伞”（**umbrella**）下，测试工作中的所有负责人（`Stakeholder`）和参与者能够定义和提炼他们将要达到的质量目标。

本手册描述了如何使用Rational TestManager去支持在RUP中定义的五个测试活动，以及如何使用Rational TestManager进行功能和性能测试。

读者（Audience）

本手册面向项目分析人员，项目设计人员和开发人员，质量保证组成员，项目经理，以及任何包含在此次测试工作中的其他负责人。

其他资源（Other Resources）

- TestManager包含完整的在线帮助。从主工具栏中，选择**Help**菜单中的一个选项。

  注意事项：本手册包含概念性的信息。对于过程的详细说明，参阅TestManager的帮助。
- 所有手册都是可在线利用的，格式为HTML或PDF。这些手册在*Rational Solutions for Windows*的在线文档CD中。
- 更多有关培训机会的信息，参阅Rational University站点：http://www.rational.com/university.

与 Rational 技术发布的联系（Contacting Rational Technical Publications）

要发送有关Rational产品文档的反馈信息，请发送e-mail到我们的技术发布部门，地址为techpubs@rational.com.

与 Rational 技术支持的联系（Contacting Rational Technical Support）

如果在你安装，使用和维护本产品时，出现了问题，请你与下面的Rational技术支持联系：

Your Location	Telephone	Facsimile	E-mail
North America	(800) 433-5444 (toll free) (408) 863-4000 Cupertino, CA	(781) 676-2460 Lexington, MA	support@rational.com
Europe, Middle East, Africa	+31 (0) 20-4546-200 Netherlands	+31 (0) 20-4545-201 Netherlands	support@europe.rational.com
Asia Pacific	+61-2-9419-0111 Australia	+61-2-9419-0123 Australia	support@apac.rational.com

注意事项：在你与Rational技术支持联系时，请准备提供下面的信息：

●你的姓名，电话号码和公司名称。

●你的电脑种类和型号。

●你所用的操作系统和版本号。

●产品发布号码和序列号。

●你的用例ID号码（如果你是在跟踪一个先前已记录过的问题）。

Part 1:

Using TestManager to Manage Testing Projects

简介（Introducing Rational TestManager）1

本章节将向你介绍Rational TestManager。它包括下面的主题：

☐什么是Rational TestManager

☐TestManager的工作流程

☐TestManager和其他Rational产品

☐TestManager与其扩展性

☐虚拟的测试人员

☐功能和性能测试

☐本地和代理的测试用机

☐Suites

☐开始运行Rational TestManager

☐TestManager的主要窗口

什么是 Rational TestManager（What Is Rational TestManager）

测试是软件开发中的反馈过程。它告诉你在一个开发过程中任意给定迭代的规定过程，哪里需要做修改。同时，也告诉你关于你将开发的系统的现存质量。

包含在项目中的每个人都是定义系统质量如何被评估并对问题做出修改的一个“stakeholder”。例如：

- 项目分析员需要了解系统的可用性，完整性，和系统use cases的质量，系统特征，以及系统的需求支持。
- 项目设计人员和开发人员需要理解他们设计或开发的组件说明和子系统。
- 质量保证组成员需要编制计划以测试系统。此外，他们需要理解和定义测试计划的要素与开发过程的其他要素之间的关系。这里的跟踪能力关系允许QA组去理解如何在他们工作的项目中做全局修改，并去定义他们将要测试的系统要素。
- 项目经理需要这些测试结果提供的信息来做出决定，系统对于发布的可接受性和准备性。他们的决定将以其他组成员的输入为基础，比如分析人员和开发人员，他们由那些相同的衡量标准来勾勒出系统的说明蓝图。

测试工作经常表现为整个项目过程的25–50%。收集需求数据，跟踪测试资产之间的关系，并提供一个普通的测试过程输出的说明描述，当然测试过程往往包含了多种工具的使用。这可以使它几乎不可能去有效地跟踪依赖的结果，和得到一个简要，以及系统说明的连续视图。

Rational TestManager是一个开放的可扩展的构架，他统一了所有的工具、制造（**artifacts**）和数据，而数据是由测试工作产生并与测试工作（effort）关联的。在这个唯一的保护伞（**umbrella**）下，测试工作中的所有负责人（Stakeholder）和参与者能够定义和提炼他们将要达到的质量目标。项目组定义计划用来实施以符合那些质量目标。而且，最重要的是，它提供给了整个项目组一个及时地在任何过程点上去判断系统状态的地方。

质量保证专家可以使用TestManager去协调和跟踪他们的测试活动。测试人员使用TestManager去了解需要的工作是什么，以及这些工作需要的人和数据。测试人员也可以了解到，他们工作的范围是要受到开发过程中全局变化的影响的。TestManager是这样一个地方，它会提供与系统质量相关联的所有问题的答案。

工作流程（TestManager Workflow）

TestManager工作流程支持RUP定义的5个主要的测试活动，他们是一个软件工程过程：

●测试的计划

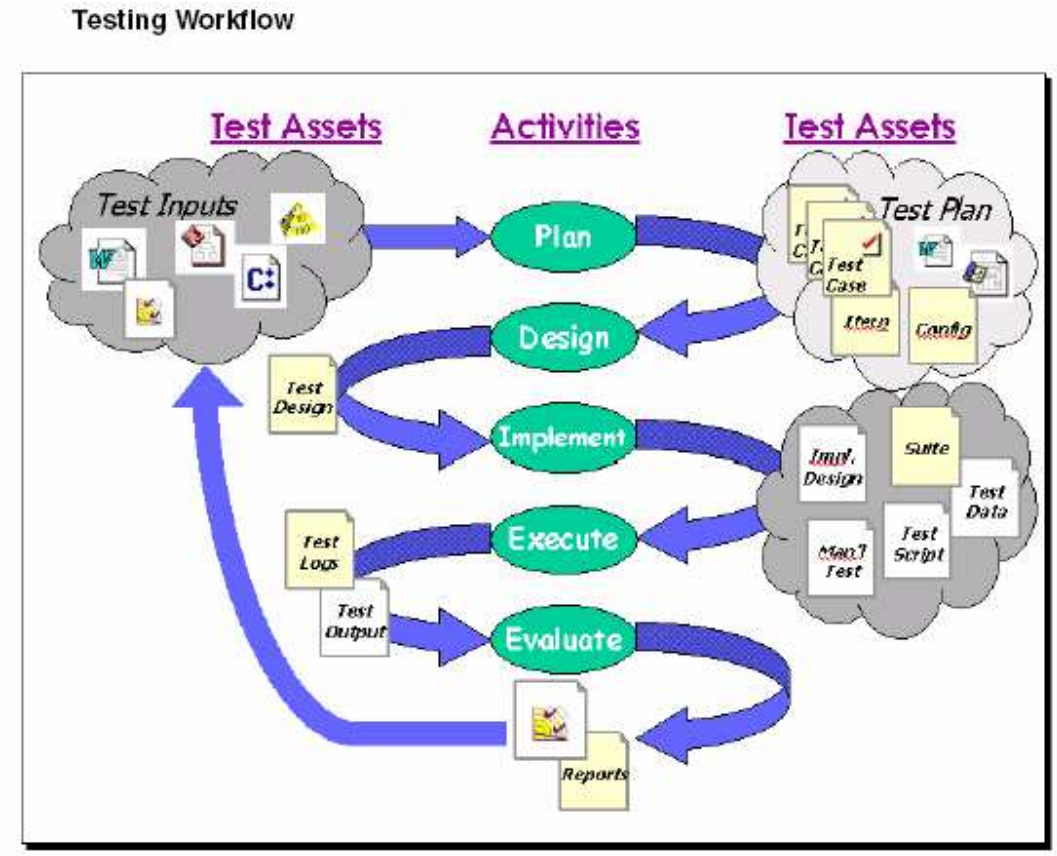
●测试的设计

●测试的实施

●测试的执行

●测试的评估

这些活动的每一个都与测试资产有输入和输出的交互，如下图所示：



编制测试计划（Planning Tests）

测试计划活动包含对下面问题的回答：

●什么和哪里？——需求，虚拟模型，和其他的测试输入告诉你要测试什么和在哪里执行这些

测试。

●为什么？——测试输入告诉你为什么要做某种测试。例如，测试可以使系统的需求被有效地执行。

●什么时候？——迭代的计划告诉你什么时候测试必须执行和必须通过。

●谁？——测试计划，迭代计划，或项目计划告诉你由谁来执行这些测试活动。

更多关于测试计划的信息，参阅23页*Planning Tests*的内容。

测试的输入（Test Inputs）

你的测试计划编制的第一步是去确定测试输入。一个“测试输入”（*test input*）是测试的依靠或是测试需要的验证。测试输入帮助你决定你需要测试的内容。他们也帮助你确定基于开发过程中可能需要的变化。迭代的开发过程是重要的，而在开发过程中的变化是一个频繁的必要的过程部分。

TestManager有两个内置的测试输入类型：

●在一个Rational RequisitePro项目中的需求

●在一个Rational Rose虚拟模型中的元素

这些内置测试输入类型使你容易地获得需求和模型元素，并让你将这些输入与其他用以跟踪目的的测试资产相关联。

TestManager也支持常规的测试输入类型。使用一个在TestManager内部的常规测试输入，你需要写一个测试输入适配器（Test Input Adapter）或使用一个由Rational Software或Rational’s合作者提供的适配器。例如，使用Microsoft Excel表格中的数值作为一个测试输入，你将需要为Excel写一个测试输入的适配器。更多信息，参阅13页*Defining Custom Test Input Types*的内容。

测试计划（Test Plans）

在你确定了你的测试输入，你就可以使用TestManager来创建一个测试计划。测试计划为项目中其他的测试资产提供一个组织结构。

测试计划可以包含一个信息的多样采集和提出的很多问题，包括：

●什么测试是必须要被执行的？

●什么时候测试必须被执行并被期望通过？

●对于每个测试由谁负责？

●哪里必须是测试被执行的？换句话说，在什么样的硬件和软件配置下，测试必须被执行？

项目可包含多样的测试计划。你可以为测试的每一个阶段编制一个计划。不同的工作组可以有他们自己的测试计划。一般地，每一个计划应当有一个唯一的高水平的测试目标（例如，测试文件的维持效用）。

每一个测试计划都包含了测试用例文件夹和测试用例。

测试用例文件夹（Test Case Folders）

在一个测试计划中，你可以创建测试用例文件夹（*test case folders*）来分层次地组织你的测试用例。普通的组织可以反映系统构架，主要的用例（use cases），需求，或是他们的混合体。

测试用例（Test Cases）

测试用例是TestManager中的测试资产，它可以回答这个问题，“我将要测试什么？” 你开发测试用例去定义单个的事件，有效地确保系统工作在假定的方式下，以及在你成品之前对于质量的必然的建立。

每一个测试用例都被一个组内成员所拥有。这也是问题“谁将进行测试？”的答案。

迭代（Iterations）

使用迭代去确定一个测试用例在什么时候必须通过。一个迭代是在一个项目期间被定义的时间跨度。一个迭代的结束是一个里程碑（milestone）。在一个迭代期间，处于一些及时点上，产品需要符合一个确定的质量标准来达到一个里程碑。质量标准由那些必须通过的测试用例定义。迭代可能由几组成员来定义，比如项目经理，产品经理，和分析人员－－反复地遍及整个测试过程。

配置（Configurations）

使用配置确定测试用例在哪里执行－－在什么硬件和软件配置下。例如，要确保你的测试用例

可以在4种不同的操作系统下执行，你就可以为每一个操作系统创建一个配置。然后将这4种配置与测试用例联系起来，以创建成配置的测试用例。为使测试用例通过，需要所有的成配置的测试用例必须通过。

配置可能由几组成员来定义，比如项目经理，产品经理，和分析人员－－反复地遍及整个测试过程。

测试的设计（Designing Tests）

设计测试的活动可以回答这个问题，“我将如何执行测试？”。一个完全的测试设计会告知读者，有关需要与系统被获得的活动和他们应该期待观察的行为和特性，当然，如果系统正在适当地运行的话。

设计测试是一个迭代的和行进中的过程。你应当能够在任何系统执行之前开始测试的设计，他们基于用例（use cases），需求，原型，和其他等等。当系统被描述的更加清晰时，测试设计应当与系统一起更加细节化。

注意事项：一个测试设计不同于设计工作，它应当被用来确定如何建立你的测试实施。

在TestManager中，你可以设计你的测试用例：

●指明基本步骤需要与应用和系统交互，以便执行测试。

●指明如何有效的使特征恰当地工作。

●说明测试的前置条件和后置条件。

●说明测试的可接受标准。

软件开发给定的迭代性质，设计比一个手工测试的执行要更抽象，但是它可以容易地发展成一个测试执行。更多有关设计测试的信息，参阅47页*Designing Tests*的内容。

测试的实施（Implementing Tests）

测试实施的活动包括了可复用测试脚本（*test scripts*）的设计和开发，测试脚本用来实施你的测试用例。在你创建了实施之后，你可以将他与测试用例联系在一起。

在每一个进行测试的项目中，实施都是不相同的。一个项目里，你可能决定既创建自动测试的脚本又有手工测试的脚本。另外一个项目中，你可能需要写一个模块使用一个工具的组合。

TestManager提供下面的内置的测试脚本类型以支持实施：

Type of Test Script	Description
GUI	A functional test script written in SQABasic, a Rational proprietary Basic-like scripting language. Created in Rational Robot. (Available only if Rational Robot is installed.)

Type of Test Script	Description
VU	A performance test script written in VU, a Rational proprietary C-like scripting language. Created in Rational Robot. (Available only if Rational Robot is installed.) **Note:** When you start to record a VU test script, you actually record a session. You can generate VU or VB (Visual Basic) test scripts from the recorded session, depending on a recording option that you select in Robot.
Manual	A set of testing instructions to be run by a human tester. Created in Rational ManualTest.

测试脚本的类型	说明
GUI	一种用SQABasic编写的功能测试脚本，一种Rational专有的类Basic脚本语言。由Rational Robot创建。（仅仅在已安装Rational Robot的情况下可用。）
VU	一种用VU编写的性能测试脚本，一种Rational专有的类C脚本语言。由Rational Robot创建。（仅仅在已安装Rational Robot的情况下可用。） **注意事项：**当你启动记录一个VU测试脚本时，你实际上记录了一个session。你可以从被记录的session中生成VU或VB脚本，这依赖于你在Robot中选择的记录选项。
Manual	一个测试指令集，以被人工执行。由Rational Manual Test创建。

TestManager也支持已注册的测试脚本其他类型的实施。更多信息，参阅13页*Defining Custom Test Script Types*的内容。

你也可以使用**suites**去实施测试。一个**suite**是一个容器，允许你设计一个更大的测试用例集合和你想要执行的实施。一个**suite**可以有参数，诸如order（顺序），dependencies（依赖），iterations（迭代），random operations（随机操作），和其他等等。更多有关实施测试用例的信息，参阅55页*Implementing Tests*的内容。

测试的执行（Executing Tests）

你的测试执行活动包含了测试实施的执行已确保系统功能的正确性。在TestManager中，你可以执行：

●一个单独的测试脚本

●一个或更多的测试用例

●一个suite，执行一些测试用例和测试脚本的混合体，通过一台或更多的测试用机和虚拟测试者。

TestManager提供下面内置的支持执行的测试脚本类型：

Type of Test Script	Description
GUI	A functional test script written in SQABasic, a Rational proprietary Basic-like scripting language.
VU	A performance test script written in VU, a Rational proprietary C-like scripting language.
Manual	A set of testing instructions to be run by a human tester.

Type of Test Script	Description
VB	A test script written in the Visual Basic language.
Java	A test script written in the Java language.
Command line	A file (for example, an .exe file, a .bat file, or a UNIX shell script) including arguments and an initial directory that can be executed from the command line.

测试脚本的类型	说明
GUI	一种用SQABasic编写的功能测试脚本，一种Rational专有的类Basic脚本语言。
VU	一种用VU编写的性能测试脚本，一种Rational专有的类C脚本语言。
Manual	一个测试指令集，以被人工执行。
VB	一种用Visual Basic语言编写的测试脚本。
Java	一种用Java语言编写的测试脚本。
Command Line	一个文件（例如，一个.exe文件，一个.bat文件，或一个UNIX shell脚本）包含了从command line中可被执行的论点和一个初始化的路径。

TestManage也支持已注册的测试脚本其他类型的执行。更多信息，参阅13页*Defining Custom Test Script Types*的内容。

更多有关测试执行的信息，参阅87页*Executing Tests*的内容。

测试的评估（Evaluating Tests）

测试的评估活动包括：

- 确定实际测试执行的有效性。执行的是否完全？执行失败是否因为不符合前置条件？
- 分析测试输出以确定结果。在执行测试过程中，你查看报告上已产生的数据来检验该执行是否是可接受的。
- 查看合计的结果以检查对测试计划，测试输入，配置等的覆盖程度。这也可以被用来衡量测试的进展和对分析的趋向。

更多有关测试评估的信息，参阅127页*Evaluating Tests*的内容。

TestManager 和其他 Rational 产品（TestManager and Other Rational Products）

TestManager可以单独购买或作为其他Rational包的一部分。

当你与其他的Rational产品一起安装时，它会与那些产品紧紧地结合在一起。

Rational 统一开发过程（The Rational Unified Process）

Rational统一开发过程是一个一个软件工程过程，以提高针对关键性活动的团队开发效率。它讲述了软件最优实践经过的指导方针，模板，和工具指导。

快速地查看直接关系到测试的RUP的范围：

- 在TestManager中，点击**Help > Extended Help**。

如果已经安装了Rational Suite，可查看完整的RUP在线版本：

- 点击**Start > Programs > Rational Suite > Rational Unified Process**。

项目和 Rational 管理员（Projects and the Rational Administrator）

与TestManager一起工作时，在Rational projects中保存那些你创建的信息。你使用Rational Administrator来创建和管理Rational项目。一个Rational项目保存软件的测试和开发信息。在你测试用机上的所有Rational组件从相同的项目中更新和回收数据。

注意事项：一个Rational项目中的数据类型依赖于你已经安装的Rational软件。

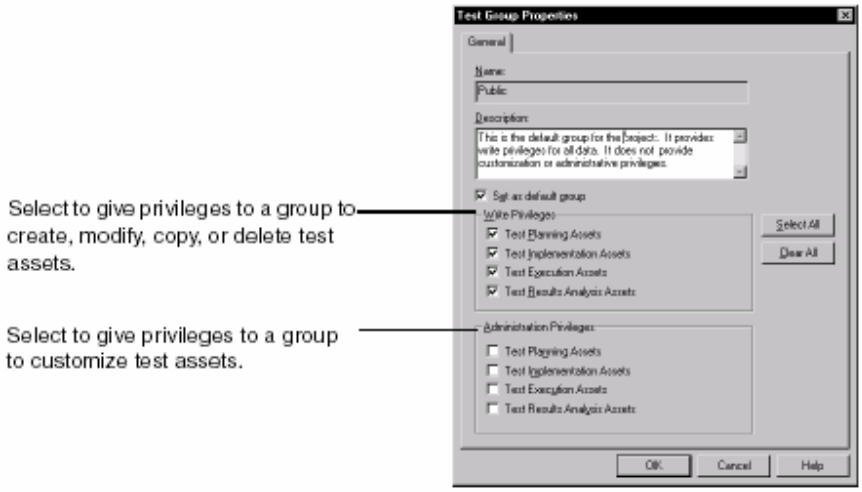
一个Rational项目可以的组成如下：

- *Rational Test datastore*——保存应用程序的测试信息，诸如测试计划，测试用例，测试日志、报告和构架。
- *Rational RequisitePro project*——保存产品或系统需求，软件和硬件需求，及用户需求。当一个RequisitePro datastore与一个在Rational Administrator中的项目建立了关联时，TestManager会自动地使用在datastore中的需求作为测试的输入。
- *Rational Rose models*——保存业务过程，软件组件，类和对象的虚拟模型，以及分布和配置部署的虚拟模型。
- *Rational ClearQuest database*——保存对于软件开发的变化需求，包括增加的需求，缺陷报告，和文档的修改。

对于 Rational Test Datastore 的安全和权限（Security and Privileges for the Rational Test Datastore）

当管理员使用Rational Administrator创建了一个Rational项目时，他们确定Rational Test datastore的安全。他们创建测试用户，这些测试用户默认为Public test group（公共测试组）的部分。测试用户获得该测试组的权限。一个管理员可以使用Rational Administrator来改变组权限和创建新组。

下面的图示展现了在Rational Administrator中的Test Group属性对话框：



更多有关权限设置的信息，参阅*Using the Rational Administrator*手册或Rational Administrator的帮助。

自动测试脚本与 Rational Robot （Automated Test Scripts and Rational Robot）

使用Rational Robot，你可以为功能测试和性能测试开发自动测试脚本。

使用Robot：

- 执行完全的功能测试。记录那些通过你的应用程序导航和检验点测试目标状态的测试脚本。
- 执行完全的性能测试。记录测试脚本，而这些脚本帮助你确定在变化的负载下，一个系统是否在用户定义的响应时间的标准范围内运行。
- 测试用IDEs（集成开发环境）开发的应用程序，诸如Java，HTML，Visual Basic，Oracle Forms，Delphi，和PowerBuilder。你可以测试那些即使是在应用程序的接口中不可见的对象。
- 在测试脚本的录制回放期间，收集关于一个应用程序的诊断信息。Robot集合了Rational Purify ，Rational Quantify，和Rational PureCoverage。你可以在一个诊断工具下录制回放测试脚本，在TestManager中的测试日志中查看结果。

组件测试与 Rational QualityArchitect （Component Testing and Rational QualityArchitect）

Rational QualityArchitect是对于测试由诸如EJB和COM技术建立的中间件组件时的一个集成工具集合。

QualityArchitect，与Rational Rose关联，为你的Rose模型中的组件和交互产生测试脚本。当脚本产生时，可以编辑并在你的部署配置环境或RationalTestManager中执行。

使用QualityArchitect，你可以：

●生成测试脚本，针对单元测试的独立方法或一个在测试下的组件中的功能。

●生成在一个集成组件集合中实现的业务逻辑的测试脚本。测试脚本可以从Rose交互图或使用Session Recorder现存组件中直接生成。

●生成你可以隔离使用测试组件的**stubs**，与其他的测试下的组件相分离。

●通过Rational PureCoverage进行跟踪代码覆盖，通过Rational TestManager进行模型水平覆盖。

需求与 Rational RequisitePro （Requirements and Rational RequisitePro）

Rational RequisitePro是帮助项目组控制开发过程的一个需求管理工具。RequisitePro由对需求的Word文档的连接和提供的跟踪能力来组织你的需求，并在整个项目生命周期中变更管理。

在你使用Rational Administrator创建一个Rational项目时，你可以将一个RequisitePro项目与这个管理项目关联起来。然后，你可以使用在RequisitePro项目中的需求作为TestManager中你的测试计划的测试输入，可以容易地建立需求与测试用例之间的关联。

你也可以使用在其他RequisitePro项目中的需求作为测试输入。

模型元素与 Rational Rose （Model Elements and Rational Rose）

Rational Rose帮助你具体化，明确化，构筑和描述你的系统架构的组织和行为。利用Rose，你可

以使用UML（统一建模语言）为系统提供一个可视化的概述，UML是可视化的和描述软件系统的工业标准语言。

你可以在TestManager中使用Rose模型元素作为测试输入，并可容易的建立模型元素与测试用例之间的关联。

缺陷与 Rational ClearQuest （Defects and Rational ClearQuest）

Rational ClearQuest是一个需求变更管理工具，以跟踪和管理整个开发过程中的缺陷，及需求的变更。利用ClearQuest，你可以管理每一种与软件开发相关联的变更活动的类型，包括扩充需求，缺陷报告，和文档修改。

利用TestManager，你可以直接从一个测试日志里提交缺陷到ClearQuest中。TestManager自动地填充缺陷到ClearQuest里的一些区域，缺陷信息来源于测试日志。

报告与 Rational SoDA （Reports and Rational SoDA）

Rational SoDA生成最新的项目数据报告，而这些数据是由Rational Suite中的一个或多个工具收取出来的。SoDA可以与一个Rational工具一起工作，诸如RequisitePro，或是从多个工具组合信息，比如RequisitePro，Rose，TestManager，和ClearQuest。这些报告提供你的团队交流的更加有效和持续的一个方法。

比如，利用SoDA你可以创建关于一个软件开发项目的报告：

- 从RequisitePro而来的需求
- 从Rose而来的软件模型
- 从TestManager而来的测试标准
- 从ClearQuest而来的缺陷跟踪信息

TestManager 与可扩充性（TestManager and Extensibility）

利用提供的多样的应用程序接口（APIs），TestManager不是有限地支持软件开发资产和由Rational工具产生的测试资产。APIs提供TestManager软件入口，能够实施suits特定测试目的

的插入功能性（The APIs provide hooks into and from the TestManager software,enabling implementers to plug in functionality that suits specific testing purposes.）。除内置的测试输入类型和测试脚本类型外，TestManager支持常规的测试输入类型和测试脚本类型。

定义常规的测试输入类型（Defining Custom Test Input Types）

对于测试的任何一种对象都可以被定义和管理为一个测试输入类型－－例如，在Microsoft Project文件或Excel表格中的对象。

作为一个例子，你可以定义C++语言项目文件以作为一个测试输入类型，如果你想知道在一个源文件改变时哪些测试需要被改变和返回。

由于TestManager支持一个常规测试输入类型，这里必须是一个用户实施动态链接库（user-implemented DLL），被称为一个测试输入适配器（TIA）。这个适配器包含了对于任务的功能，而对于的任务，诸如链接和断开测试输入源，检查一个输入是否被更改，及设置多种过滤器。更多有关常规适配器的信息，参阅*Rational TestManager Extensibility Reference*手册。

有关在TestManager中定义新的测试输入类型的信息，参阅27页*Custom Test Input Types*的内容。

定义常规的测试脚本类型（Defining Custom Test Script Types）

TestManager的可扩展的测试脚本类型的功能性可以让你使用任何与你测试环境相适应的测试工具来实施常规测试脚本。

这里有几种扩展TestManager的方法以支持一个新的测试脚本类型：

- 使用Command Line Test Script Console Adapter去创建，打开和编辑测试脚本。
- 建立一个常规的Test Script Console Adapter或使用Rational（或它的合作商）提供的一个适配器去打开（可选择性地创建和编辑）测试脚本。（更多有关常规适配器的信息，参阅*Rational TestManager Extensibility Reference*手册。）
- 使用Command Line Test Script Execution Adapter去执行测试脚本。
- 建立一个常规的Test Script Execution Adapter或使用Rational（或它的合作商）提供的一个适配器去执行测试脚本。（更多有关常规适配器的信息，参阅*Rational TestManager Extensibility Reference*手册。）

有关在TestManager中定义测试脚本类型的信息，参阅57页*Custom Test Script Types*的内容。

虚拟的测试者（Virtual Testers）

一个虚拟的测试者是在一台测试机上执行测试脚本的一个单一的实例。对于功能测试，在一台测试机上，仅仅是每次一个虚拟的测试者可执行。对于性能测试，一台测试机上可以同时执行多个虚拟测试者。

注意事项：虚拟的测试者执行在已安装了代理（Agent）软件的测试机上。有关信息，参阅15页*Local and Agent Computers*的内容。

一个虚拟的测试者执行一个客户端与其服务器之间的仿真通信。例如，当你在Robot中记录一个**session**时，Robot记录一个客户端到服务器的请求－－诸如Oracle，Microsoft SQL Server，和HTTP请求。

性能测试允许你添加一个工作量到一个C/S系统在一台服务器上执行多个虚拟的测试者。虚拟的测试者也能让你确定可测量性和衡量服务器响应次数。

功能和性能测试（Functional and Performance Testing）

当你计划测试时，你可能需要考虑是否需要功能测试，性能测试或两者皆有。

功能测试（Functional Testing）

在功能测试中，代表性地测试应用程序的准确性和如何在不同的测试机上运行。

功能测试趋向于已明确的对象和结果。例如，如果有一个特征是向disk存储文件，那么测试这个特征是相对简单的。如果一个文件被正确地存储，那它就通过了该测试。反之，如果没有被正确地存储，那它就失败了。

有关功能测试的信息，参阅第二部分，*Functional Testing with Rational TestManager*.

性能测试（Performance Testing）

在性能测试中，你可以衡量下面的内容：

●客户端响应时间。这是整个结束到结束的作为一个用户看到的响应时间。这个时间被认为是一个用户输入一个请求，服务器端响应该请求，和用户最终看到响应结果的时间。

●服务器响应时间。这个时间被认为是服务器处理一个请求的时间。

性能测试比功能测试复杂的多，因为性能测试本身是主观的。像这样，一个用户可能觉得太慢，而另一个用户可能认为这是可以接受的。因此，在编制性能测试的测试计划时，你需要加入一些看法到构成的可接受性能中去。

另一个性能测试的复杂性是性能依赖工作量条件下变化的巨大。

查询一个系统上的数据库，主要使用是针对CPU-intensive活动产生的一个响应时间，而在针对于产生I/O-intensive数据库报告的情况下，执行相同的查询，得到的响应时间是与前者不相同的。（Querying a database on a system that is primarily used for CPU-intensive activities yields a different response time than performing the same query on a system used primarily for generating I/O-intensive database reports.）

相关性能测试的信息，参阅第三部分，*Performance Testing with Rational TestManager*。

本地与代理测试机（Local and Agent Computers）

你协调从一台运行TestManager的单一窗口的测试机上而来的你所有的测试用例、测试脚本和suites的活动。这就是所说的本地测试机（*Local computer*）

在一个测试执行期间，你在一台本地机或一台被设计为代理机（*Agent computers*）上录制回放测试脚本。你使用一台代理机如下：

●添加工作量到服务器。如果你执行一个有大量虚拟测试者的测试，那么你可以使用代理机来添加工作量到服务器。

●在多于一台测试机上执行测试脚本。如果你正在执行一个功能测试，那么你可以在一台代理机上执行测试脚本来节约时间，而这台可利用的代理机代替了执行所有测试脚本的本地机。

●执行有多个虚拟测试者的功能测试。如果你正在执行一个有多个虚拟测试者的功能suite，那么你就需要一个代理机，因为一个虚拟测试者仅仅执行在一台测试机上。

●测试配置。如果你正在测试不同的硬件和软件配置，那么你可以在不同的代理机上执行测试脚本，而这些代理机设置了你需要的配置。

Suites

多重测试脚本和并联的测试机可以包含在一个测试中去。*Suites*能让你去协调测试脚本执行的方法。在功能测试中，suites让你在一台可用的测试机上并行地执行测试脚本，以便你的测试可以执行地更快点。在性能测试中，suites添加工作量到服务器中。

在你已经使用TestManager创建了一个suites，你可以执行这些suites反复地对照产品的连续构建，然后使用TestManager的报告工具分析结果。

关于suites的信息，参阅67页*Implementing Tests as Suites*的内容。

Rational TestManager 的开始（Starting Rational TestManager）

在你开始使用TestManager之前，你需要：

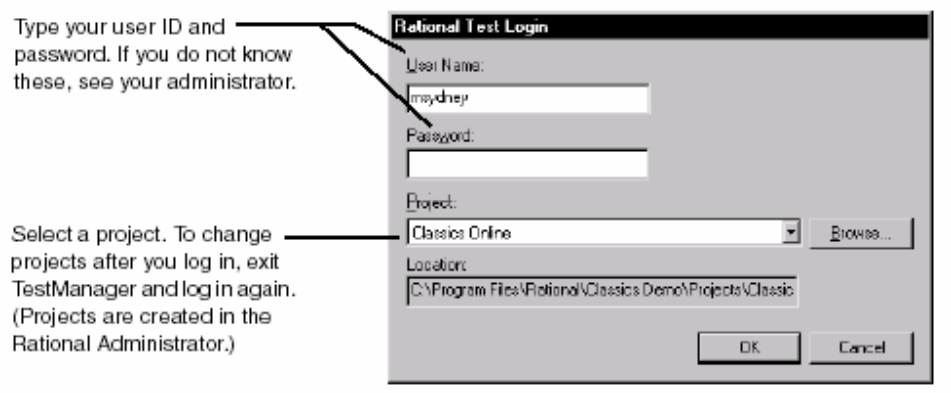
- 安装Rational TestManager。有关信息，参阅*Installing Rational Testing Products*手册。
- 一个Rational产品。有关信息，参阅*Using the Rational Administrator*手册或Rational Administrator帮助。

登录到 TestManager（Logging into TestManager）

当你登陆TestManager时，你需要提供你的用户ID和密码，这些是你的管理员分配给你的。你也可以指定项目来进入。

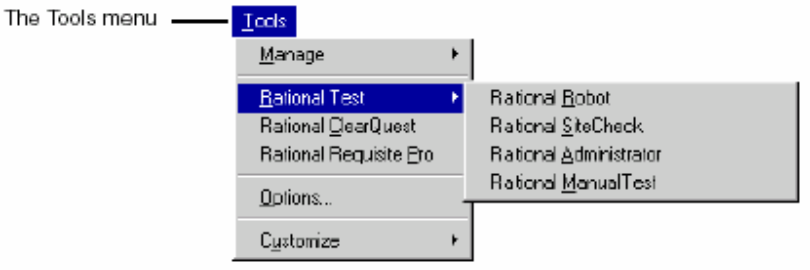
登陆：

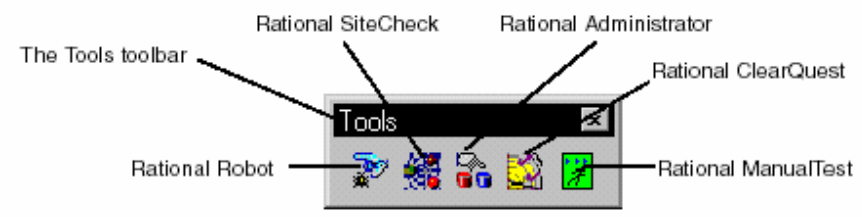
- 点击**Start > Programs > *Rational product name* > Rational TestManager**打开Rational Test登陆对话框。



从 TestManager 进入其他的 Rational 产品和组件（Starting Other Rational Products and Components from TestManager）

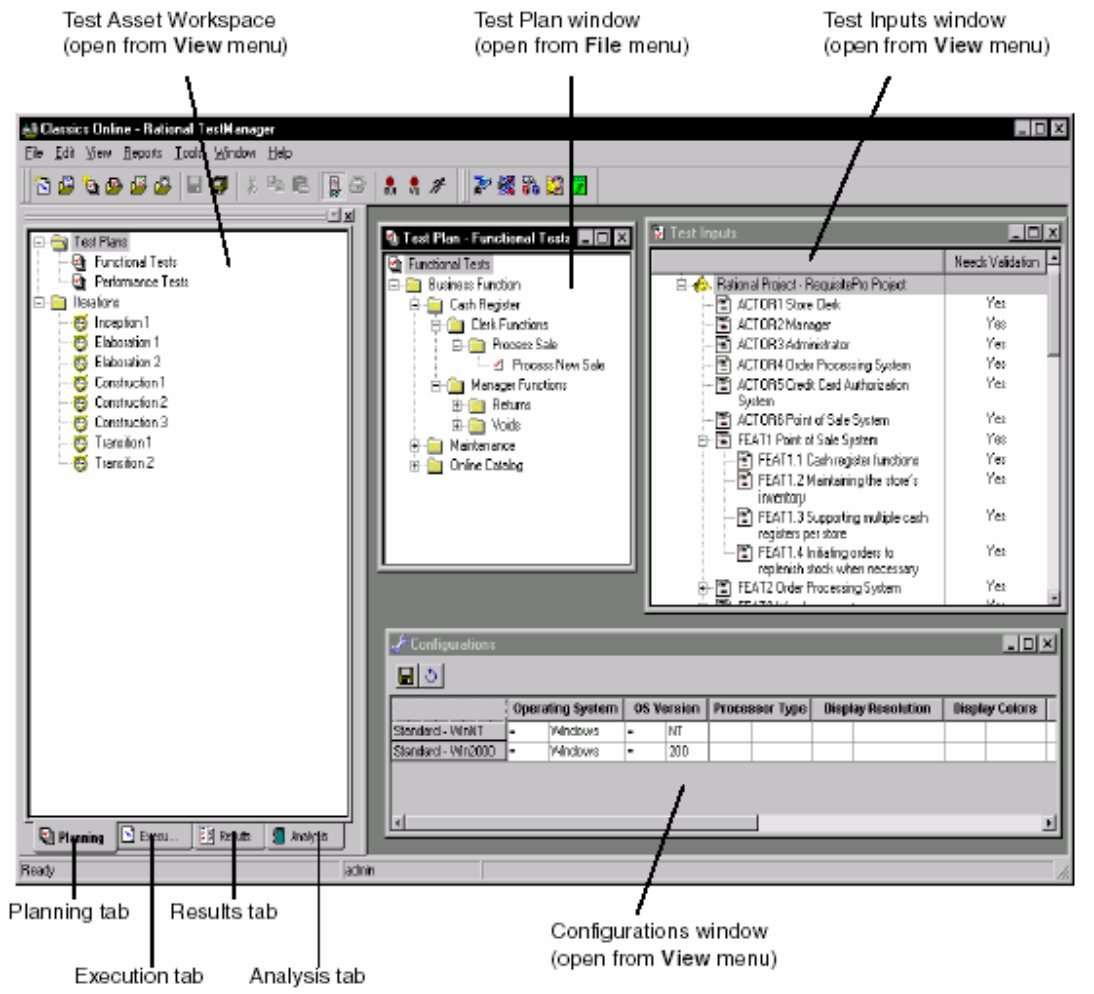
在你进入到TestManager中，你可以从其他的工具菜单或工具栏中开始其他的Rational产品和组件。





TestManager 的主要窗口（The TestManager Main Window）

下图展示了TestManager的主要窗口和它的一些子窗口：



测试资产工作区（Test Asset Workspace）

Test Asset Workspace给你项目中的测试资产不同的视图。它有4个标签：计划编制，执行，结果和分析。

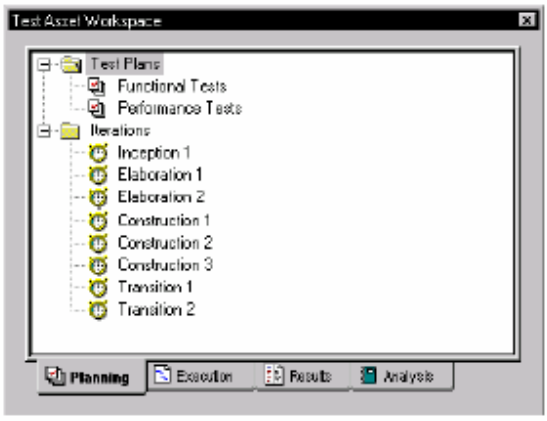
● 点击**View > Test Asset Workspace**。

右键点击在Workspace中的任何一个测试资产来显示一个捷径菜单。

右键点击窗口底部附近（在一个空区域）来允许Workspace的进入或使它浮动在主窗口中。

编制计划标签（Planning Tab）

Planning标签编列项目中的测试计划和迭代。



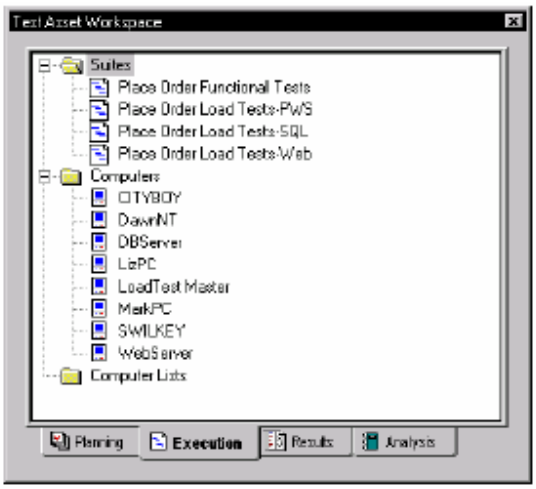
右键点击任意一个测试资产来显示一个捷径菜单。

有关测试计划的信息，参阅28页上的*Creating a Test Plan*的内容。

有关迭代的信息，参阅43页上的*Specifying When to Run Tests*的内容。

执行标签（Execution Tab）

Execution标签编列项目中的suites，测试机，和测试机列表。



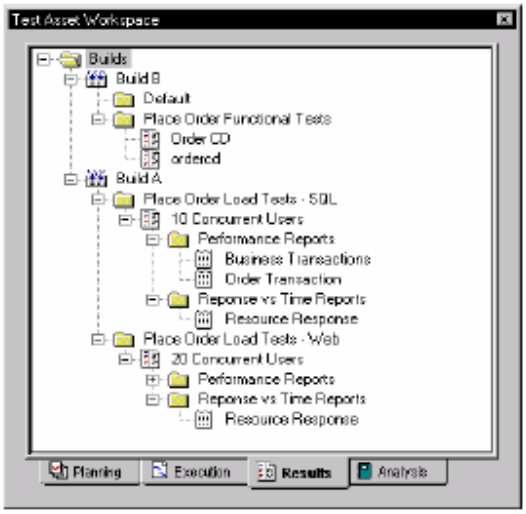
右键点击任意一个测试资产来显示一个捷径菜单。

有关suites的信息，参阅67页*Implementing Tests as Suites*的内容。

有关测试机和测试机列表的信息，参阅69页*Defining Agent Computers and Computer Lists*的内容。

结果标签（Results Tab）

Results标签编列项目中的builds，测试日志文件夹，和测试日志。

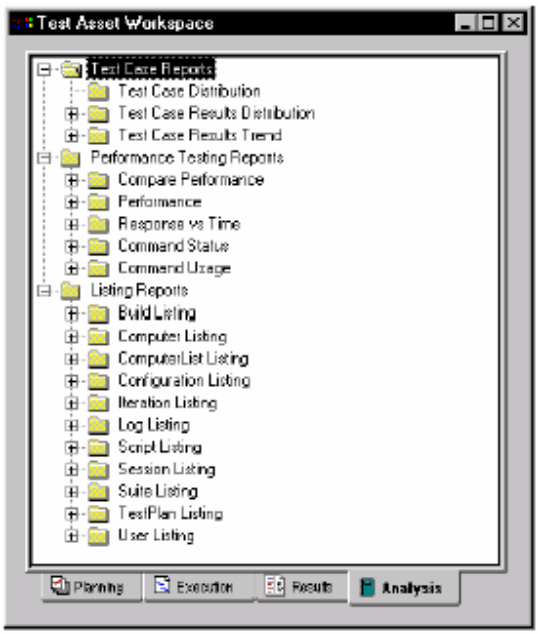


右键点击任意一个测试资产来显示一个捷径菜单。

有关builds，测试日志文件夹和测试日志，参阅127页*Evaluating Tests*的内容。

分析标签（Analysis Tab）

Analysis标签编列项目中的报告。



右键点击任意一个测试资产来显示一个捷径菜单。

有关报告的信息，参阅145页*Reporting Results*和291页*Reporting Performance Testing Results*的内容。

其他 TestManager 窗口（Other TestManager Windows）

下图所示的表编列了其他的TestManager窗口和有关这些窗口的信息搜寻。

Window	Description	See
Test Input	Shows the test inputs associated with the project.	*Identifying What to Test by Using Test Inputs* on page 24
Test Plan	Shows a test plan and all of its test case folders and test cases.	*Creating a Test Plan* on page 28
Configuration	Shows all of the configurations and configuration attributes in the project.	*Defining the Configurations to Test* on page 35
Suite	Shows all of the items contained in a suite.	*Implementing Tests as Suites* on page 67
Monitoring	Shows up-to-date information as a test case, test script, or suite runs.	*Monitoring Suites* on page 103
Test Log	Shows test logs created after you run a suite, test case, or test script.	*About Test Logs* on page 127
Reports	Shows the results of running reports.	*Reporting Results* on page 145

窗口	**说明**	**参阅**
Test Input	显示测试输入与项目的关联	24页
Test Plan	显示一个测试计划和它含有的全部测试用例文件夹和测试用例	28页
Configuration	显示所有的配置和项目中的配置属性	35页
Suite	显示包含在一组suite中所有的条目	67页
Monitoring	显示在测试用例，测试脚本，或suite执行时的更新信息	103页
Test Log	在你执行一组suite，测试用例，或测试脚本后，显示被创建的测试日志。	127页
Reports	显示报告的执行结果	145页

测试的计划(Planning Tests) 2

这一章描述如何计划我们的测试，它包括以下的内容：

☐ 关于测试的计划编制

☐ 确定测试需要的条件输入

☐ 创建测试计划

☐ 组织测试用例文件夹

☐ 创建测试用例

注意条款：对于细节过程，参见 **TestManager** 的帮助文档。

◆关于测试计划的编制（About Test Planning）

测试计划编制的动机就是下面这个问题的答案：“在我们需要达到 **agree-upon** 质量目标的测试时，我们该做什么？”

当我们完成测试计划时，你已经有了一份限定了你将要测试什么的测试计划。

测试计划的编制是跨时间的，你偶尔可以加一些要测试的东西到它里面去。

在一个团队中，有不同的成员和角色，就如产品（项目）经理，分析人员，测试人员和开发人员，会提出你需要定义的一些新的测试用例，你需要测试的一些新的情况，以及你仅仅只需要学习的新的特征。

换句话说，你不在测试的开始阶段制定测试计划，那么你的目标是停滞和不灵活的。

测试计划是一个迭代定义的不断完善的测试资产。

在 **TestManager** 中，一个测试计划包含很多测试用例，而测试用例在测试用例文件夹中被组织起来。

在 **TestManager** 中，测试计划的编制有以下几个主要工作组成：

☐ 收集并确定测试输入

☐ 创建测试计划

☐ 创建测试用例文件夹

☐ 创建测试用例

☐ 限定你需要测试时的资源配置

☐ 定义你的迭代，在你执行测试时

◆确定测试需要的测试输入（Identifying What to Test by Using Test Inputs）

当你首先开始编制测试计划时，你的目标应该是创建一个测试列表，让他的里面包含你所有需要测试的东西。

一个方法是在你编制计划开始阶段，去找那些可利用的资源，它们能够帮助你决定需要测试的是什么。

举个例子，你可以看到：

☐ 原型

☐ 软件构架

☐ 功能描述

☐ 需求分析

☐ 可视模型

☐ 源代码文档

☐ 需求变化

作为一个测试人员可能看到的那些资源，帮助你决定:“你需要测试什么？”这些资源（**materials**）就是你的测试输入。他们是测试计划编制阶段时的输入。他们帮助你构建测试列表。

在你构建了你的测试列表，也就是确定了你需要的测试之后，你就可以创建你的测试用例了。测试用例限定你以测试输入为基础，将要进行的测试。于是，你能够把测试输入和测试用例结合起来对测试目的进行跟踪，在搭建起这些联系时，你能够更容易地跟踪到这些测试输入的改变，而这些改变可能引起测试用例的变化，或是改变这些测试用例的实施。对于该内容，可见 45 页的**利用测试输入搭建跟踪（Trace ability）。**

你也可以执行报告（Run reports）来确定这些将用例和用例实施联系在一起的测试输入，以及确定那些已经执行过的测试用例。举个例子，分析人员可能对基于需求方面的报告感兴趣，而设计人员对基于模型元素方面的报告感兴趣。对于相关报告的内容，可见 145 页的**报告的结果**

（Reporting Results）

几乎任何的事物都能够作为测试的输入，而 **TestManager** 提供了内置的测试输入类型，当然，你也可以定义自己习惯的测试输入的类型，以作为你测试环境的需求。

为看到这些可利用的测试输入：

点击 **View** 菜单下的 **Test Inputs（View > Test Inputs）**打开测试输入窗口。

内置测试输入的类型（Built-in Test Input Types）

TestManager 有两种内置的测试输入的类型：

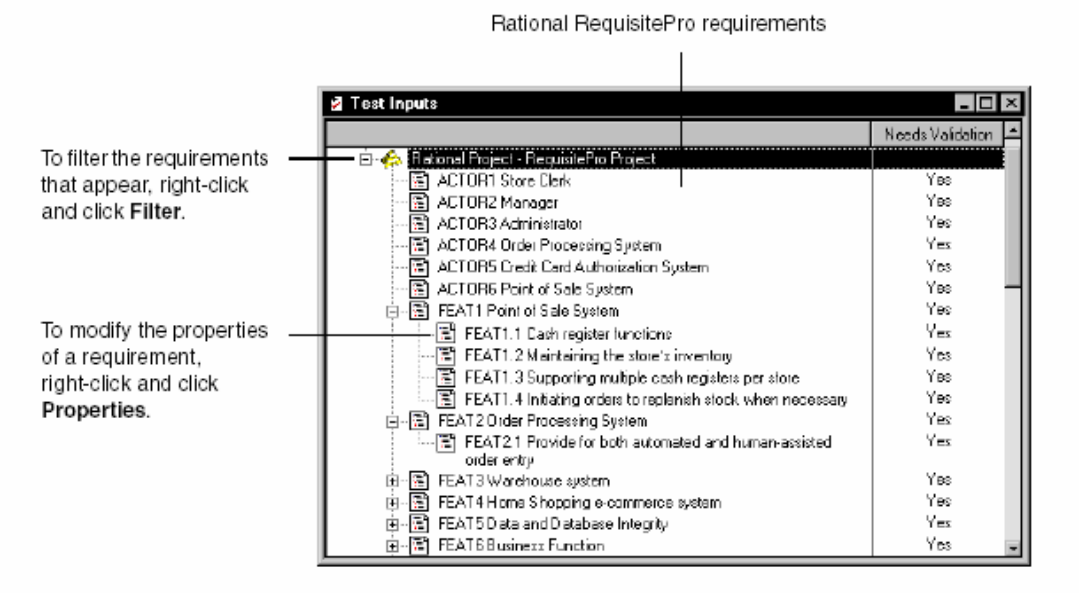
☐ 在一个 **Rational RequisitePro** 工程中的需求类型。

☐ 在一个 **Rational Rose** 可视模型中的元素类型。

来自 Rational RequisitePro 中的需求（Requirements from Rational RequisitePro）

你可以方便地利用 **RequisitePro** 中的需求作为你测试的输入。你或者管理人员可以利用 Rational Administrator 将一个 RequisitePro 中的工程和一个 Rational 中的工程联系起来，当你这么做时，这些需求会自动出现在你测试输入的窗口中，不过，你得先登陆到 TestManager 的工程中。然后，你可以创建一个联合体，在需求与测试用例之间。你也可以利用其他的一些 RequisitePro 工程中的需求作为你的测试用例。

注意条款：在 RequisitePro 中，需求自己被创建和管理，但你能够从 TestManager 中修改需求的属性。下图展示了一些来自于一个 RequisitePro 工程中的需求。当你打开测试输入的窗口时，你就可以这些需求了。



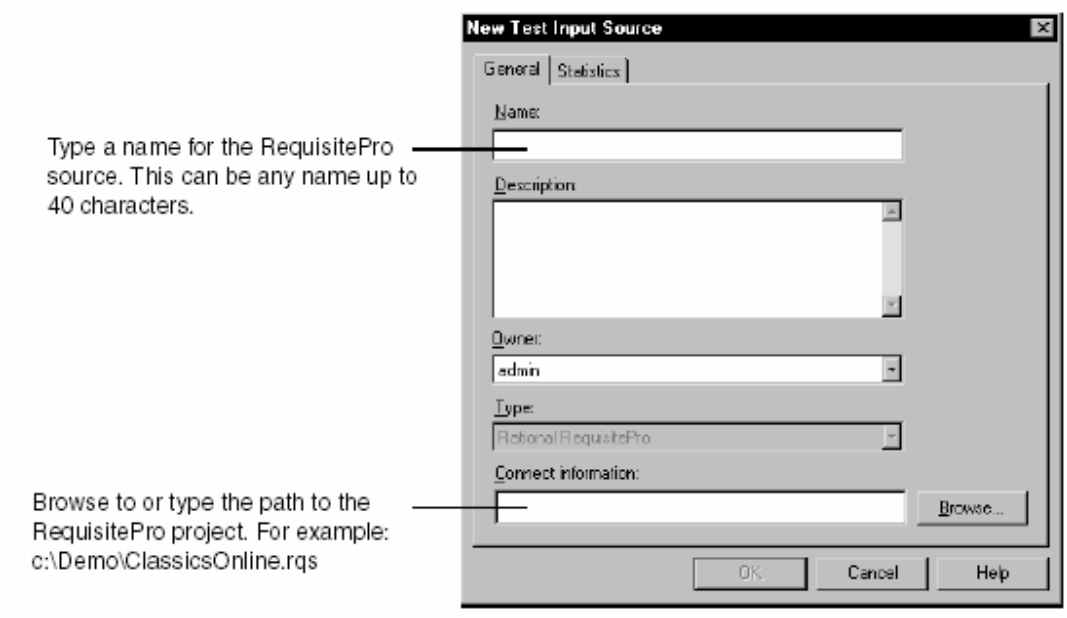
即使你没有将 RequisitePro 工程和 Rational 中的工程联系起来，你也一样可以利用一个或者更多的 RequisitePro 工程来作为你制定测试输入的资源。然而，既然是那样，你必须将 RequisitePro 工程与 TestManager 一起注册。

注册一个 RequisitePro 工程作为测试输入的资源：

1. 点击Tools菜单中的Manage项，打开Test Input Types。（**Tools > Manage > Test Input Types**）
2. 打开Rational RequisitePro，并点击编辑（**Edit**）。

注意：如果**Edit**不可用，那就是你还没有管理员权限。对于该内容，可参见Rational管理员使用手册（**Using the Rational Administrator manua**）或他的帮助文档。

3. 点击**Sources**标识和**Insert**项。



惯用的测试输入的类型（Custom Test Input Types）

TestManager还支持除RequisitePro需求和Rational Rose模型元素之外的测试输入的类型。举个例子，你可能想使用Microsoft Excel的电子数据表中的数值作为测试的输入。

如果你想知道在资源文件发生变化时，哪些需要的测试会被改变或返回，你就可以定义C++中的工程文件作为测试输入的类型。

对于TestManager支持的一个扩展类型，就需要你这个测试组中的某个人写一个习惯的测试输入转换器。一个转换器是一个为TestManager必要时调用的动态连接库（DLL），和某种需要的函数。就像这个例子，当连接到测试输入资源或从该资源断开的时候。除了要你的这个测试组写转换器外，Rational公司的软件或者他的合作伙伴有一些可利用的惯用转换器。（对于该内容，可参见Rational TestManager扩展机制参考手册（Rational TestManager Extensibility Reference manual）.

在实施DLL之后，你需要在TestManager中定义新的类型，并且需要注册资源。

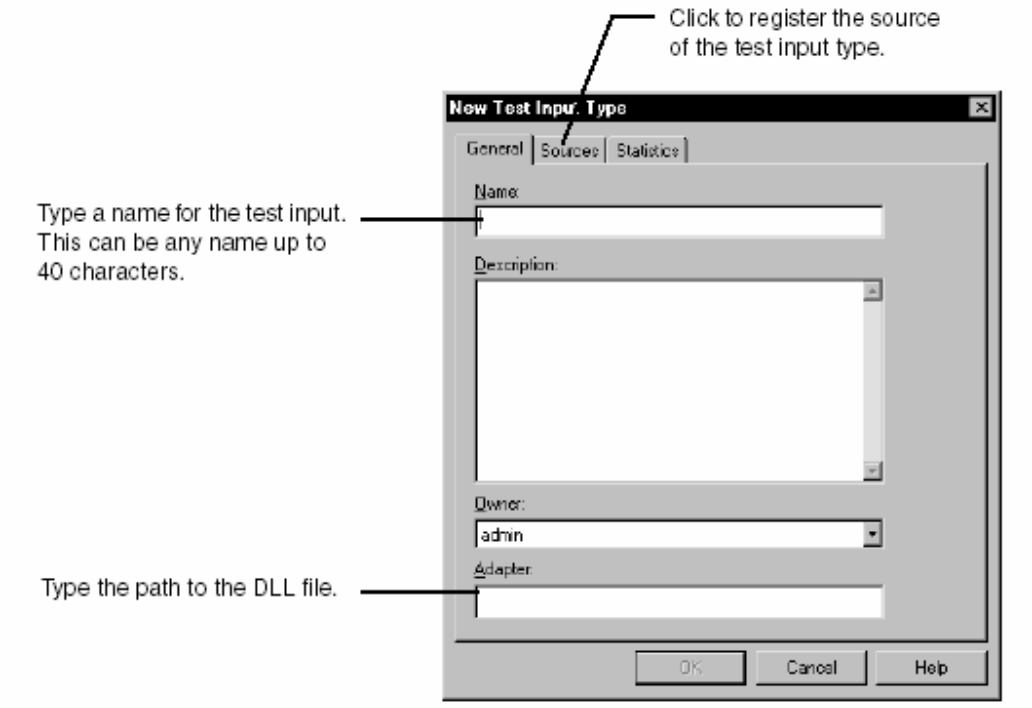
有如下操作：

☐ 点击**Tools > Manage > Test Input Types**。

☐ 点击 **New**。

注意事项：如果点击**New**无效，那就是你还不具有管理员的权限。

以上内容可参见**Rational管理员使用手册**或是他的帮助文档。



有关此方面更多的信息，可参见TestManager帮助文档索引中的新测试输入类型的注册（**test input types:registering new**）

在TestManager中定义新的类型，并且需要注册资源后，资源显示在测试输入的窗口中。

（**View > Test Inputs**）

◆创建测试计划（Creating a Test Plan）

在**TestManager**中，一个测试计划是一个Rational Test数据存储的资产。

你可以在一个项目中有一个或多个测试计划，可以用任何方法来组织它们，而这些方法应该为你的测试情况做出判断。

例如，你有了一个完整测试的项目的测试计划，或者是测试一个项目的主要模块的测试计划。

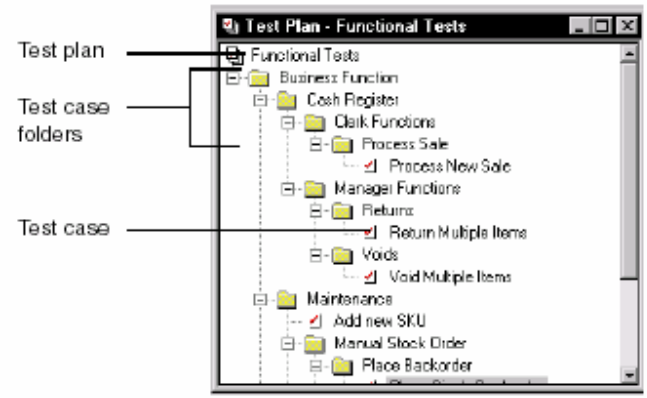
每个测试计划都能够包含复合的测试用例文件夹和测试用例。

在测试计划窗口中（Test Plan window），你与测试计划一起工作。

打开测试计划窗口：

●在测试资产工作区的计划编制标签（**Planning** tab）中，右键点击测试计划，点击**Open**。

在下面的例子中，名为功能测试的测试计划包含了复合的测试用例文件夹和测试用例。



你可以运行报告来观察一个项目的测试计划的信息。更多有关报告的信息，请参见 *Reporting Results* 中 145 页的内容。

创建测试计划（Creating Test Plans）

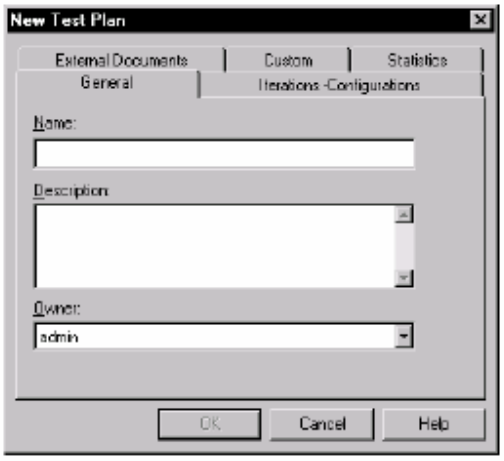
TestManager为你提供一个名为Test Plan1（默认）的空测试计划，你可以使用它开始你的计划编制。当然，你也可以创建自己命名的测试计划。

开始创建新的测试计划：

●在测试资产工作区（**Test Asset Workspace**）的计划编制标签（**Planning** tab）中，右键点击测试计划，点击**New Test Plan**。

注意事项：如果**New Test Plan**菜单命令不可用，说明你还不具有管理员的权限。更多信息，请参见the *Using the Rational Administrator*手册或帮助。

如图所示的新建测试计划：



测试计划的属性（Properties of a Test Plan）

一个测试计划具有很多的属性，这里包括：

●测试计划的名称（必需的）。

●测试计划的描述。

●测试计划所有者，请参见*Specifying the Owner*中34页的内容。

●对于测试计划的配置关联。更多信息请参见*Defining the Configurations to Test*中35页的内容。

●对于测试计划的迭代关联。更多信息请参见*Specifying When to Run Tests*中43页的内容。

●对于测试计划的外部文档关联。比如，你能够建立一个记录有关你测试计划细节信息的Word文档的关联。

测试计划的名称是必需的。对于所有其他的属性，你能够在第一次创建测试计划时添加它们，或者在晚些时候添加或修改它们。

注意事项：你能够定制一些测试资产（Test assets）的属性，在**Tools**工具栏中选择**Customize**菜单（**Tools > Customize**）。

任何对于测试计划的迭代关联和配置关联，系统自动建立与新测试用例文件夹的关联，这些测试用例文件夹是直属与测试计划。

换句话说，如果你在测试计划的窗口中创建直属与一个测试计划的测试用例文件夹，这个新的文件夹将继承所有的对于该测试计划的迭代关联和配置关联。

你能够容易的改变那些不合适的文件夹的关联。

开始改变测试计划的属性：

●在测试计划窗口或者是测试资产工作区的计划编制标签（**Planning** tab）中，右键点击测试计划，点击属性（**Properties**）菜单。

你也可以复制一个已存在的测试计划和它的所有属性。

◆组织测试用例文件夹（Organizing Test Cases with Folders）

在一个测试计划里，你可以创建测试用例文件夹来分层次组织你的测试用例。你可以用任一种方法来组织测试用例，这些方法应该为你的测试结果做出判断。

例如，你已经创建了一个测试用例文件夹：

●对于你项目组中的每一个测试人员。

●对于测试每一个种类或类型（单元测试，功能测试，执行测试和其他）。

●对于系统的每一个用例（use case）。

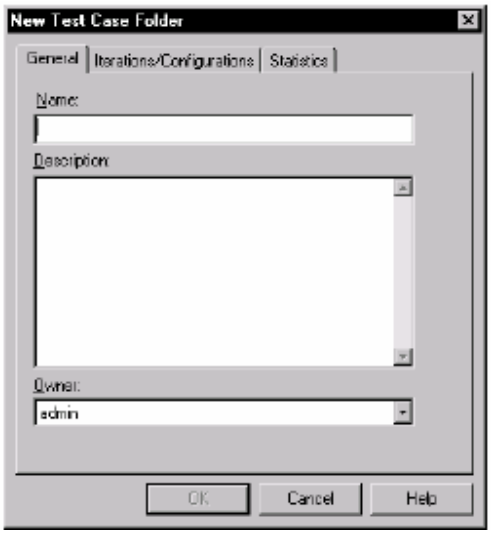
●对于应有程序的每一个主要模块。

●对于测试过程的每一个阶段。

你可以在测试用例文件夹（test case folders）中再创建一个 test case folders。例如，你已经有了一个测试人员的文件夹，然后那个测试人员可以为每一个需要测试的功能块继续创建文件夹，当然是在已有的这个文件夹中。

创建一个 test case folder：

●在测试计划窗口中，右键点击测试计划或测试用例文件夹（test case folder），点击**Insert Test Case Folder**。



就像测试计划一样，一个测试用例文件夹（test case folder）有确定的属性。

这里包括：

●文件夹的名称（必需的）。

●文件夹的一个描述。

●文件夹的所有者。更多信息，请参见*Specifying the Owner*中34页的内容。

●文件夹的配置关联。更多信息请参见*Defining the Configurations*中35页的内容。

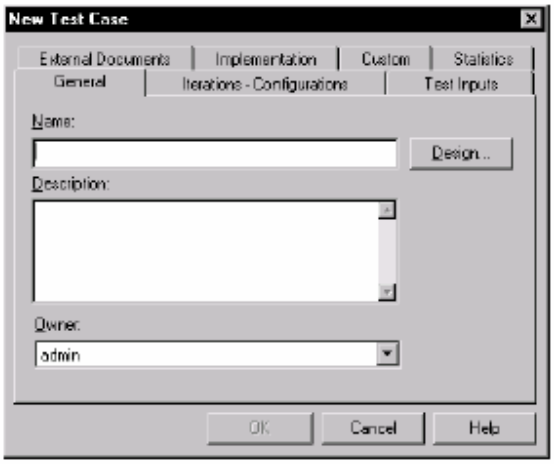
●文件夹的迭代关联。更多信息请参见*Specifying When to Run Tests*中43页的内容。

文件夹的名称是必需的。对于其他所有的属性，你可以在第一次创建文件夹的时候来添加它们，或者在晚些时候添加或是修改这些属性。

任何对于一个测试用例文件夹（test case folder）的迭代关联和配置关联，系统自动建立与新文件夹和测试用例的关联，这些测试用例是测试用例文件夹（test case folder）的直接子集。
换句话说，如果你在测试计划的窗口中创建直属与一个test case folder的测试用例，那个新的测试用例将继承所有的对于该test case folder的迭代关联和配置关联。你还能够容易的改变那些不合适的测试用例的关联。

◆创建测试用例（Creating Test Cases）

测试用例集中于测试计划。在你定义了你的测试输入和决定自己如何测试之后，你就可以创建自己的测试用例了。
（The *test case* is the test asset in TestManager that answers the question, “What am I going to test?” You develop test cases to validate that the system is working the way that it's supposed to work and is built with the quality necessary before you can ship it.）
测试用例作为Test Manager中的测试资产，也是“我将要测试什么？”的问题的答案。你开发测试用例为使系统有效的工作在假定的方式下，以及在你成品之前对于质量的必然的建立。
一个测试用例经常是属于一个测试计划中一个测试用例文件夹的。
你可以用两种途径来创建测试用例：
●在测试计划窗口中，右键点击一个测试用例文件夹，点击**Insert Test Case**。
●在测试输入窗口中，右键点击一个测试输入，点击**Insert Test Case**。



注意事项：你可以运行若干类型的测试用例报告来收集相关项目的测试用例信息。更多关于报告的信息，请参阅*Reporting Results*中的145页的相关内容。

测试用例的属性（Properties of a Test Case）

一个测试用例有很多属性。

这里包括：

●测试用例的名称（必需的）。

●测试用例的一个描述。

●测试用例的所有者。更多信息，请参阅*Specifying the Owner*中34页的内容。

●对于测试用例的配置关联。更多信息，请参阅*Defining the Configurations to Test*中35页的内容。

●对于测试用例的迭代关联。更多信息，请参阅*Specifying When to Run Tests*中43页的内容。

●对于测试用例的任何测试输入关联。更多信息，请参阅*Setting up Traceability Using Test Inputs*中45页的内容。

●对于测试用例的任何外部文档的关联。

●测试用例手册和测试用例的自动执行。这里有可以运行的实际的测试脚本。更多信息，请参阅*Implementing Tests*中55页的内容。

●测试用例的设计（换句话说，当测试用例被实施时，用例执行的步骤和检验点）。更多信息，请参阅*Specifying the Testing Steps and Verification Points*中49页的内容。

●前置条件，后置条件，和测试用例的验收标准。更多信息，请参阅*Specifying Conditions and Acceptance Criteria of Test Cases*中49页的内容。

测试用例的名称是必需的。对于其他所有的属性，你可以在第一次创建测试用例的时候添加，或是在晚些时候添加或修改它们。

注意事项：你可以定制一些测试资产的属性，选择**Tools**工具栏中的**Customize**菜单（**Tools > Customize**）。

改变测试用例的属性：

●在测试计划窗口中，右键点击测试用例，点击**Properties**。

指定所有者（Specifying the Owner）

你可以在新测试用例（New Test Case）对话框的**General**标签中的**Owner**列表中选择测试用例的所有者。

在**Owner**列表中包含测试用例的用户ID，这些用户ID是在早些时候通过**Rational Administrator**添加到项目中的。（更多信息，请参阅*Using Rational Administrator*的手册和帮助。）

所有者对于计划的编制和目的的跟踪是很重要的。例如，你可以运行一个测试用例的分布记录，用以了解测试用例针对所有者的分布。更多有关记录的信息，请参阅*Reporting Results*中145页的内容。

对测试进行配置定义（Defining the Configurations to Test）

你能够使用配置（*configurations*）来设立测试用例，以便他们在特定的硬件和软件支持下的电脑上自动运行。举个例子，你可能需要确定一个测试用例，他能够成功的在某种操作系统和浏览器下运行。你可以在测试中对每一个操作系统和浏览器分别进行配置。或者，你可能需要在某种混和（combinations）操作系统和浏览器一起工作的情况下进行测试。

For example, a test case might need to run successfully on the following combinations of an operating system and a Web browser:

举个例子，一个测试用例可能需要在以下的混和（combinations）操作系统和Web浏览器上成功运行：

●Windows 2000 and Internet Explorer 4

●Windows 2000 and Netscape 4

●Windows NT 4 and Internet Explorer 4

●Windows NT 4 and Netscape 4

这些混和体的每一种都是你需要测试的一个配置。在你用TestManager定义了配置之后，你可以将配置与测试用例关联起来，从而创建配置的测试用例（*configured test cases*）。然后，你就可以在合适的电脑上来运行这些配置的测试用例。

在你设置配置时，主要有一下的四个步骤：

1 由于许多属性并非TestManager已经内置的属性，所以要定义常规的常规的属性和他们可能的值。（参阅*Defining Configuration Attributes and their Values* 36页的内容。）

例如，浏览器就不是内置属性。你要创建一个命名为“浏览器”的属性，而他的属性值则是“Internet

Explorer 4”和“Netscape 4”。

2 为你将要运行一个“配置的测试用例”（configured test case）的电脑创建一个命名为“tmsconfig.csv”的文件。这个文件包含为那台电脑设置的常规属性和适当的属性值。（更多信息，参阅“*Setting Up Custom Attributes in tmsconfig.csv* ”38页的内容。）

例如，假设一台电脑使用“Internet Explorer 4”，那么，你必须在电脑上创建一个“tmsconfig.csv”文件，用来指明这个浏览器就是该电脑上使用的浏览器。

3 定义你需要测试的特定配置。（参阅“Defining the Configurations You Need to Test”39页的内容。）

4 每一个配置与一个测试用例相关联，来创建一个“配置的测试用例”（configured test case）（参阅“Associating a Configuration with a Test Case”42页的内容。）

例如，你的测试用力需要运行在“Windows 2000”和“Internet Explorer 4”中，你要将这些配置与测试用例关联起来。

定义属性和配置是一个迭代的过程，你将最大可能地继续在整个测试项目中去添加和精炼这些属性和配置。

定义配置的属性和属性的值（Defining Configuration Attributes and their Values）

当你开始计划你的测试类型时，考虑到如何兼有这些恰当的配置属性（例如，“浏览器”和“操作系统”），以用来定义与测试形成对照的配置（例如，“浏览器”＝Netscape 4和“操作系统”＝Windows）。

记住一点，你可以在运行了你的测试用例后，来运行与这些配置相对照的报告。例如，你可以创建并运行一个测试用例的结果分布报告，而这个分布结果覆盖了这些配置。由于这些报告是可用的，你必须适当地定义你的属性。这个过程是迭代的。遍及整个测试项目，你将有可能继续去扩充和精炼这个（属性）列表。

例如，当你开始项目时，你可能仅仅需要去测试“Internet Explorer 4” 和“Netscape 4”。在项目进行的晚些时候，分析员可能决定你也需要测试“Internet Explorer 5”。你可以打开命名为“浏览器”的配置属性，添加“Internet Explorer 5”作为一个新的属性值。

查看内置的配置属性（Viewing Built-in Configuration Attributes）

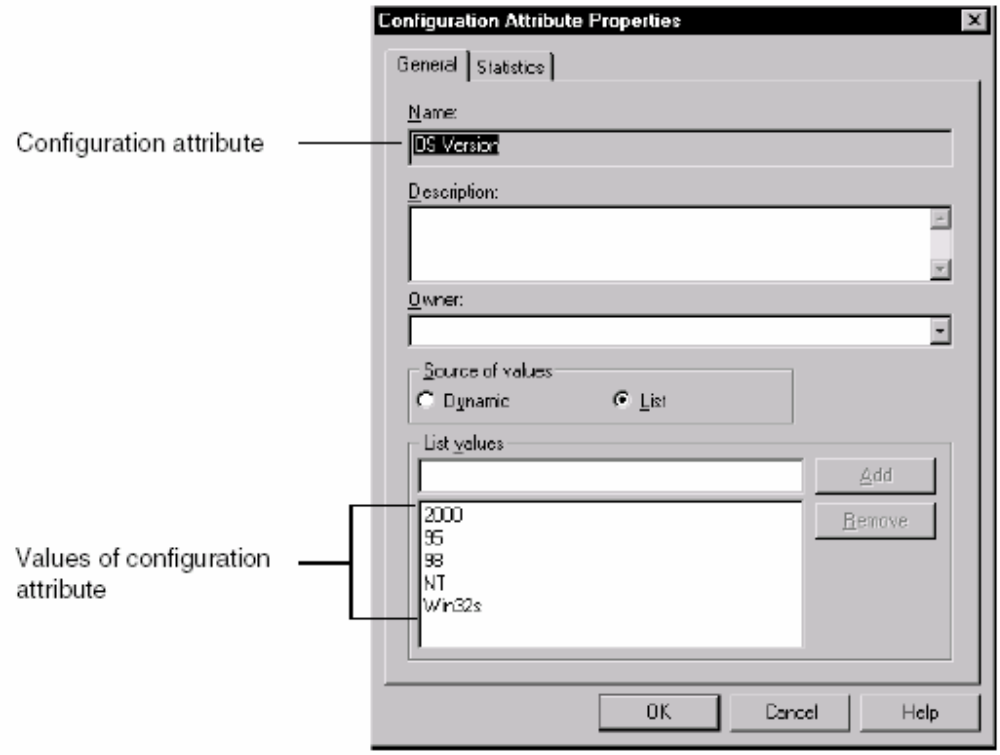
TestManager 伴随着许多的内置属性。这些属性包括：显示颜色，显示分辨率，内存，操作系统，OS service pack，OS版本，CPU大小，CPU数目，以及CPU类型。

查看这些内置属性和属性的值：

1 点击**Tools > Manage > Configuration Attributes**.

2 查看每一个属性的**properties**，选择属性并点击**Edit**。

任何被定义的属性值出现在列表中的属性值区域中，像下面图示的这个例子：



如果一个属性在列表中有了一个值，那你可以在创建一个配置时来选择那个值。当你运行一个配置的测试用例时，TeatManager检测测试用机并决定是否与该值存在一个匹配。

注意事项：如果对于属性的值，其来源设置为**Dynamic**，你可以在你创建一个配置时写入一个值。为了确定测试用机的适合值，可在该测试用机上运行一个测试脚本，并在可Test Log窗口中查看本测试用机的配置。更多信息，参阅“Viewing Events Details”133页内容

定义常规的配置属性（Defining Custom Configuration Attributes）

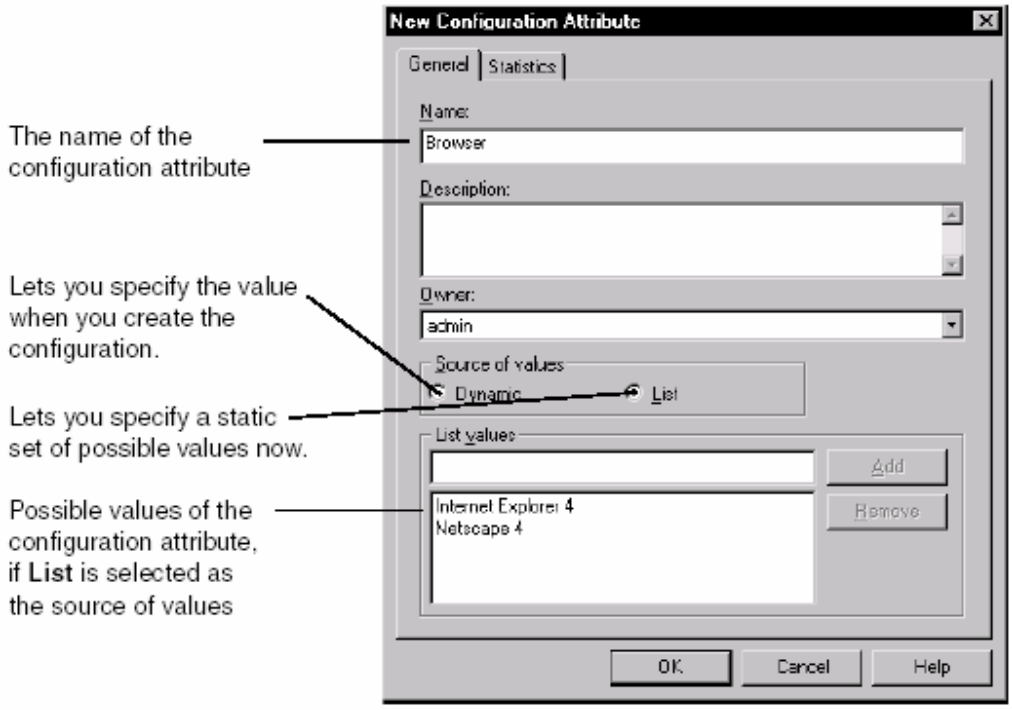
要使用一个非内置的配置属性，你将定义一个常规属性。

定义一个常规的配置属性：

● 点击**Tools > Manage > Configuration Attributes**。点击**New**按钮。

注意事项：如果**New**按钮不可用，说明你还不具有管理员的权限。更多信息。参阅Using the

Rational Administrator manual或相关帮助。



如果你在一个配置中使用一个常规属性，你需要为你将要运行该“配置”（configured ）的测试用机创建一个命名为“tmsconfig.csv”的文件。更多信息，参阅下一节，Setting Up Custom Attributes in tmsconfig.csv。

在 tmsconfig.csv 文件中设置常规属性（Setting Up Custom Attributes in tmsconfig.csv）

在前一节已经说明，如果你已经定义了常规属性和他的值，那么你就需要在你将要运行一个配置的测试用例的测试机上创建一个tmsconfig.csv文件。对于那台测试机而言，这个文件包含了恰当的属性和属性值。在你运行一个配置的测试用例，只有当常规和内置属性匹配与该测试用例的配置时，TestMannager检查该文件并运行此测试用例。

例如，假设你有了一个仅仅运行在已安装了Internet Explorer 4的测试机上的配置的测试用例。你已经定义了常规属性，并点击**Tools > Manage > Configuration Attributes**，再点击按钮**New**，从而定义了他的值，就像前一节描述的那样。你现在就需要创建一个tmsconfig.csv文件，且包含了关于那台测试机的属性和属性值。

设置常规属性和属性值：

1 创建名为“*tmsconfig.csv*”的文件。你可以通过Excel或通过任何一种文本编辑器来创建这个文件。（要确定将文件保存为csv格式。）

2 向文件中添加成对的合适属性和值。

在这个例子中，该测试机运行的是Internet Explorer 4。因此，这个配置文件包含：

```
Browser,Internet Explorer 4
```

在tmsconfig.csv文件中那些属性和值成对的用例必须匹配与你在TestManager中定义的用例，这些包含常规属性和属性值的用例。

3 将文件保存为`tmsconfig.csv`。

4 在恰当的本地或代理测试机上，将该tmsconfig.csv文件异动到Rational Test文件夹中。

如果一个测试用例的配置使用常规属性，而这些属性完全匹配与测试机上tmsconfig.csv文件中定义的属性，则这个配置的测试用例将仅仅运行该测试机上。

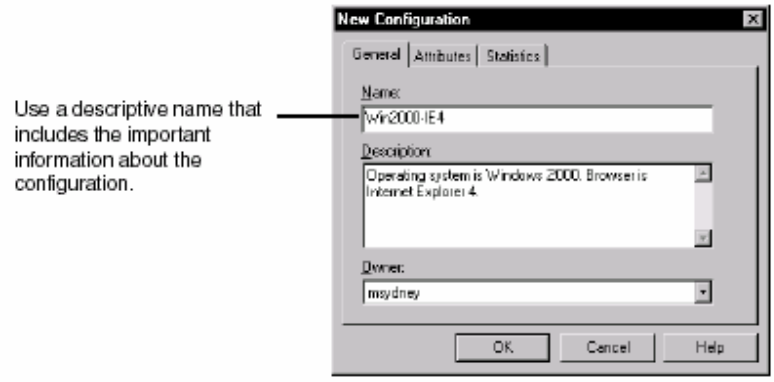
定义你需要测试的配置（Defining the Configurations You Need to Test）

现在你已经定义了配置的属性和属性值，你就可以定义你需要测试的配置了。这个过程是迭代的。通过进行测试的项目，你将可能继续扩充和精炼这个列表。

定义一个配置：

1 点击**Tools > Manage > Configurations**。点击**New**按钮。

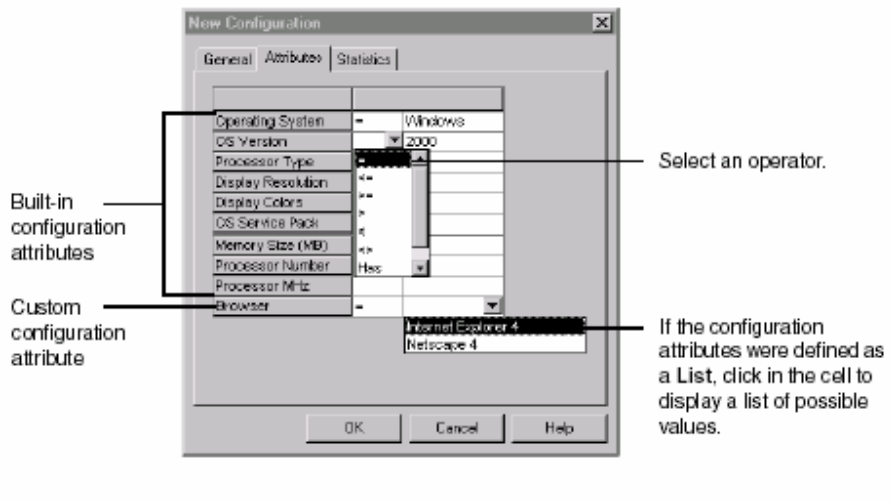
注意事项：如果**New**按钮不可用，那么你还不具有管理员的权限。更多信息，参阅Using the Rational Administrator手册或帮助。



在这副图中，配置的名称是Win2000-IE4。这个名称说明了测试机上将被用来进行测试的配置，

显然他是一个Windows 2000和Internet Explorer 4的混合体。

2 点击**Attributes**标签。



在这个标签中，你可以：

●为每一个合适的内置配置属性选择一个属性值（在此例中，是操作系统和OS版本）。

●从你已创建配置属性时定义的值的列表中，为每一个常规配置属性选择一个属性值（在此例中，为浏览器）。

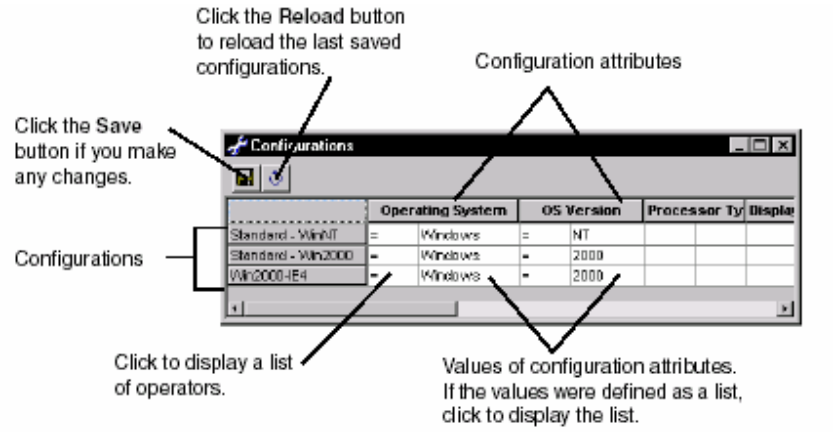
注意事项：如果配置属性被定义为动态的，你将需要写入一个属性值。对于常规属性，有效的属性值应在测试机的tmsconfig.csv文件中。对于内置属性，为测试机确定合适的属性值，在测试机上运行任何测试脚本，并可在Test Log的Log Event窗口中查看该测试机的配置。更多信息，参阅Viewing Events Details133页的内容。

查看并编辑你的配置（Viewing and Editing Your Configurations）

你可以容易地查看并编辑任何存在与Configurations窗口中的配置。

打开Configurations窗口：

●点击**View > Configurations**。



Configurations窗口打开，你就可以由**Edit**菜单插入配置的属性和相关的配置，或者，右键点击窗口中的一行。

将一个配置与一个测试用例相关联（Associating a Configuration with a Test Case）

在你创建了配置后，你可以将一个配置与一个测试用例相联系，从而创建一个配置的测试用例。你需要使一个功能块可以有效地工作在多种配置下，在这个时候，配置的测试用例是很有用的。例如，假设你有一个测试用例，“关闭应用程序。”你需要有效地将这个测试用例通过两种配置：Windows 2000与Internet Explorer 4，Windows 2000 与 Netscape 4。你应该创建两种配置的测试用例与主要的测试用例相关联。要使这些测试用例通过，就需要所有的配置的测试用例都通过。

在运行配置的测试用例之后，你可以创建一个测试用例的结果分布报告，这个报告被过滤的只含有你感兴趣的那些特定的配置。更多有关报告的信息，参阅*Reporting Results*145页的内容。

你可以用下面的几种方法来建立配置与测试用例的关联：

●在创建一个新的测试用例时，在New Test Case的对话框中点击**Iterations – Configurations**标签。

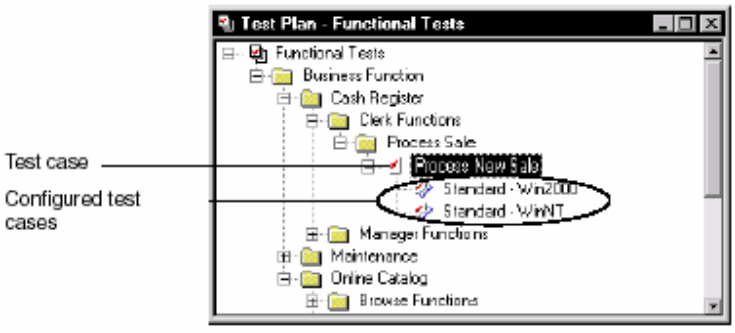
●在对一个现存测试用例的属性进行编辑时，在Test Case Properties对话框中点击**Iterations -Configurations**标签。

●在Test Plan窗口中，右键点击一个测试用例，并点击**Associate Configuration**。选择配置进行关联。

你也可以将配置与测试计划和测试用例文件夹相关联。当你建立了一个配置与一个测试计划或

文件夹的关联时，这些配置会自动地与所有的直属与这个测试计划或文件夹的测试资产相关联。当你在Test Plan窗口中关联了一个配置时，你也可以将此配置与该测试计划或文件夹的所有现存子集相关联。

配置的测试用例会出现在Test Plan窗口中测试用例的下面：



在执行测试时进行指定说明（Specifying When to Run Tests）

许多测试组织计划比之在任意给定时间内实际执行具有更多的测试用例。

你可以在TestManager中创建所有的测试用例，然后使用迭代去确定你实际需要执行和通过的那些特定的测试用例。

在一个项目期间，一个迭代是一个被定义的时间跨度。一个迭代的结束是典型的一个主要的项目里程碑。在一个迭代中，产品符合一个指定的质量标准来达到一个里程碑。质量标准被测试用例定义，而这些用例是必须通过的。在许多组织中，测试人员与一个分析人员或项目经理一起工作来确定需要通过的测试用例。

例如，在项目的初始阶段，你开始创建能够为该系统考虑到的所有的测试用例。分析人员查阅你的测试计划并说明用例1，2，3和8 对于”Construction 2 iteration”是重要的。你或者分析人员进入TestManager中并建立这4个测试用例与“Construction 2 iteration”的关联。

在你进行测试期间，你提供另一个测试用例。分析人员决定用例对于“Construction 2”是一个重要的测试用例，于是你向这个用例添加迭代。

TestManager提供一个基于RUP的初始迭代设置。（对于这些迭代的一个描述，参阅TestManage帮助和RUP相关文档。）你可以使用这些迭代，或添加你自己的，而添加的这些迭代是基于你的机构形成的观念的。

创建并编辑迭代（Creating and Editing Iterations）

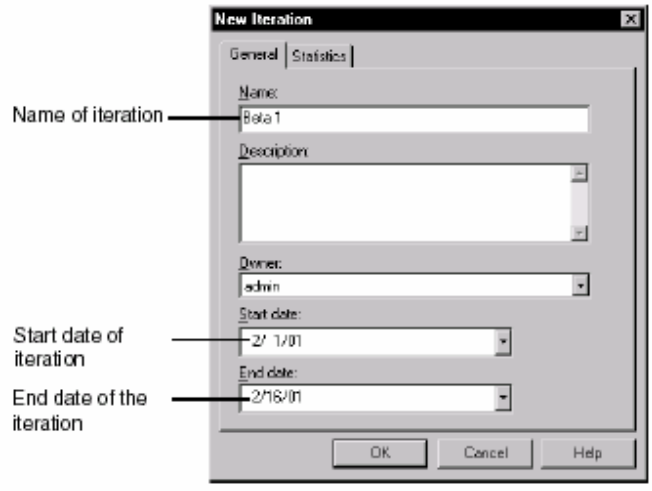
创建或编辑迭代：

1 选择**Tools > Manage > Iterations**。

2 点击**New**创建一个新的迭代，或选择一个现存的迭代并点击**Edit**。

注意事项：如果**New** 和**Edit**按钮不可用，说明你还不具有管理员的权限。更多信息，参阅*Using the Rational Administrator*手册或帮助。

你也可以右键点击**Iterations**或是Test Asset Workspace的**Planning**标签中的一个特定的迭代。



建立迭代与测试用例的关联（Associating Iterations with a Test Case）

你可以用下面的方法来建立一个迭代与一个测试用例的关联：

●在你创建一个新的测试用例时，点击New Test Case 对话框中的**Iterations – Configurations**标签。

●在你编辑一个现存的测试用例的属性时，点击Test Case Properties对话框中的**Iterations - Configurations**标签。

●在Test Plan窗口中，右键点击一个测试用例并点击**Associate Iteration**。

你也可以建立迭代与测试计划和用例文件夹的关联。在你将一个迭代与一个测试计划或文件夹关联起来的时，这个迭代会自动地与所有的直属与这个测试计划或文件夹的测试资产相关联。

当你在Test Plan窗口中关联了一个迭代时，你也可以将此配迭代与该测试计划或文件夹的所有现存子集相关联。

你可以执行关联与一个特定迭代的所有的测试用例。更多信息，参阅*Running a Test Case*91页的内容。

你可以创建并执行分布覆盖迭代的测试用例报告。例如，你创建分布覆盖一个特定迭代的测试用例结果分布报告。对于给定的迭代，当定义你的质量验收标准的所有的测试用例一通过，对于那个里程碑而言，你就已经符合了你的质量目标。

使用测试输入建立跟踪（Setting up Traceability Using Test Inputs）

正如*Identifying What to Test by Using Test Inputs*24页的内容所描述的，测试输入帮助你决定测试内容。当你创建你的测试用例时，你可以建立测试输入与他们的关联。这样做，你可以确定一个测试用例是否需要修改，因为他关联的测试输入发生了改变。

你也可以使用关联来确定是否有一个测试用例覆盖了这些测试输入。例如，假设你用需求作为测试输入。当每个测试输入与一个测试用例相关联时，你知道了你的所有需求都被覆盖到了。当每个测试用例通过时，你知道了所有的测试输入都生效了。

你可以用下面的方法建立一个测试输入与一个测试用例的关联：

●在你创建一个新的测试用例时，点击New Test Case 对话框中的**Test Inputs**标签。

●在你编辑一个现存的测试用力属性时，点击Test Case Properties对话框中的**Test Inputs**标签。

●在Test Plan窗口中，右键点击一个测试用例并点击**Associate Test Input**。

●在Test Inputs窗口中，右键点击一个测试输入并点击**Associate Test Case**。

测试设计（Designing Tests） 3

本章描述如何设计测试。它包含了以下标题内容：

●关于测试的设计

●指明测试步骤和检验点。

●指明测试用例的前置条件，后置条件和可接受标准。

●一个测试设计的例子。

注意事项：过程的细节化，参阅TestManager帮助。

关于测试的设计（About Designing Tests）

一旦你已经定义了需要去测试的特征，你就需要决定如何去进行测试。测试的设计活动首先回答问题“我如何执行这个测试用例？”

作为测试用例设计的部分，你可以确定：

●以执行测试需要的基本步骤集合。

●如何使你测试的项目或特征有效适当地工作。

●测试用例的前置条件－－如何设置应有程序和系统以便测试用例可以执行。

●测试用例的后置条件－－如何在测试用例执行后做清除。

●可接受标准－－如何决定测试用例是否通过。

在实际的系统实施之前或期间，你应当能够设计基于测试输入－－诸如特征描述和软件说明（例如，需求），来设计你的测试。这是一个测试与系统的实施并行开发的关键方面。

一些人应当到那时能够获得测试的设计和一个系统（伴随文档的）实施，以及知道如何实施这些测试。

例如，如果你正在使用一个自动测试工具，像Rational Robot，那么你应当能够启动你的工具和按照在测试用例的设计中记述的步骤去创建一个自动测试脚本。这个测试脚本成为被设计的测试用例的一个实施，因此也是测试用例本身。

正如另外一个例子，在你实施测试用例之前，你可以看到所有的测试设计（一对一的测试用例）。你可能发现在测试用例中的模式，它显示一个更为有效的方法以实施这些测试用例。例如，你可能看到每一个测试设计开始的第一步都这样说：“从Start菜单中，启动应用程序。”你可能决定不必在每一个测试脚本中记录这一步骤，因为如果这个应有程序的名称发生变更，那么所有的脚本都需要被变更。相反的，你可能创建一个子程序来启动这个应有，并有测试脚本来调用那个子程序。这将借对测试设计的考虑而变得显而易见。

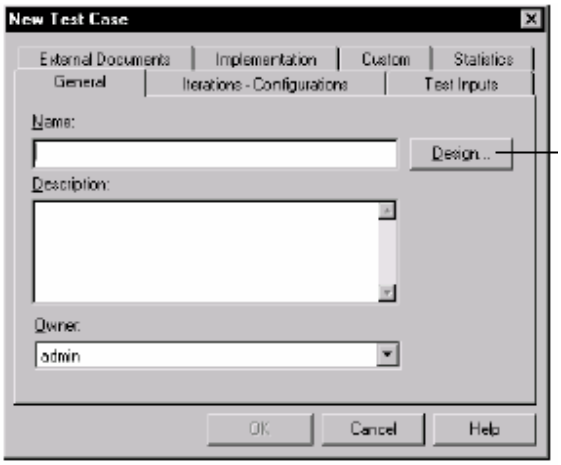
你可以轻易地输入一个测试设计到手工测试脚本中，到那时手工测试脚本成为测试用例的实施。有关手工测试脚本的信息，参阅61页*Creating Manual Test Scripts*的内容。

指明测试步骤和检验点（Specifying the Testing Steps and Verification Points）

在第一次创建测试用例或晚些时候，你可以设计一个测试用例。

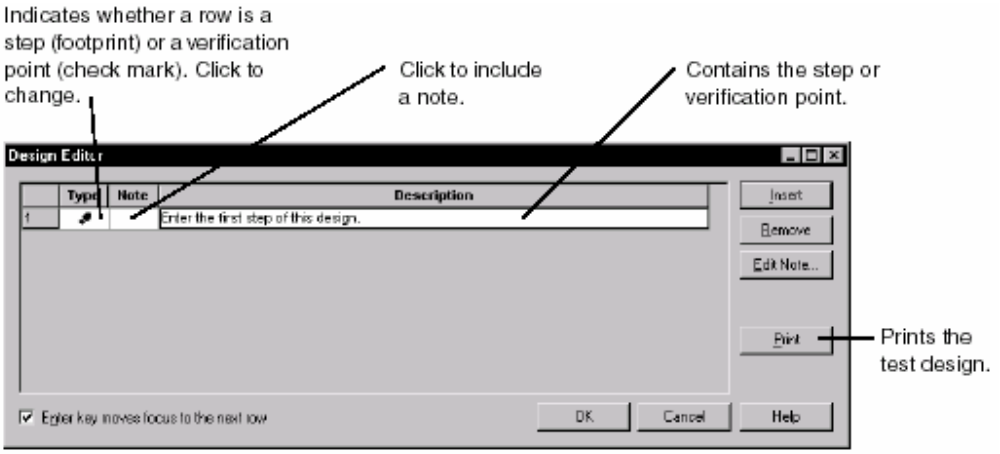
设计一个新的测试用例：

●在Test Plan窗口中，右键点击一个测试用例文件夹。点击**Insert Test Case**。



测试一个现存的测试用例：

●在Test Plan窗口中，右键点击测试用例。点击**Design**。



使用Design Editor来包含测试步骤和检验点，它们应当被包含在测试脚本中。

步骤－－在应有或系统中被获得的一个活动。在你第一次开始设计时，这可以是一般的，久而久之会变得更明确具体。

检验点－－在一个测试脚本中的一个检验点可以进一步确定一个或更多目标的状态。

在你点击Design Editor中的**OK**时，那个设计成为测试用例的一个属性。

测试设计将形成开发的迭代过程。。当你熟悉了系统将如何实施的更多细节时，你可以添加更多的步骤和检验点到设计中去。

注意事项：由一个测试用例设计创建一个手工测试脚本：点击**Implementation**标签，然后点击**Import from Test Case Design**按钮。更多信息，参阅61页*Creating Manual Test Scripts*的内容。

指明测试用例的条件和可接受标准（Specifying Conditions and Acceptance Criteria of Test Cases）

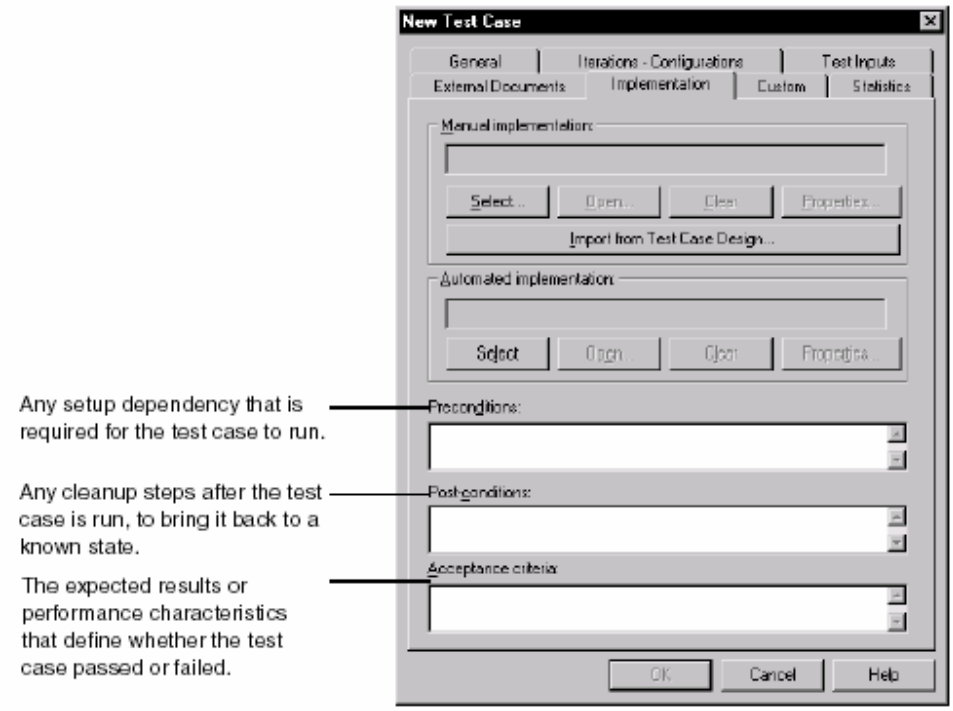
前置条件（*Preconditions*）和后置条件（*post-conditions*）为测试执行者提供信息。他们描述在一个操作开始或结束时必须是准确的系统约束，因此要确保测试用例可以恰当地执行并脱离系统在一个适合的状态中。一个前置条件或后置条件的失败并不意味着测试的行为或功能不能工作。它意味着与约束不符。

可接受标准表明为一个特殊测试用例需要被明确的东西，用以该测试用例的通过。

指明条件和可接受标准：

1 在Test Plan窗口中，右键点击一个测试用例。点击**Properties**。

2 点击**Implementation**标签。



例如，如果测试用例需要验证登陆到一个系统的响应时间是否是可接受的，那么你可以在一个测试用例中包含以下信息：

前置条件——你必须拥有可登陆系统的合适的用户ID，系统必须是在一个退出状态中。

后置条件——在你登陆并成功地验证该测试用例后，你需要退出登陆（或引导系统为随后的测试回到一个已知状态中）。

可接受标准——针对此测试用例的通过，响应时间的范围应当在0.5和2.0秒之间。

在另一个例子中，你可以有5个检验点在你的测试中。不过，在某个验证点，为测试用例的通过而仅仅需要他们中的3个通过即可。在这个用例中，可接受标准可能基于迭代进行变更。

一个测试设计的例子（Example of a Test Design）

这一节提供针对测试用例的一个设计实例。由于测试设计是基于测试输入的，所以他可以在任何代码编写之前被开发。

在这个例子中，你正在测试一个自动取款机（ATM）。你的需求包括由一个确定的帐户类型来提取现金的测试用例。另外一个需求是确定去执行与ATM的任意交易，用户必须被确定和使之有效。由这些需求，你定义一个测试用例来确保你可以提取一笔现金，当然，需要一个检查帐户，帐户在开始此次交易时包含的金额多余提取的金额数。

首次测试设计的迭代可能是：

前置条件

●确保我们有一个有效的帐户设立和已知的用户ID，以及有效的信息（密码或PIN）。

●确保这里有一个该用户的检查帐户，以及我们所知道的现金余额。

●现金余额必须大于0。

设计

●步骤——确认用户使用ATM并使之有效。

●检验点——确认我们已经登陆。

●步骤——选择“Checking”作为帐户类型，选择“Withdraw”作为交易类型。

●步骤——指明提取金额小于现金余额。

●检验点——确保被分配的金额与指明的金额匹配。

●检验点——执行帐户余额交易以确保新的余额等于旧的余额减去提取的数量。

后置条件

●确保用户退出对ATM的使用。

可接受标准

所有的检验点必须成功。

在更多的系统细节成为可利用时——在你移动通过对测试资产的迭代，像可视化模型，软件的详细说明，原型等等——你可以添加更多的细节到测试设计中去。例如，你可能在晚些时候了

解到用户将自己通过提款卡和PIN来确认。你可以更新设计来添加步骤－插卡，输入PIN，并在结束时取回提款卡。

测试的实施（Implementing Tests） 4

本章描述如何实施测试。它包括以下标题内容：

- 关于测试的实施
- 测试用例的实施
- 从测试脚本中调用Test Script Services
- 创建手工测试脚本
- 建立一个实施与一个测试用例的关联
- 实施的测试作为suites

关于测试的实施（About Implementing Tests）

在你为每一个测试用例创建了测试设计后，你已经可以准备去实施测试用例了。实施测试用例是通过创建一个测试脚本并建立测试脚本与测试用例的关联来实现。

在每个组织中，实施都是不相同的。你可以使用你更喜欢的工具或手工测试去创建任何一种针对你的测试的合适的测试脚本。

例如，一个测试组织可能决定通过使用Rational Robot记录测试脚本来实施所有的测试用例。

另一个组织可能决定去写模块化的软件，使用一个可视化的测试脚本，批处理文件和Perl脚本的混合体，然后有计划地将他们组织在一起形成一个更高级别的脚本。

在你实施一个测试脚本后，你可以在TestManager中建立它与一个测试用例的关联。有关信息，参阅65页上*Associating an Implementation with a Test Case*的内容。

然后你可以在TestManager中执行测试用例或测试脚本。你也可以插入测试脚本到一个suite中，并执行这个suite。有关执行实施的信息，参阅87页*Executing Tests*的内容。

测试用例的实施（Implementing Test Cases）

你可以通过创建一个测试脚本或一个suite来实施一个测试用例。

当你创建了一个测试脚本时，你可以使用一个Rational测试实施工具去创建一个内置类型的测试脚本，或者你可以创建一个常规类型的测试脚本。

内置测试类型（Built-in Test Scripts Types）

TestManager紧密地集成了Rational的测试实施工具。由TestManager启动，你可以容易地实施：

- 在Rational Robot中记录的自动测试脚本
- 在Rational ManualTest中创建的手工测试脚本

自动测试脚本（Automated Test Scripts Recorded in Rational Robot）

TestManager包含下面的内置的测试脚本类型以提供给测试的实施（如果安装了Rational Robot，这些测试脚本在其中执行）：

●**GUI——**用SQABasic编写的一种测试脚本，一种Rational专有的类Basic脚本语言。GUI测试脚本主要被用来做功能测试。

●**VU**——用VU编写的一种测试脚本，一种Rational专有的类C语言的脚本语言。VU测试脚本主要被用来做性能测试。

（在你开始记录一个VU测试脚本时，你实际上记录了一个session。你可以由这个被记录的session中产生VU或VB测试脚本，依赖你在Robot中选择的一个记录选项。）

在Robot中记录一个测试脚本，开始于TestManager：

●点击**File > Record Test Script > GUI**或**VU**。

这样就启动了Robot并打开了Record对话框。

有关记录测试脚本的信息，参阅*Using Rational Robot*手册和Robot帮助。

手工测试脚本（Manual Test Scripts Created in Rational ManualTest）

TestManager包含内置的手工测试脚本类型以提供给测试的实施，这些手工测试脚本在Rational ManualTest中实施。一个手工测试脚本包含一个测试指令集合以人为的执行测试。

有关信息，参阅61页*Creating Manual Test Scripts*的内容。

常规的测试脚本类型（Custom Test Script Types）

TestManager扩展测试脚本的功能性可以让你去实施其他工具的测试脚本，当然这些测试工具要符合你的测试环境。

Command Line and Custom Adapters

这里有两种方法来扩展TestManager以支持一种新的测试脚本类型：

●使用command line适配器。

●创建常规适配器或使用Rational和她的合作商提供的适配器。

Command Line Adapters

TestManager提供两种command line适配器：

●Command Line Test Script Console Adapter—测试脚本的测试工具或编辑器针对测试脚本创建和编辑提供一个command-line接口时，测试脚本可以以标准的File Open对话框形式被打开时（例如，Perl脚本），可以使用该适配器。

●Command Line Test Script Execution Adapter—测试脚本的测试工具针对测试脚本的执行提供一个command-line接口时，可以使用该适配器。

使用这些适配器的优势是他们不需要常规的编程。

你正好需要在TestManager中去定义新的测试脚本类型，有关信息，参阅58页*Defining a New Test Script Type*的内容。

常规适配器（Custom Adapters）

代替command line适配器的使用，你可以创建自己的常规适配器或使用Rational Software和她的合作商提供的适配器。一个适配器就是一个动态连接库（DLL）。

这里有两种类型的常规适配器：

●Custom Test Script Console Adapter—在测试工具针对测试脚本的创建和编辑不提供一个command-line接口时，对于测试脚本不以标准的File Open对话框形式打开时，需要使用该适配器。

●Custom Test Script Execution Adapter—如果测试工具针对测试脚本的执行不提供一个command-line接口，就需要使用该适配器。

有关创建常规适配器的信息，参阅*Rational TestManager Extensibility Reference*手册。

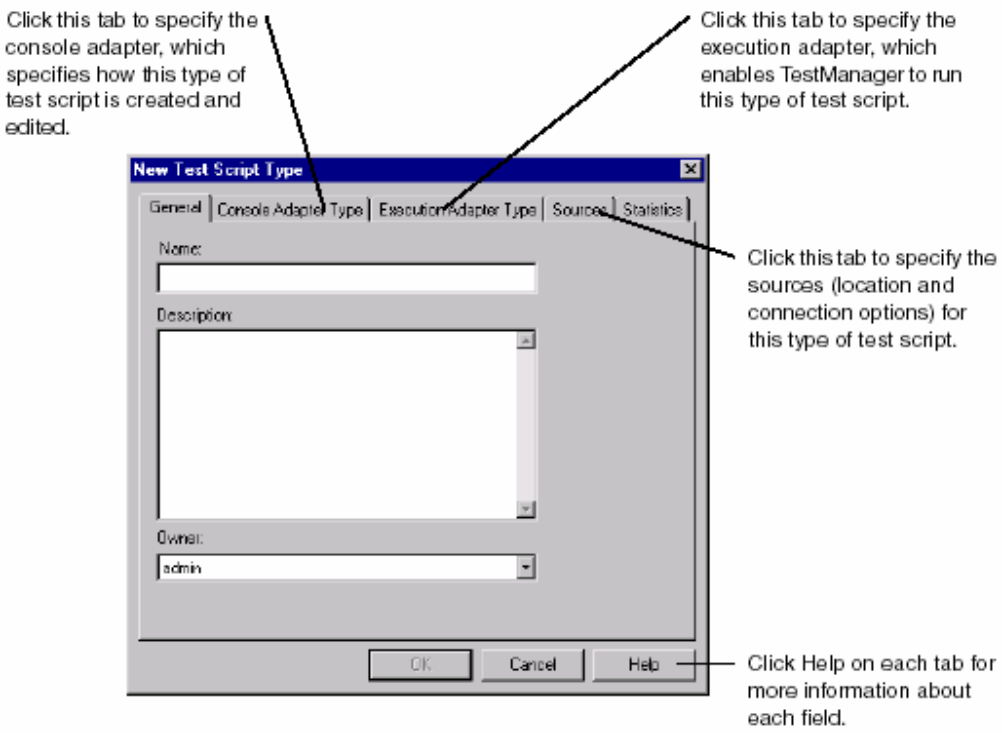
在这些适配器被创建之后，你或者其他属于测试组的人员必须在TestManager中定义新的测试脚本类型并注册该DLL。（有关信息，参阅下一节，*Defining a New Test Script Type*。）你应当可以到那时打开并执行那种类型的测试脚本。依赖于Test Script Console Adapter的如何实施，你也可能创建和编辑那种测试脚本类型。

定义一种新的测试脚本类型（Defining a New Test Script Type）

在TestManager中定义一个新的测试脚本类型：

●点击**Tools > Manage > Test Script Types**。点击**New**。

注意事项：如果**New**按钮不可用，那么你还没有管理员的权限。有关信息，参阅*Using the Rational Administrator*手册或帮助。



创建 Suites（Suites Created in TestManager）

TestManager允许你由测试脚本、测试用例、和其他物件来创建测试的*suites*。

Suites为通过一个point-and-click接口创建的功能和性能测试提供很好的灵活性和效率。

Suite的基础部分参阅67页*Implementing Tests as Suites*的内容。有关功能测试suites的细节内容，参阅165页*Creating Functional Testing Suites*。有关性能测试suites的细节内容，参阅225页*Designing Performance Testing Suites*。

由测试脚本调用 Test Script Services （Calling Test Script Services from Test Scripts）

Rational Test Script Services (TSS) 是测试的服务，你可以由你的测试脚本使用Test Script Services API中的命令来调用。例如，你可以从测试脚本调用logging，同步（synchronization），定时（timing），以及数据池（datapool）等服务。你可以调用检验点服务来验证（validate）一个组件或系统的状态和行为。下面的表列举了TSS提供的服务种类：

Category	Description
Datapool	Provides variable data to test scripts during playback, allowing virtual testers to send different data to the server with each transaction.
Logging	Logs messages for reporting and analysis.
Measurement	Provides the means of fine-tuning and controlling your tests through operations such as timing actions, setting think time delays, and setting environment variables.
Utility	Performs common test script operations such as retrieving error information, controlling the generation of random numbers, and printing messages.
Monitor	Monitors playback progress of a test script.
Synchronization	Synchronizes multiple virtual testers running on a single computer or across multiple computers.
Session	Manages test script session execution and playback.
Advanced	Advanced features, such as setting values for internal variables.
Verification Point	Validates the state or behavior of a component or system.

种类	描述
Data pool （数据池）	在测试脚本录制回放期间为其提供变量数据，允许每个虚拟测试者在对于各事务处理时，向服务器发送不同的数据。
Logging （记录日志）	为报告和分析提供日志（logs）信息。
Measurement （衡量）	提供微调（fine-tuning）的方法，并通过诸如定时活动、设置考虑时间延迟、和设置环境变量等操作来控制你的测试。
Utility （用途）	执行一般的测试脚本操作，如回收错误信息，控制随机数字生成，

	和打印信息。
Monitor （监控）	监控一个测试脚本的录制回放过程。
Synchronization （同步）	同步化并联的虚拟测试者在一台测试机上或者通过并联的测试机执行。
Session	管理（处理）测试脚本session的执行和录制回放。
Advanced （高级的）	高级的特征，如为内部变量设置值。
Verification Point （检验点）	验证一个组件或系统的状态或行为。

TSS 与测试脚本类型（Test Script Services and Test Script Types）

VB，Java，和command line的测试脚本类型－－如同常规测试脚本类型，可添加到TestManager中－－可以带给TSS以优势。

你可以在测试脚本编辑期间手工添加TSS命令到测试脚本中去。

TSS命令可以在下面的操作期间自动添加到测试脚本中：

- 用Rational Robot记录的VB测试脚本。
- 用EJB Session Recorder（包括了Rational QualityArchitect）的记录。Session Recorder允许你可视化地与EJBs链接并交互。在你执行针对组件的业务时，交互数据被记录和存储在一个扩展的XML文件中。然后，你可以使用XML Script Generator去为测试EJB生成Java测试脚本。
- 使用Rational QualityArchitect产生测试脚本。QualityArchitect分别为测试EJB组件和COM/DCOM组件产生Java测试脚本和VB测试脚本。

使用下表作为一个指南，包括了不同类型测试脚本的TSS。细节信息，参阅*documentation listed for each test type*。

Type of Test Script	Method of Adding Test Script Services Commands	Documentation
VB	Recording a session and generating the test script with Rational Robot or manually editing	*Rational Test Script Services for Visual Basic*
Java	Manual editing	*Rational Test Script Services for Java*
Command Line	Manual editing	*Command Line Interface to Rational Test Script Services*

测试脚本类型	添加的方法	资料
VB	记录一个session并由Rational Robot或手工编辑生成测试脚本	Rational Test Script Services for Visual Basic
Java	手工编辑	Rational Test Script Services for Java
Command Line	手工编辑	Command Line Interface to Rational Test Script Services

注意事项：Robot的VU和GUI测试脚本类型自动提供TSS。例如，在你用Rational Robot记录了一个脚本时，这个脚本就包含了一个数据池（datapool）（如果适合的话）。TestManager也提供内置的监控和session函数。

TSS 与 TestManager（Test Script Services and TestManager）

TSS为与TestManager一起使用而设计。作为一个结果，TSS的特征是包含了任何一种测试脚本类型－－包括常规的测试脚本类型－－完全地集成了TestManager的报告记录、监控、和分析框架。

例如：

●TestManager 遵守任何的同步和延迟你的功能测试脚本，当它录制回放（执行）的时候，在测试脚本的一组suite里面回放（运行）测试脚本。

●在测试脚本录制回放期间，一个测试人员可以通过测试脚本监控指令来监控有关测试脚本的

状态信息。

●在测试脚本录制回放期间，TestManager通过数据池的使用提供给测试脚本以真实和可变的数据。

●定时的活动结果在TestManager报告中展示。

●TestManager测试用例可以建立与测试脚本之间的关联，这些测试脚本包含了检验点命令以验证一个组件或系统的状态和行为。

●TestManager可以在一组单一的suite中执行不同类型的测试脚本。例如，VU、VB和常规测试脚本类型可以全部在同一组suite中执行。

创建手工测试脚本（Creating Manual Test Scripts）

一个*manual test script*是一个测试的说明（指令）集合以被一个真实测试人员执行。测试脚本可以由你输入到一个编辑器的步骤和检验点组成。

一个*step*是一个说明，在一个手工测试脚本执行时，它被测试人员所执行。这可以像一个单一的句子那样简单（比如“Reboot the computer”）或是像一个完全的文档那样复杂。

在一个手工测试脚本中，一个*verification point*（检验点）是关于应用程序状态的一个问题（例如，“应用程序开始了吗？”）。一个检验点可以由任意数量的文本组成，但这些可能是一个或两个句子，通常以一个问题标志作为结束。

在你创建一个手工测试脚本之后，你可以建立它与一个测试用例的关联。当你执行这些测试用例或手工测试脚本时，测试脚本在ManualTest中打开。

当你执行一个手工测试脚本时，你执行每一步并确定是否每一个检验点都通过或失败。你可以到那时打开TestManager中的Test Log窗口并查看结果。如果所有的检验点都通过了，那么测试脚本也通过。如果有任何一个检验点失败，那么测试脚本即失败。

与其他的测试脚本类型一样，你可以在TestManager报告中包含你的手工测试脚本。

注意事项：有关手工测试脚本的细节过程，参阅Rational ManualTest的帮助。

你可以用两种方法创建一个手工测试脚本：

●通过一个测试用例设计的输入。

●用Rational ManualTest。

由一个测试用例设计创建（Creating a Manual Test Script from a Test Case Design）

如同47页上*About Designing Tests*的内容描述，一个测试用例的设计，首先要回答问题“我如何执行这个测试用例？”一设计可以包含步骤和检验点。

你可以由一个测试用例的设计容易地创建一个手工测试脚本。设计中的步骤和检验点成为手工测试脚本中的步骤和检验点。

由一个测试用例设计创建一个手工测试脚本：

●在Test Case对话框中，点击**Implementation标签**。点击**Import from Test Case Design**按钮。手工测试脚本成为测试用例的实施。

你可以通过点击**Implementation**标签中**Open**按钮来查看和编辑手工测试脚本。这样就打开了Rational ManualTest。有关信息，参阅下一节，*Creating a Manual Test Script in Rational ManualTest*。

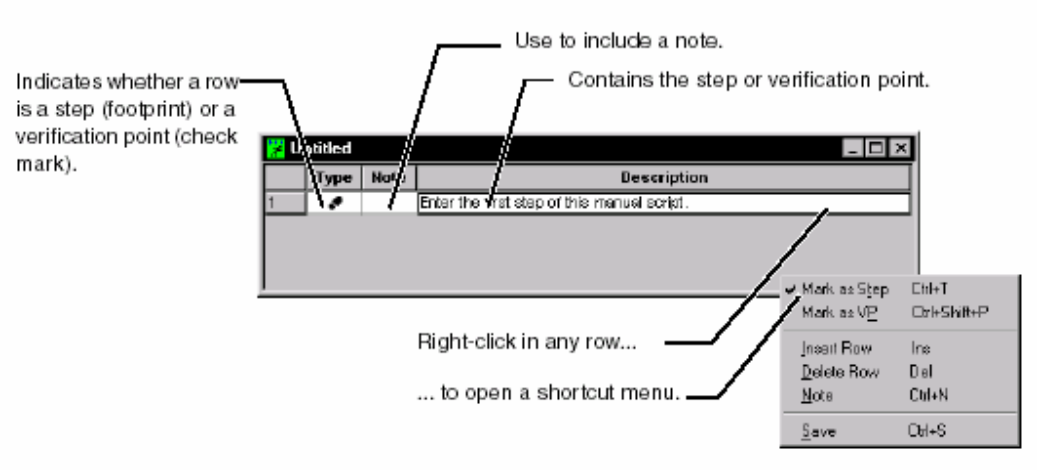
用 Rational ManualTest 创建（Creating a Manual Test Script in Rational ManualTest）

Rational ManualTest与Rational TestManager集成在一起。使用Rational ManualTest去创建和执行手工测试脚本。

启动（Starting Rational ManualTest）

启动Rational ManualTest并创建一个新的手工测试脚本：

●在TestManager中，点击**File > New Test Script > Manual**。

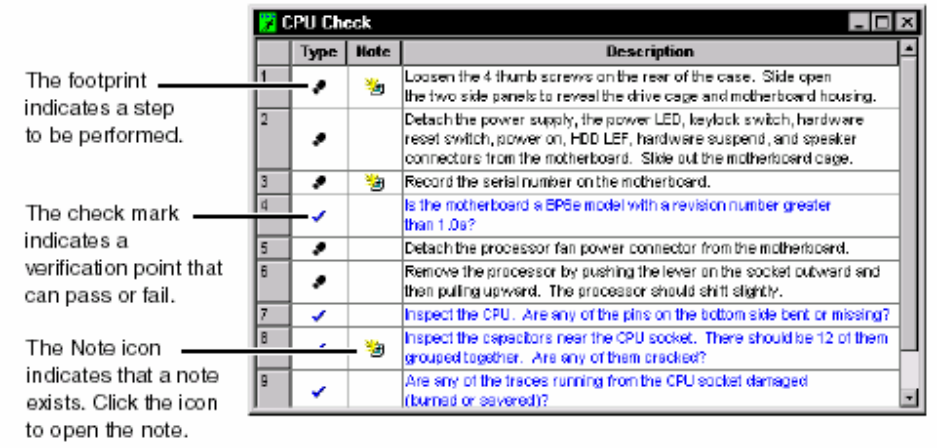


一个例子（Example of a Manual Test Script）

下面的手工测试脚本包含了五个步骤和四个检验点。

●步骤是在你执行该测试脚本时针对你做出的活动。

●检验点时你要回答的问题。



设置默认的编辑器（Setting the Default Editor for Manual Test Scripts）

你可以在你创建一个手工测试脚本时使用表格编辑器或文本编辑器。表格编辑器是一个结构化的编辑器，制作它很简单，以输入你的步骤和检验点。文本编辑器是一个自由形态编辑器，制作简单以

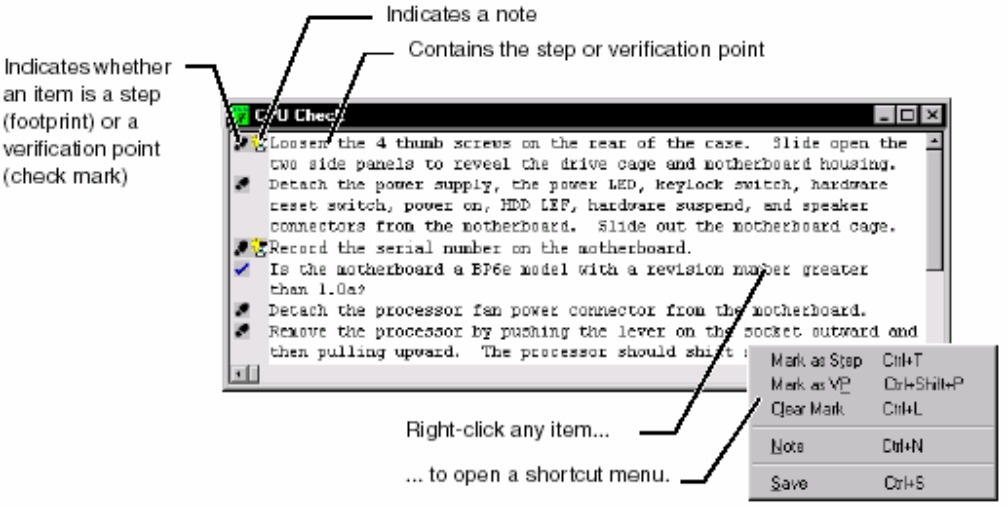
操作文本。

在ManualTest中设置默认的编辑器：

●点击**Tools > Options**。

此次设置在你下次创建或打开一个手工测试脚本时生效。

在下一个编辑器中，你使用一个快捷菜单去标明条目作为步骤和检验点，以及去创建和查看注意条款。



一个条目（步骤或检验点）的开始是被足迹（footprint）或检查标记图标来确定。所有的排列作为先前条目的一部分，排列不开始于这些图标的任一种。

创建测试脚本问题（Creating Test Script Queries）

Rational ManualTest提供*queries*以帮助你在你的Rational项目中选择测试脚本。Queries允许你说明在选择对话框中出现的区域，测试脚本如何被分类，以及哪些测试脚本会出现。

在你的queries中使用*filters*（过滤器）以说明从一个项目中回收的信息。

过滤器通过收缩你要搜索的信息帮助你使queries更加明确化。你可以创建简单的过滤器或简单过滤器的组合到更复杂的一个中去。在你创建或编辑一个query时可以使用过滤器。

创建一个新的query：

●点击**Tools > Manage Script Queries**。

定制测试资产（Customizing Test Assets）

当你创建一个手工测试脚本时，你可以添加常规属性去裁剪术语建立测试脚本的关联以达到你组织内使用的标准和实践。

你可以做一个测试脚本的如下属性：

●总计三个常规属性和值。（这些出现在New Test Script/Test Script Properties对话框的**Custom**标签中。）

●添加新的操作环境或修改、删除现存的一个。

●添加一个新的目的或修改、删除现存的一个。你分配一个目的去说明你为什么会使用一个测试脚本。

在Rational ManualTest中参阅一个手工测试脚本的标准属性：

●点击**File > Properties**。

在Rational ManualTest中定制一个测试脚本。

●点击**Tools > Customize Test Script**。

（You can define both the property itself (the label) and the values that can be used with that property.）

你可以定义属性和值使用那些属性。

有关定制测试脚本的信息，参阅ManualTest的帮助。

你也可以在TestManager中查看并定制任意测试脚本的属性。

建立一个实施与一个用例的关联（Associating an Implementation with a Test Case）

在你已经创建一个实施之后，你可以建立它与一个测试用例的关联。你可以到那时执行该测试用例，执行它的实施。通过建立测试脚本与测试用例的关联，你可以执行报告以提供测试的覆

盖信息。

TestManager来自于内置型的提供给下面实施类型的关联：

●GUI测试脚本

●VU测试脚本

●VB测试脚本

●Java测试脚本

●Command-line可执行编程

●Suites

●手工测试脚本

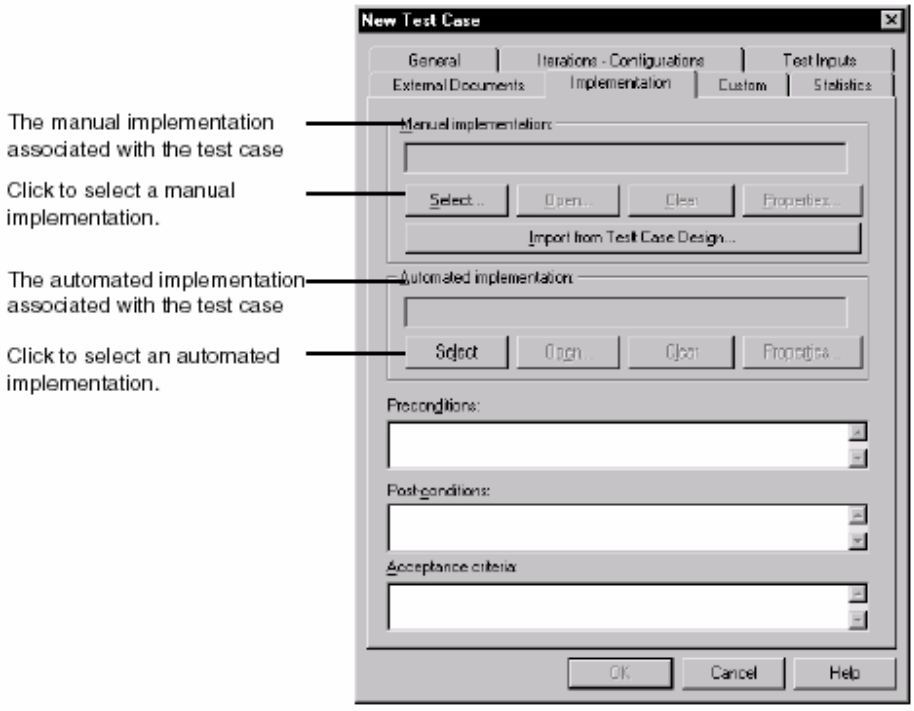
TestManager也提供与其他你已经注册的测试脚本类型的关联。

有关信息，参阅57页的*Custom Test Script Types*。

建立一个实施与一个测试用例的关联：

1 在Test Plan窗口中，右键点击一个测试用例。点击**Properties**。

2 点击**Implementation**标签。

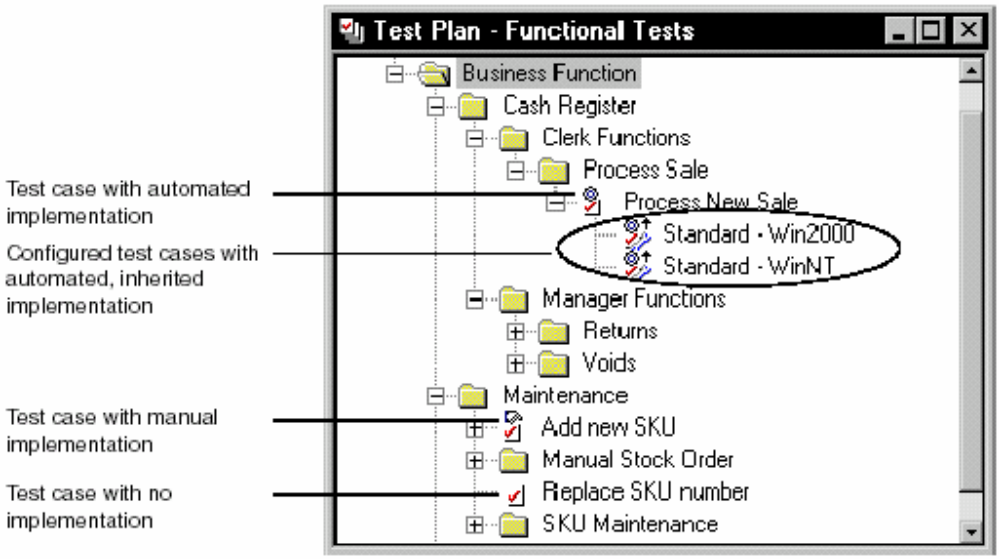


你可以用最多两个实施与一个测试用例相关联：一个手工的和一个自动的。如果他们都关联与一个测试用例，那么在你执行测试用例时，TestManager执行那个自动的实施。有关执行测试用例的信息，参阅90页*Running Test Cases*的内容。

注意事项：如果你由Rational ManualTest Web Execution组件执行一个测试用例，那么手工测试脚本将执行。有关信息，参阅351页*About ManualTest Web Execution*的内容。

当你创建一个配置的测试用例时，它继承它的父测试用例的实施。你可以在Configured Test Case对话框中通过点击**Select**按钮来改变一个配置的测试用例的实施。

Test Plan窗口中的图标指明了是否一个测试用例或配置的测试用例有一个实施，并是否是自动的或手工的。对于一个配置的测试用例，该图标也可以指明该实施是否继承于他的父测试用例。下图所示一些图标：（对于一个完整的图标列表，参阅Rational TestManager的帮助。）



注意事项：即使一个图标显示的这个实施是自动的，测试用例也可以有一个手工实施。

实施测试作为 Suites （Implementing Tests as Suites）

Suites是另一个在TestManager中实施测试的方法。一组*suite*展示了你要测试的工作的一个分层描述，或你要添加到系统中的工作量。它展示了诸如用户或计算机组之类的条目，每个组的资源分配，执行哪个组的测试脚本，以及每个测试脚本执行的次数。

在实施的一个测试作为一组suite时，你可以：

●定义用户或计算机组，并将资源应有于其中以确定它们执行的地方。

组是应用程序中执行相同任务的虚拟测试者的集合。

●添加测试脚本。

测试脚本是说明的集合。测试脚本可以被用来导航一个应用的用户接口以确定所有的特征工作，或去测试执行在接口后面的应用的活动。

●添加suites 到suites。

你可以使用suites作为suites内部的构建块。

●添加测试用例。

一个测试用例是在一个目标测试系统中的一个可测试和可验证的行为。

你即可以在功能测试中使用suites，也可以在性能测试中使用它。虽然在两种测试类型中，suite的概念是相同的，但是你将要插入不同的suite条目，并选择不同的选项，和依赖于你是否正在执行一个功能测试或性能测试。下面的两节给出了有关在功能和性能测试中使用suites的一个概述。

在功能测试中使用 Suites （Using Suites in Functional Tests）

在你针对功能测试而使用一组suite时，你开始于设置一台计算机组。这个计算机组可以在指定的测试机上执行测试用例或测试脚本。一个测试用例一次只能执行在一台测试机上。你可以设置测试脚本以便他们执行在：

●你预先指定的特定的测试机。

●在执行期间指定的特定的测试机。

●任何一台空闲的以执行一个测试脚本的测试机。

你可能想在一台特定的测试机上执行一个功能测试。例如，你的测试可能被指定于一台特定的测试机上。或者，你的测试可能需要特殊的软件，而这些软件要被安装在一台特定的测试机上。相反地，你可能想在几台测试机上分布（执行）一个功能测试的测试脚本。例如，你的测试可能不必被指定于一台特定的测试机上。你可能想在一个计算机组上执行你的测试，以便它们能够完全的尽可能地快速执行。

实施的一个测试作为一组suite能够使你到达每一个目标。有关在功能测试中使用suites的更多信息，参阅165页*Creating Functional Testing Suites*的内容。

在性能测试中使用 Suites（Using Suites in Performance Tests）

在性能测试中，一个*suite*不仅仅能够使你执行测试脚本，也可以模拟用户的活动以添加工作量到一台服务器中去。一组suite可以如同一个虚拟者执行一个测试脚本那样简单，或如同上百个虚拟者在不同的组内，每组在不同的时间内执行不同的测试脚本。

有关在性能测试中使用suites的更多信息，参阅225页*Designing Performance Testing Suites*的内容。

对于性能和功能测试的一般活动（Activities Common to Performance and Functional Testing）

无论你是在执行性能测试还是功能测试，你将要执行某种一般的活动。例如，你定义测试机和测试机列表，创建suites，打开suites，编辑测试脚本，并在suites中编辑条目和属性。这些章节将论述你在性能和功能测试suites中要做的工作。

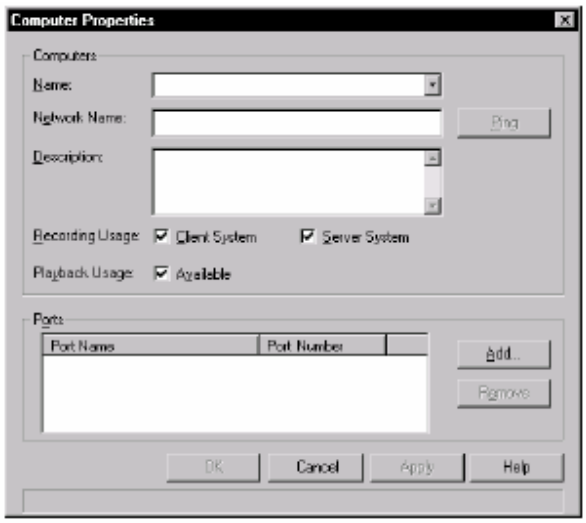
定义代理测试机和测试机列表（Defining Agent Computers and Computer Lists）

测试执行，以默认方式，是在本地测试机上。然而，你可以添加其他的测试机，名为*Agents*，对于你的测试环境。这些代理测试机在性能和功能测试中是有用的。在性能测试中，你使用代理（测试机）去添加工作量到服务器中。在功能测试中，你使用代理（测试机）来并行地执行你的测试，从而以节约时间。

添加一台代理测试机（Adding an Agent Computer）

添加一台代理测试机：

● 点击**Tools > Manage > Computers**，然后点击**New**。



在你添加一个代理测试机到TestManager中时，你可以包含下面的属性：

●**名称**－－测试机的名称。使用名称来帮助你容易地识别该测试机，如果网络名称不是非常的具有描述性。

●**网络名称**－－该测试机的名称作为其在计算机网络中的标识。以检查并确认该测试机的网络名称是可用的，点击**Ping**。

●**描述**－－测试机的一个描述，在测试过程中可能注意到的规则。

●**记录的用途**－－TestManager在记录期间是否视测试机为一台客户机或服务器。

●**录制回放的用途**－－系统对于测试脚本的录制回放是否可用。

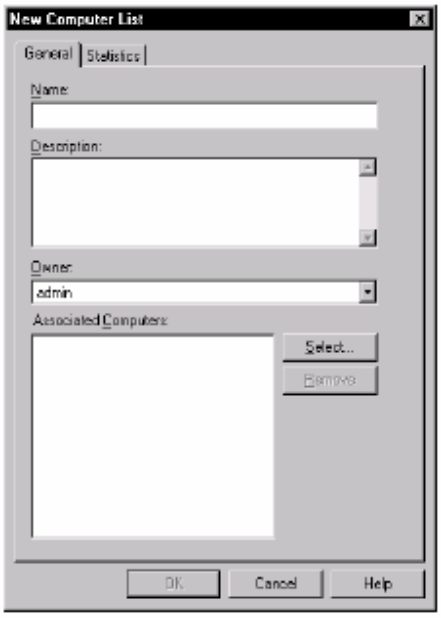
●**端口信息**－－TCP/IP端口信息关联与系统。有关端口设置的信息，参阅331页*Configuring Local and Agent Computers*的内容

合并测试机到列表中（Combining Computers into Lists）

你已经在TestManager中定义了测试机，之后，你可以合并他们到列表中去。如果你在多台测试机上执行测试，那么列表是有用的，或者如果机台测试机有一个相似的配置或执行一个相似组织的任务。通过使用一个列表，你可以参考列表以代替单个的测试机。

创建一个测试机列表：

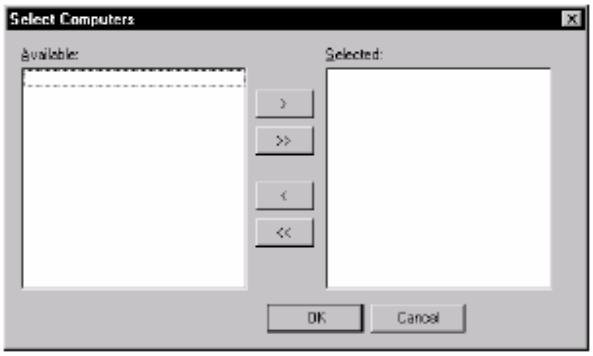
●点击**Tools > Manage > Computer Lists**，然后点击**New**。



在你定义了测试机列表以及它的一个名称和描述之后，添加测试机到列表中。一个测试机包含在一个列表之前，你必须单个地添加该测试机到TestManager中。

添加测试机到一个测试机列表：

● 由Computer List Properties的对话框，点击**Select**。



在你已经定义了测试机和测试机列表之后，它们对于suites的执行就是可利用的资源了。

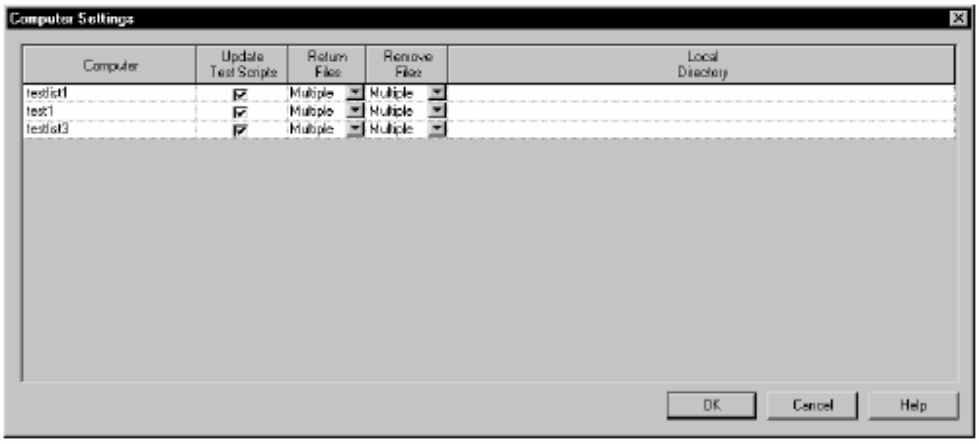
注意事项：确定复制suite需要的任何常规创建的外部C库，Java类文件，或COM组件到代理测试机中。

变更一台代理测试机的设置（Changing the Settings of an Agent Computer）

你可能想变更与一台代理测试机关联的默认设置。代理测试机的配置可能已经变更和在TestManager中的设置需要反映这。

变更测试机的设置：

●打开一组suite，然后点击**Suite > Edit Computers**。



创建一组 Suite （Creating a Suite）

你可以用几种方法来创建一组suite。

你创建一组suite：

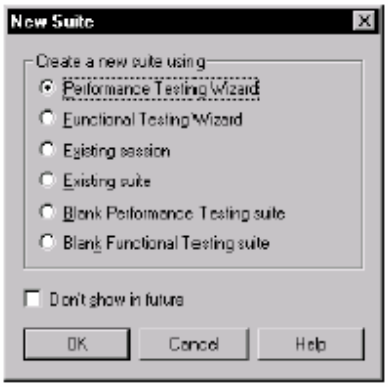
●利用向导。

●基于一个Robot session或另一个suite。

●从开始，使用一个空白的模板。

使用这些方法的任意一个创建一组suite：

●点击**File > New Suite**。



有一个向导创建一组 Suite （Creating a Suite from a Wizard）

如果你是新的测试，使用suite向导可能是最简单的方法去创建一个工作测试。你必须针对在suite中的使用具有可用的测试脚本或测试用例。

在你利用性能测试向导创建一组suite时，TestManager帮助你选择将执行测试的测试机，并帮助你关联测试脚本，这些测试脚本成为此次测试的根据。

在你利用功能测试向导创建一组suite时，TestManager帮助你选择测试用例和测试脚本，它们成为此次测试的根据。

打开一组 Suite （Opening a Suite）

你可以从一个菜单或是Test Asset Workspace中打开一组suite。

从菜单中打开一组suite：

●点击**File > Open Suite**。

从Test Asset Workspace 中打开suite：

●从**Execution**标签中，双击树中的suite。

编辑一个测试脚本（Editing a Test Script）

在你正在用一组suite工作的时候，你可能想编辑一个测试脚本。

通过TestManager，你可以：

●编辑一个测试脚本的属性。

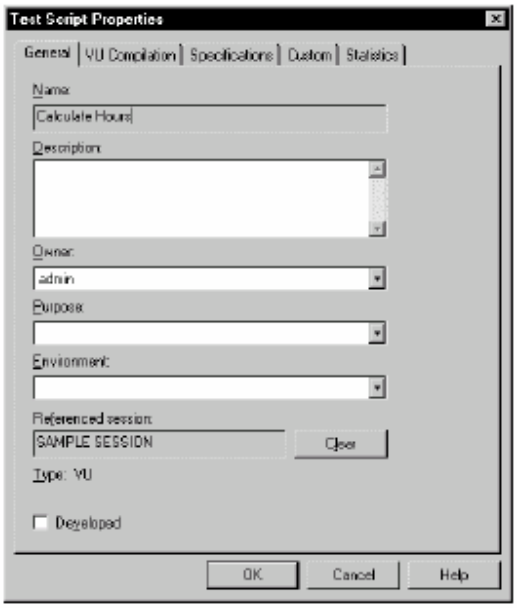
●编辑一个测试脚本的文本。

编辑一个测试脚本的属性（Editing the Properties of a Test Script）

一个测试脚本可以有除测试脚本名称和类型之外的属性相关联。测试脚本实例的属性包括测试脚本的一个描述和测试脚本的目的。

编辑一个测试脚本的属性：

●打开一组suite，选择该测试脚本编辑，然后点击**Edit > Properties**。



注意事项：在这个对话框中的标签有轻微地变化，依赖与你打开的测试脚本的类型。

编辑一个测试脚本的文本（Editing the Text of a Test Script）

编辑一个测试脚本的文本：

●选择测试脚本，然后点击**Edit > Open Test Script**。

脚本是已载入的，合适的编辑器是已启动的。

有关编辑测试脚本的信息，针对你所用的语言，参阅*VU Language Reference*或Rational Test Script Services手册。

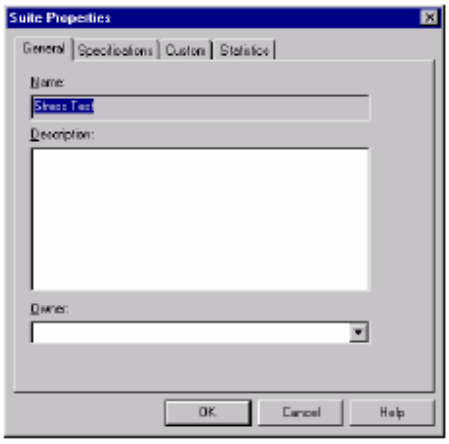
编辑一组 Suite 的属性（Editing the Properties of a Suite）

一组suite有属性关联与它，可以使它唯一化并帮助你区分类似的suites。

Suite实例的属性包括一个suite的描述和suite的拥有者。

编辑一组suite的属性：

●打开suite，然后点击**File > Properties**。



更换一组 Suite 的条目（Replacing Items in a Suite）

使用直接插入编辑来替换除delays和选择器之外的在suite中的任意条目。替换一个条目－－尤其是在suite结构中高级的条目。例如，你的suite可能包含复杂的一个用户组结构，测试脚本，和剧本（scenarios）。删除一个条目和重新创建该suite下面的结构，不如去替换这些条目（item）。

更换一个条目：

1 点击**Tools > Options > Create Suite**，并清除**Show numeric values**，检查box。

清除该选项允许你使用直接插入编辑来重命名条目名称。

2 打开一组suite，选择条目，并输入新条目名称。

编辑一组 Suite 条目的执行属性（Editing the Run Properties of Items a Suite）

在你的测试过程的进展中，你可能要编辑执行属性与suite条目相关联。你可以编辑包含在一个

suite的任何条目的执行属性。

编辑一个条目的执行属性：

1 打开一组suite，并选择条目以编辑。

2 点击**Edit > Run Properties**。

TestManager展示相同的对话框，在你创建了item之后显现的。

你可以编辑每一个对话框中的值。

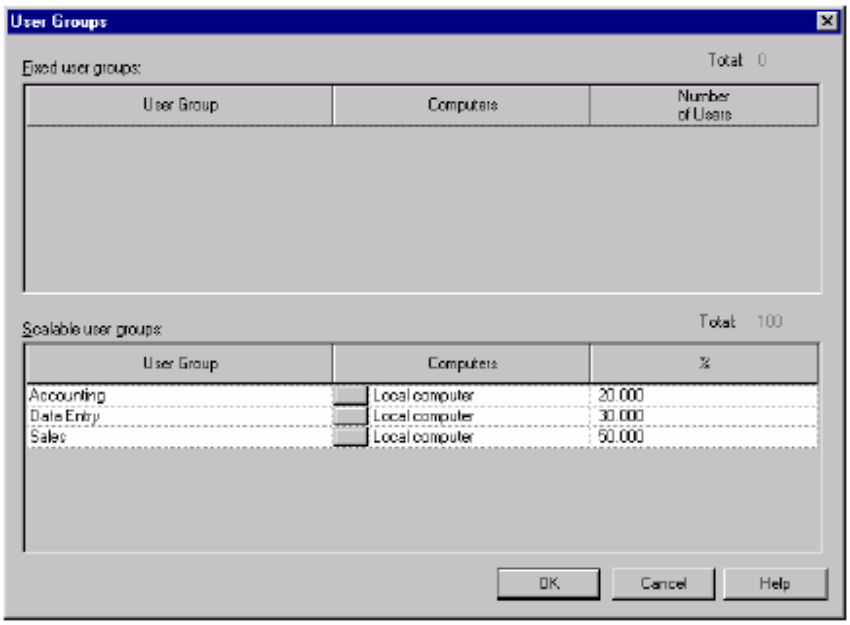
注意事项：当你编辑一组suite条目的执行属性时，这种变更只影响条目的实例。例如，如果你将一个测试脚本插入到一组suite，且插入了两次，并改变了第一个测试脚本的执行属性，那么第二个测试脚本依然会保持先前的执行属性。

针对所有用户和计算机组编辑信息（Editing Information for All User and Computer Groups）

有时候，你可能想要为一个以上的用户或计算机组编辑信息。虽然你可以单个的编辑每一个组，但为所有的组在相同的时间内来编辑信息是更容易的。

为所有的组编辑信息：

●打开一组suite，然后点击**Suite > Edit Groups**。

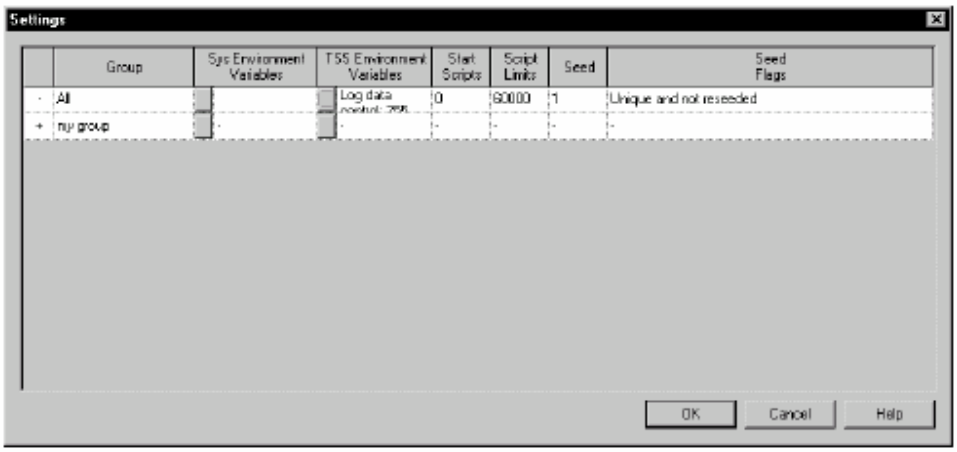


针对虚拟测试者编辑设置（Editing Settings for Virtual Testers）

你可以改变与虚拟测试者关联的默认设置。你可以设置TSS环境变量来改变你执行一组suite时登陆的信息。例如，如果你执行一组时suite有了问题，那么设置一个虚拟测试者去log所有的事件和其他虚拟测试者去失败log事件，只为了你可以更加充分地调查问题。

编辑虚拟测试者地设置：

●打开一组suite，然后点击**Suite > Edit Settings**。



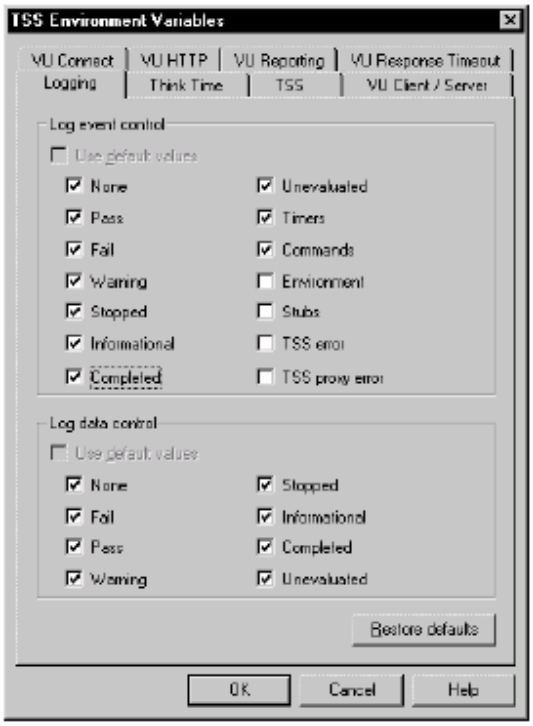
初始化 TSS 环境变量（Initializing TSS Environment Variables）

你可以通过TestManager初始化TSS环境变量的大部分值。通过TestManager，一个环境变量被初始化时，属性值是有效针对一个实体suite的执行，除非测试脚本对属性值有特殊的变更。

初始化一个TSS的环境变量：

1 打开一组suite，然后点击**Suite > Edit Settings**。

2 在Settings对话框中，点击**TSS Environment Variables**栏中按钮。



3 在TSS Environment Variables对话框中，选择下面的标签：

●**Logging**——Logging环境变量隶属于在suite中的任何测试脚本。它们允许你设置细节级别，这些细节出现在关联与一组suite执行的日志文件中。

●**Think time**——Think time环境变量隶属于在suite中的任何测试脚本。他们控制虚拟测试者的“think time”行为。这是一个典型的用户延迟，或考虑，及提交命令之间的时间。

●**TSS**——TSS能或不能提供测试脚本服务，将在下一节做描述。

●**VU Client/Server**——Client/server环境变量隶属于VU测试脚本，一个SQL数据库的入口。这些变量允许你包括SQL的行头（column headers），设置位数以包括一个响应，以及在一个SQL表的末尾停止数据恢复。

●**VU Connect**——Connect环境变量隶属于VU测试脚本，记录HTTP和socket协议。它们控制一个测试脚本的次数，重复试验链接一个服务器和重复使用测试脚本之间的延迟。

●**VU HTTP**——HTTP环境变量隶属于VU测试脚本，记录HTTP协议。这些变量允许你模拟对于http的排列速度和socket请求。它们也可以使一个HTTP测试脚本成功地录制回放，当然如果服务器有数据响应，这些数据不是确切地匹配于记录的内容——例如，服务器有局部数据响应，响应被保存，或者测试脚本在录制回放期间直接地到另一个服务器中。

●**VU Reporting**——Reporting环境变量隶属于VU测试脚本。这些变量允许你设置如何检查SQL unread row的结果和设置最小位数来接受一个SQL响应。

●**VU Response Timeout**——Response timeout环境变量隶属于VU测试脚本，记录HTTP, SQL, IIOP, 或者socket协议。这些变量允许你设置针对一个VU仿真命令（emulation command）的超时时间，按比例设置超时时间，以及设置活动来获取是否有出现一个超时。

注意事项：有关TSS的环境变量的信息，参阅*VU Language Reference*或Rational Test Script Services手册。

禁用 TSS（Disabling Test Script Services）

你可以在一组suite执行期间禁用TSS。禁用TSS可使你减少运行时间开支。禁用也允许你创建一个测试脚本，这个脚本使用所有的TSS，然后，基于可用的或禁用的，测试仅仅是功能关联的或是性能关联的服务。

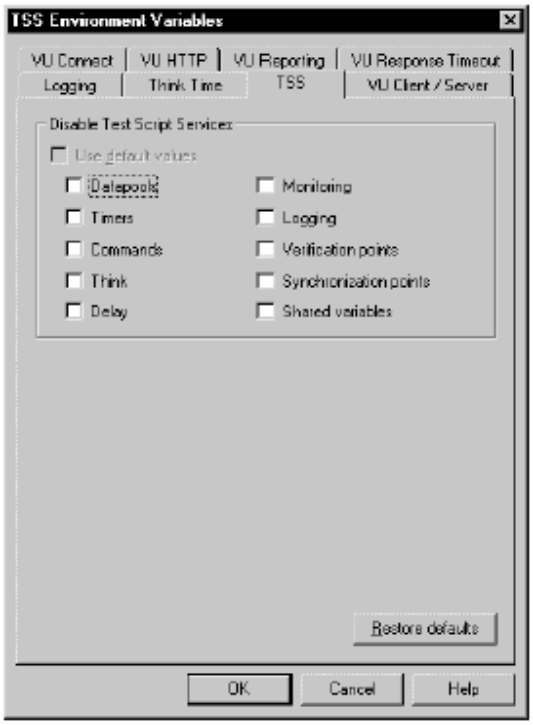
在默认情况下，没有TSS是禁用的。

禁用TSS：

1 打开一组suite，然后点击**Suite > Edit Settings**。

2 点击**TSS Environment Variables**栏中的按钮。

3 选择TSS Environment Variables对话框中的TSS标签。



可用或禁用的服务：

●**Datapools**——在测试脚本中datapools的使用。

●**Timers**——在测试脚本中timers使用。

●**Commands**——在测试脚本中使用TSS开始和结束命令。Commands将不被计时，命令信息不被作为事件而写入到日志中，这里也没有任何的报告来分析。

●**Think time**——在测试脚本中TSS 的使用think time常规事务。

●**Delays**——在测试脚本中TSS 的使用delay常规事务。

●**Monitoring**——一组suite的Monitoring执行。

●**Logging**——在一组suite执行期间Logging。

●**Verification points**——在测试脚本中对检验点（verification points）的使用。

●**Synchronization points**——在测试脚本中同步点（synchronization points）的使用。

●**Shared variables**——测试脚本中shared variables的使用。

变更测试脚本的开始数量（Changing the Number of Start Test Scripts）

如果在组内你开始虚拟测试者，那么TestManager允许你确定测试脚本的数量，当然，在下一个组开始之前该组的虚拟测试者必须完全成功。你可能需要这样做来控制虚拟测试者的最小数量，来在一个给定的时间内登陆一个系统。

例如，假定Data Entry用户组含有100个虚拟测试者。每个虚拟测试者执行一个登陆的测试脚本，然后以一个随机的顺序选择3个测试脚本：添加新的记录，修改记录和删除纪录。你已经变更了执行期间的设置使得那100个虚拟测试者并不是一次性的全部开始，而是开始该组的25个。

如果你设置第一组的脚本开始数量，那么第二组25个是在第一组25个的每一个虚拟测试者完成登陆测试脚本时开始。同理，第三组的25个开始于第二组的25个中的每一个虚拟测试者完成登陆测试脚本时，等等。

变更测试脚本的开始数量：

●打开一组suite，然后点击**Suite > Edit Settings**。

限制测试脚本的数量（Limiting the Number of Test Scripts）

TestManager允许你限制测试脚本的数量，这些测试脚本中的虚拟测试者可在没有从用户组中移除任何测试脚本的情况下执行。你可以测试suite的一个简略的执行，要确保所有的suite条目正确地工作。

例如，限制测试脚本的数量：

- 你在一个suite中不确定地重复一个序列之前，确保它可执行在一个有限制的时间。假定，你的suite有一个含有8个测试脚本的序列。确保虚拟测试者能够开始、执行以及完成这个序列两次，在测试脚本之前插入一个连续的选择器，并设置**Script Limits**到16。
- 从suite中暂时地禁用一个用户或测试机组，但不要删除它。通过设置对组的**Script Limits**到0，你就禁用了它。（你也可以通过设置虚拟测试者的数量到0以禁用一个固定的组。）
- 检查一个数据池正确地工作。例如，假定一个虚拟测试者输入记录到一个数据库100次。检查该虚拟测试者进入一个唯一的记录，设置**Script Limits**到2。如果一个虚拟测试者进入相同的记录两次，则此次执行失败。

限制测试脚本的数量：

- 打开一组suite，然后点击**Suite > Edit Settings**。

变更方法随机数被产生（Changing the Way Random Numbers Are Generated）

每一个虚拟测试者有一个“种子”（*seed*），它在一个测试脚本中产生随机数。这些随机数影响一个虚拟测试者的think time，随机数库常式（routines），以及在数据池中的随机入口。“种子”是，主要地，或唯一的，或相同的：

- 利用唯一的种子，每一个执行相同测试脚本的虚拟测试者有一个轻微不同的表现（行为）。例如，如果在执行第一个命令之前，一个虚拟测试者可能考虑（think for）1.3秒，第二个虚拟测试者可能考虑2.4秒。虽然这些单独的考虑时间是变化的，但是它们有围绕一个平均值的相同的分布。

  这些“种子”也影响库常式包含的随机数。例如，在VU语言中，如果第一个虚拟测试者调用了统一常式（`uniform` routine）两次并收到数字5和3，那么在那个组内的其他虚拟测试者就

可能收到不同的数字，只有被最小和最大的值绑定，这些值是在测试脚本中设置的。

●利用相同的种子，每一个执行相同测试脚本的虚拟测试者有确切的相同的表现。

例如：

◆如果在第一个命令执行之前，第一个虚拟测试者考虑1.3秒，第二个虚拟测试者（和所有随后的虚拟测试者）在那个命令执行之前也考虑了1.3秒。

◆如果第一个虚拟测试者调用统一常式两次并收到数字5和3，那么在那个组内的所有的其他虚拟测试者也接受到5和3。

你也可以设置随机数产生器在每一个测试脚本开始时是否是被再播种的（reseeded）。一般而言，不再播种是比较好的，因为一个长的假随机（pseudorandom）序列比起许多的短的要更真实。

“种子”可用下面的方法定义：

●*Unique and not reseeded*－－为每一个虚拟测试者产生一个唯一的“种子”（seed）和随机数产生器在每一个测试脚本开始时不再被播种（reseeded）。每一个在一个用户组中的虚拟测试者的表现（行为）有细微的不同。在性能测试中，这是最普通的使用选项。

●*Unique and reseeded*－－为每一个虚拟测试者产生一个唯一的“种子”（seed）和随机数产生器在每一个测试脚本开始时被再播种（reseeded）。每一个在一个用户组中的虚拟测试者的表现（行为）有细微的不同，但是该数字是在每一个测试脚本开始时被再播种的。该选项对于政府的LTD（Live Test Demonstration）测试是很有效的。

●*Same and not reseeded*－－为每一个虚拟测试者产生相同的“种子”（seed）和随机数产生器在每一个测试脚本开始时不再被“播种”（reseeded）。这不是一般的一个去选择的可取选项，在模型化一个现实的工作量时，因为执行相同测试脚本的每一个虚拟测试者是以相同的方法表示（行为）。然而，这个选项对于某种压力测试类型可能是有用的。

●*Same and reseeded*－－为每一个虚拟测试者产生相同的“种子”（seed）和随机数产生器在每一个测试脚本开始时被再“播种”（reseeded）。这不是一般的一个去选择的可取选项，在模型化一个现实的工作量时，因为执行相同测试脚本的每一个虚拟测试者是以相同的方法表示（行为），并且短的假随机（pseudorandom）序列不是现实的。

然而，该选项对于某种压力测试类型可能是有用的。例如，如果你有一组suite，它有一个共享的数据池，并且它的seed设置为唯一的和不再被“播种”，每个虚拟测试者和迭代有一个不同的“seed”，该“seed”通过所有的虚拟测试者和迭代给予随机数据。要在所有虚拟测试者获得相同的数据模式(pattern)时反复的查看会发生什么，那就针对所有的虚拟测试者将seed设置为相同的和再播种的。

要变更随机数产生器的默认行为：

●打开一组suite，然后点击**Suite > Edit Settings**。

产生随机数的VU常式（routines）是`NegExp`，`Rand`，以及`Uniform`。

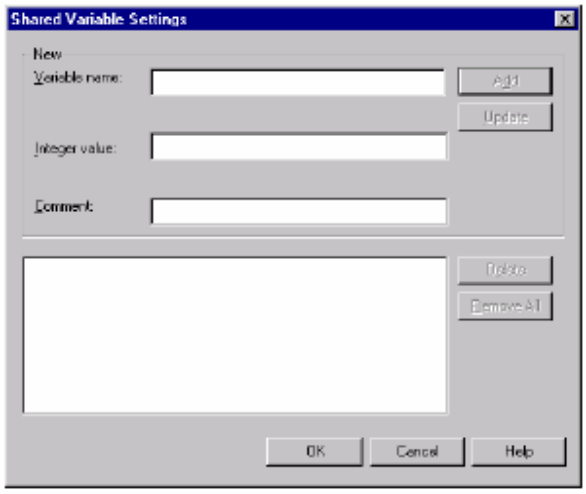
有关这些常式的信息，参阅*VU Language Reference*或Rational Test Script Services手册。

初始化共享变量（Initializing Shared Variables）

TestManager允许你在一组suite中初始化共享变量。一个共享变量通过所有的在一组suite中的测试脚本包含了它的值。每个虚拟测试者可以获取和变更这些共享变量。

要初始化一个共享变量：

●打开一组suite，然后点击**Suite > Edit Shared Variables**。



在下面的情况下，共享变量是有用的：

●在你需要比一个同步点提供的更加明确的协调时，使虚拟测试者同步。例如，你可以限制一个事务处理以使每次只有5个虚拟测试者执行它。在那种情况下，使用一个共享变量通过合适的等候常式在你的脚本语言中。

●从执行中阻断一个虚拟测试者直到一个全局事件发生。设置一个事件和一个依赖比设置一个共享变量更容易。然而，如果该事件依赖于一个测试脚本中的一些逻辑，那么你就必须使用一个共享变量。

●计算在一个测试脚本中的循环次数。如果你要为一个完整的测试脚本设置一次循环，那么在suite中设置一个选择器或一个迭代是更容易的。然而，如果仅仅是测试脚本的一部分循环的话，那就要设置一个共享变量以控制那次循环的迭代的数量了。

●监控特定事务处理的计数和情况。在你监控一组suite时，共享变量提供有关一组suite执行的过程和状态的细节信息。

你声明（*declare*）一个共享变量在一个测试脚本或资源文件中，通过使用一个“shared“常式。有关该常式的信息，参阅*VU Language Reference*或Rational Test Script Services手册。

你初始化（*initialize*）一个共享变量在一组suite中。这是可选择的——默认值为0。通过在一个测试脚本中的逻辑或在你监控该suite时，你可以使用一个共享变量的值。

打印和输入一组 Suite（Printing and Exporting a Suite）

设计一组suite可以包含许多的迭代和变更。你可能它帮助去检查一组suite的打印预览。你可以打印一组suite或作为一个“.txt”文件输出。

要打印一组suite：

●打开该suite以打印，然后点击**File > Print**。

要作为一个“.txt”文件输出：

●打开该suite以输出，然后点击**File > Export to File**。

保存一组 Suite（Saving a Suite）

你完成了一组suite的修改之后，要保存你的变更。一组不能保存的suite在标题栏中有一个星号。执行一组suite自动地执行一次保存。

要手动的保存一组suite：

打开该suite以保存，点击**File > Save**。

要保存一个以上的suite：

1 点击**File > Open Suite**去打开suites以保存。

2 点击**File > Save All**。

注意事项：如果你点击**Tools > Options**，点击**Create Suite**标签，然后选择**Check suite when saving**对话框，为每个要被保存的suite有一个检验屏幕出现。

要在一个不同的名称下保存一组suite：

打开该suite以保存，然后点击**File > Save As**。

测试的执行（Executing Tests） 5

本章描述如何执行测试。它包括了下面的标题内容：

●有关测试的执行

●对于执行测试脚本的内置支持

●自动化测试脚本的执行

●手工测试脚本的执行

●测试用例的执行

●Suites的执行

●Suites的监控

●Suites的停止

注意事项：对于细节过程，参阅TestMnaager的帮助。

有关测试的执行（About Running Tests）

执行你的测试脚本的活动主要是执行每个测试用例的实施，以此去验证（validate）测试用例打算验证的特定行为。

在 TestManager 中，你可以执行：

●自动化的测试脚本

●手工测试脚本

●测试测试用例

●Suites

对于执行测试脚本的内置支持（Built-in Support for Running Test Scripts）

TestManager提供对于下面的测试脚本类型执行的内置支持：

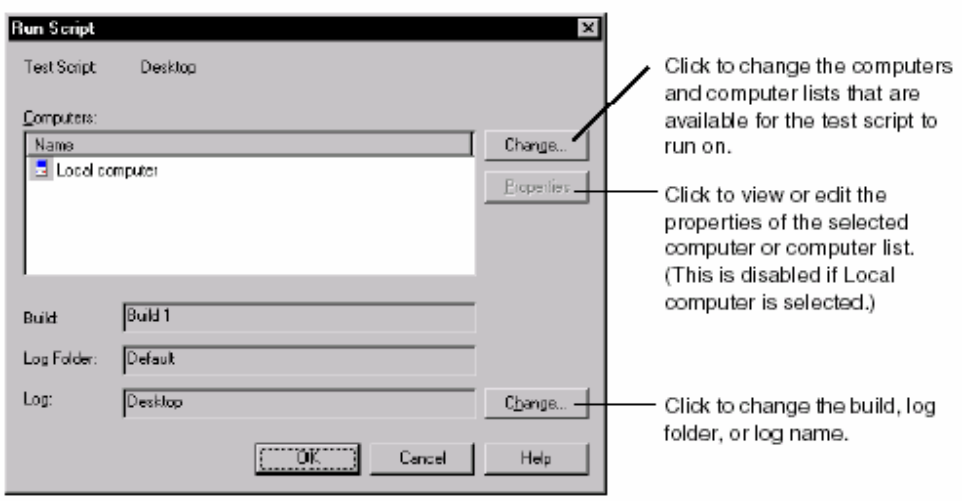
Type of Test Script	Description
GUI	A functional test script written in SQABasic, a Rational proprietary Basic-like scripting language.
VU	A performance test script written in VU, a Rational proprietary C-like scripting language.
Manual	A set of testing instructions to be run by a human tester.
VB	A test script written in the Visual Basic language.
Java	A test script written in the Java language.
Command line	A test script (for example, an .exe file, a .bat file, or a UNIX shell script) that can be executed from the command line.

自动化测试脚本的执行（Running Automated Test Scripts）

要执行来自TestManager的一个自动化的测试脚本：

1 点击**File > Run Test Script**，并选择测试脚本类型。

2 选择该测试脚本以执行并点击**OK**打开Run Script对话框。



当你点击**OK**时，TestManager可在测试机列表中的第一台可用的测试机上执行测试脚本。在测试脚本执行时，你可以监控它的过程并在晚些时候查看测试日志中的结果。

更多有关测试机和测试机列表的信息，参阅69页*Defining Agent Computers*和*Computer Lists*的内容。

有关过程监控的信息，参阅103页*Monitoring Suites*的内容。

有关测试日志的信息，参阅127页*About Test Logs*的内容。

注意事项：在你执行一个测试脚本时，不生成测试用例的覆盖结果。（即使是该测试脚本与一个测试用例相关联）。要生成测试用例结果，那就执行测试用例而非测试脚本。有关信息，参阅90页的*Running Test Cases*内容。

手工测试脚本的执行（Running Manual Test Scripts）

在你执行一个手工测试脚本时，你可以像下面这样做：

- 可任选地，设置一个执行选项去登陆未加抑制的步骤和验证点作为警告。（在Rational ManualTest中，选择**Tools > Options**）。
- 指明你已经执行了的每个步骤。
- 指明验证点首否通过或失败。

注意事项：有关创建手工测试脚本的信息，参阅61页*Creating Manual Test Scripts*的内容。

要执行一个手工测试脚本，如下面的做法：

- 在TestManager中，点击**File > Run Test Script > Manual**并选择一个测试脚本。Rational ManualTest打开。

●在Rational ManualTest中，点击**File > Run**并选择一个测试脚本。

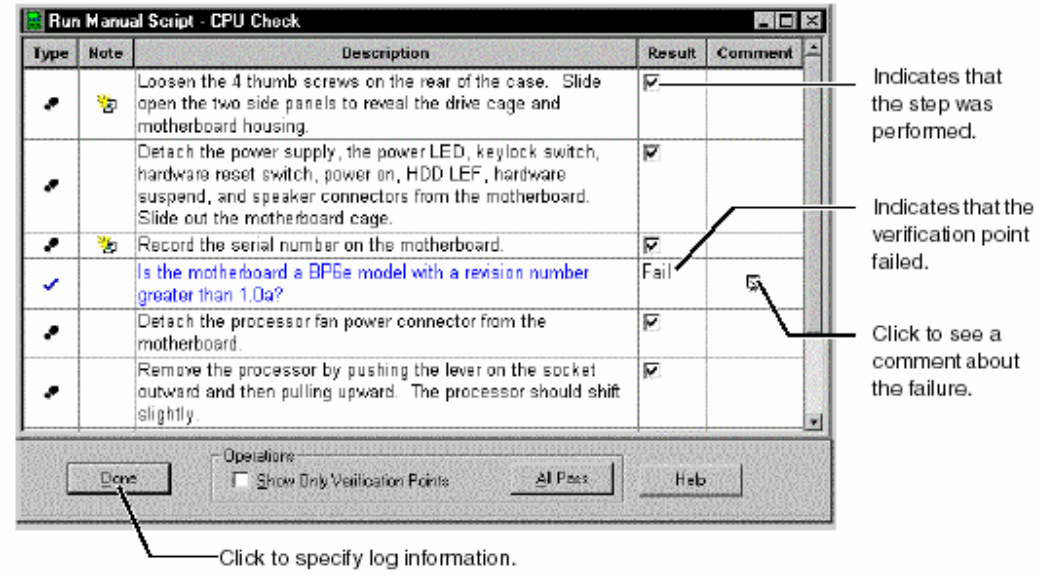
执行每一个在Run Manual Script窗口中列举的步骤和验证点：

●对于一个步骤，选择Result检查对话框以指明你已经执行了的步骤。

●对于一个验证点，点击Result单元，并点击None，Pass，或Fail。

例子（Example of Running a Manual Test Script）

下面的图说明一个手工测试脚本的执行结果。在该手工测试脚本执行时，第一个验证点是失败的。Comment的图标指明这里有一个关于失败的注释。在你查看测试日志时，你将能够看到失败和相关的注释。



在你执行了一个手工测试脚本之后，你可以查看TestManager中测试日志中的结果。

For information about test logs, see *About Test Logs* on page 127.

有关测试日志的信息，参阅 127 页 *About Test Logs* 的内容。

测试用例的执行（Running Test Cases）

当你执行一个测试用例时，你实际上执行的是该测试用例的实施。该实施是一个关联于这个测试用例的一个测试脚本或suite。

查看被关联的实施（Viewing the Associated Implementations）

要查看或变更关联于一个测试用例的实施：

1 在Test Plan窗口中，右键点击一个测试用例，然后点击**Properties**。

2 点击**Implementation**标签。

你可以有最多的2个实施关联到一个测试脚本中：一个手工的，一个自动的。在你执行测试脚本时，TestManager将执行那个自动化的实施。

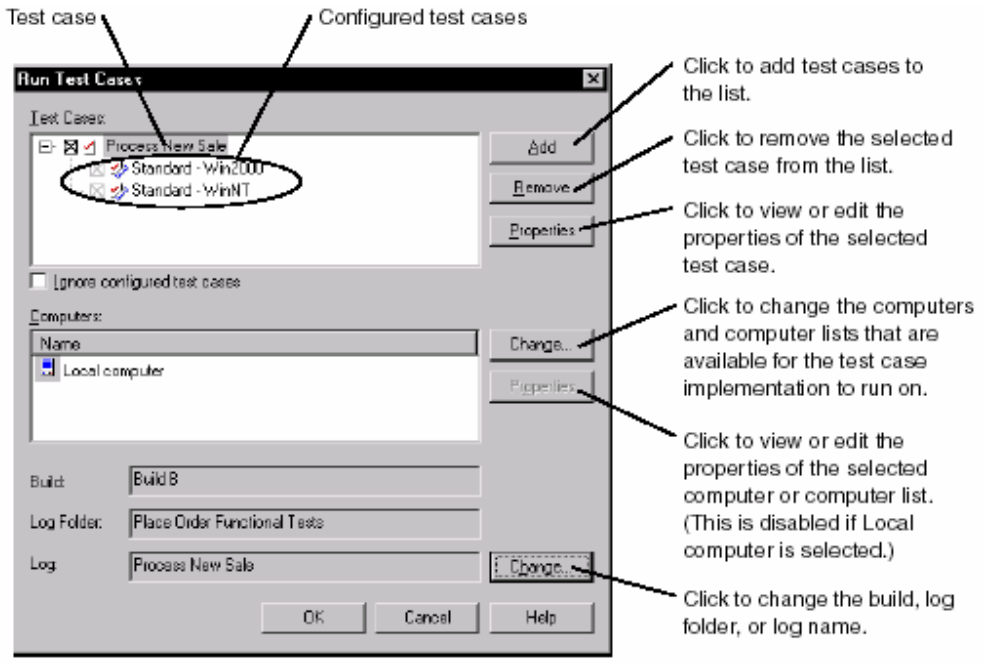
注意事项：如果你执行来自于Rational ManualTest Web Execution组件的一个测试用例，那么将执行手工的测试用例。有关信息，参阅351页*About ManualTest Web Execution*的内容。

有关测试用例的实施和实施关联的信息，参阅55页*Implementing Tests*的内容。

执行一个测试用例（Running a Test Case）

要执行一个测试用例，如下面的方法：

- 点击**File > Run Test Case**。选择测试用例以执行并点击**OK**。
- 点击**File > Run Test Cases for Iteration**。选择迭代并点击**OK**。



注意事项：在你执行一个测试用例时，TestManager一个临时的suite并实际执行这个suite。在该suite执行完成之后，TestManager会移除它。

When you click **OK**, TestManager runs the test case as follows:

在你点击**OK**，TestManager依下列情况执行这个测试用例：

- 如果你执行一个已经有了一个自动实施的测试用例，那么它会执行在列表中的第一台可利用的测试机上。

  一个配置的测试用例将只能在与该测试用例的配置完全匹配的测试机上执行。TestManager会考虑该测试机的内置配置属性的值和该测试机的tmsconfig.csv文件中确定的常规属性的值。（有关tmsconfig.csv文件的信息，参阅38页的*Setting Up Custom Attributes in tmsconfig.csv*。）例如，假设这个配置的测试用例指明它将会执行在一台有Windows 2000的测试机上，那么TestManager会检查每个在Computers列表中的测试机，直到它找到了一台有Windows 2000的机子，然后在这台测试机上执行这个测试用例。如果在这个列表中没有与该配置匹配的测试机，那么在测试日志中会有出现一条消息。
- 如果你执行一个已经有了一个手工实施而非自动实施的测试用例，那么Rational ManualTest会启动。你可以在该手工测试脚本中执行这些步骤和验证点，然后在测试日志中查看结果。有关信息，参阅89页*Running Manual Test Scripts*的内容。

忽视配置的测试用例（Ignoring Configured Test Cases）

如果你选择Run Test Cases对话框中的**Ignore configured test cases**检查框，那么TestManager将忽视该配置并将在任意一台可用的测试机上执行该测试用例。

- 如果选择的测试用例已有配置的测试用例和一个实施（例如，一个测试脚本或suite），那么TestManager会在任意一台可用的测试机上执行这个被选择的测试用例，但是不会执行任意一个配置了的测试用例。
- 如果选择的测试用例已有配置了的测试用例但无一个实施，那么TestManager不执行这个测试用例或任意一个配置了的测试用例。
- 如果是一个单独的已配置测试用例被选择，那么TestManager会在被指定的配置下执行该测试用例。

Suites 的执行（Running Suites）

执行一组suite将包含下面的步骤：

- 检查该suite。

●检查代理（Agent）测试机。

●控制该suite的执行期信息。

●控制如何终止该suite。

●为suite的执行确定虚拟测试者和配置。

●终止该suite。

下面的章节将描述这些步骤。有关创建一个功能测试suite的信息，参阅165页*Creating Functional Testing Suites*的内容。有关创建一个性能测试suite的信息，参阅225页*Designing Performance Testing Suites*的内容。

检查一组 Suite（Checking a Suite）

当你工作在一组suite中时，你可能改动了它以至它不能正确地执行。

例如，你可能在一组suite被记录之前插入了一个测试脚本到它的里面。虽然TestManager在一组suite执行之前会自动地对它进行检查，但是，你可以在没有实际执行它的情况下来检查一组suite。这可以帮助你确定和纠正问题。

要检查一组suite：

1 点击**File > Open Suite**，点击一组suite。

2 点击**Suite > Check Suite**。

TestManager针对多种错误类型检查一组suite，包括下面的内容：

●该suite不包含任意用户或测试机组。一组suite必须有至少一个用户或测试机组来执行。

●该suite包含一个空用户或测试机组。不是删除该用户就是测试机组，或者添加测试脚本和其他条目到suite中。

●一个或测试机组包含一个空的剧本。不是删除该剧本就是添加条目到其中。

●该suite包含一个空的选择器。不是删除这个选择器就是添加属性到其中。

●一个在该suite中的测试资产（测试机，测试机列表，测试脚本，suite，或测试用例）已被删除。

注意事项：你可以设置选项以便你无论何时保存该suite时，它都会自动地被检查。要自动地检查该suite，点击**Tools > Options**，点击**Create Suite**标签，并选择**Check suite when saving**检查对话框。

检查代理测试机（Checking Agent Computers）

如果你在代理测试机上执行虚拟测试者，那么在你执行suite之前检查代理是个好主意。用这个方法，你可以在你执行该suite之前确定是否有问题存在。

在你检查代理（Agent）测试机时，TestManager确保：

●对于虚拟测试者确定的所有代理测试机实际存在。

例如，如果你错误地输入了一台代理测试机的名称，那么TestManager会提示你。

●该代理测试机是可用的和运行的。

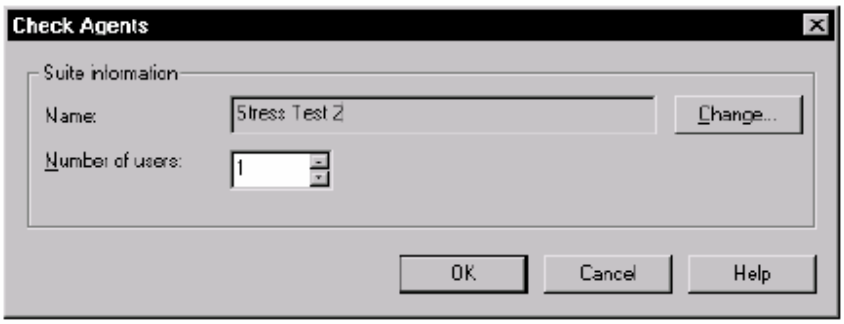
●代理软件是运行的。

TestManager软件的相同发布必须被安装在本地和代理测试机上。

要检查代理测试机：

1 点击**File > Open Suite**，选择一组suite。

2 点击**Suite > Check Agents**。



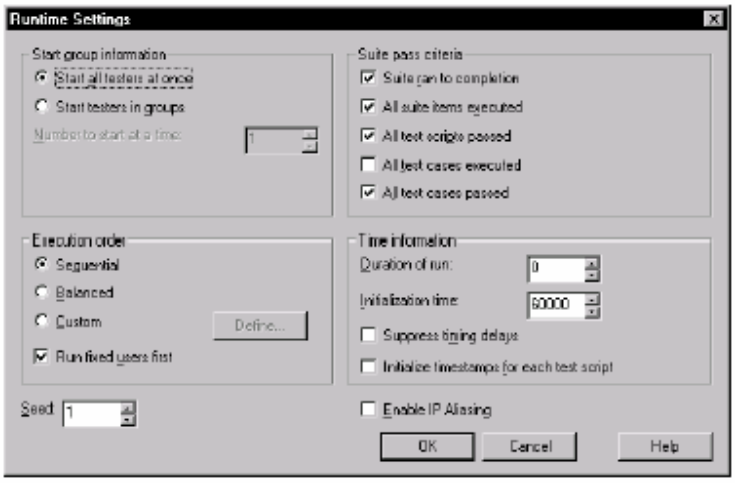
TestManager 在一个单独的窗口内显示代理测试机的任何问题。

控制一组 suite 的执行期信息（Controlling Runtime Information of a Suite）

针对一组suite设置执行期的设置：

1 点击**File > Open Suite**，选择一组suite。

2 点击**Suite > Edit Runtime**。



通过对执行期设置的修改，你可以控制：

●**启动组的信息（Start group information）**——在性能测试中，控制虚拟测试者如何开始。

●**Suite 通过标准（Suite pass criteria）**——针对一组 suite 是否通过或失败的标准。

●**执行顺序（Execution order）**——在性能测试中，控制顺序以确定哪一个用户组会开始。

●**时间信息（Time information）**——控制 suite 执行的时间长度。

●**种子（Seed）**——设置数字以提供给随机数字产生器。

●**可进行 IP 别名判断（Enable IP aliasing）**——在 VU HTTP 脚本中控制别名判断。

启动组信息（Start Group Information）

在性能测试中，**Start group information**控制虚拟测试者如何开始。虚拟测试者或一次全部启动或在组内启动。

要避免一个服务器超负载，在一个组内启动虚拟测试者，并确认该启动组中虚拟测试者的数量。

注意事项：如果你在组内启动虚拟测试者，那么你也应该对于该组确认开始的脚本数量。

这样做，点击 **Suite > Edit Settings**，修改 **Start scripts** 对话框。

Suite 通过标准（Suite Pass Criteria）

Suite pass criteria控制一组suite是否通过或失败。选择下面的一个：

●**Suite执行到完成（Suite ran to completion）**——Suite在执行中没有手工终止的情况下必须执行到完成为止。

●**所有的suite条目都执行（All suite items executed）**——所有在suite中的条目都必须完成它们的分配工作。

●**所有的测试脚本都通过（All test scripts passed）**——无失败事件，无超时指令。

●**所有的测试用例都执行（All test cases executed）**——在suite中的所有测试用例必须完成他们全部的分配工作。

●**所有的测试用例都通过（All test cases passed）**——所有的测试用例都通过，它的意思是说应用程序的测试符合给定测试用例的测试目标。

如果不符合标准，Test Log窗口列举的Suite Start和Suite End事件为“失败”。

执行顺序（Execution Order）

在性能测试中,执行的顺序定义哪个虚拟测试者开始的顺序，如果你执行的虚拟测试者少于被定义的最大数量，那么你可以决定哪个用户组可被执行。

选择下面的其中之一：

●**连续的（Sequential）**——以一个连续的顺序执行的Suites可运行每一个在该suite中出现的虚拟测试者（从顶部到低端）。

●**均衡的（Balanced）——**以一个均衡的顺序执行的Suites均匀地分布与该suite成比例的用户组之间的执行。

●**执行固定的用户先（Run fixed users first）——**在可按比例增减用户组前执行固定的用户组，不考虑执行顺序。如果你的用户组是全部固定不变的，那这个选项就没有作用。

假定你已经定义了四个总数有101个虚拟测试者的用户组，你的suite在此顺序中包含它们：

●End of Month Accounting：1个虚拟测试者

●Accounting：20个虚拟测试者

●Data Entry：30个虚拟测试者

●Sales：50个虚拟测试者

虽然你的suite包含101个虚拟测试者，但你要执行它的话，只能仅仅有10个虚拟测试者。

下面的表罗列了执行的顺序如何影响用户的执行：

If you select	You run these user groups
Sequential	1 End of Month Accounting 9 Accounting
Balanced	2 Accounting 3 Data Entry 5 Sales
Balanced and Run Fixed Users First	1 End of Month Accounting 2 Accounting 3 Data Entry 4 Sales

●常规的——以一个常规的顺序执行的suites需要你去选择特定的用户组或虚拟测试者来执行。只能对固定不变的用户组实施一个常规的执行顺序。

以一个常规的顺序执行用户组对于错误命中（troubleshooting）是有利的。例如，如果一个测试脚本不工作，并且这个测试脚本仅仅通过Accounting组被使用，那么就只能执行这个组了。

要创建一个常规的执行顺序，点击Runtime Settings 对话框中的**Define**。

你也可以暂时地禁用一个固定不变的用户组，通过对它的选择，然后点击**Edit > Run Properties**，设置**User Count**到0。

你可以先执行固定不变的虚拟测试者，从而在可按比例增减用户组之前执行固定不变的用户组，不考虑执行顺序。如果你的用户组是全部固定不变的，那这个选项就没有作用。

时间信息（Time Information）

这个选项控制诸如下面的时间信息：

●执行应当获得的最大时间数量。一个 0 值的开始增加无时间限制。

●对于所有虚拟测试者的最大秒数来确认它们完成的初始设定。如果你已经改变了测试脚本的开始数量，那么就确认你设置该时间的足够的高度。

●防止时间的延迟，从而很快速地执行该 suite。如果是测试一组 suite 来查看它的执行是否正确，并且你对时间的延迟不感兴趣，那么这个选项是有利的。然而，这需要在服务器、本地机和代理测试机上创建一个最大工作量。

如果你是执行大量的虚拟测试者，那么就不要防止时间的延迟。

●为每个测试脚本初始化时间标志，这些脚本指明时间标志是否结束从一个脚本到一个脚本的携带，或者是随每个脚本重新初始化。

Seed

该选项设置数字以提供给随机数字产生器。

TestManager使用一个被说明的seed来为选择器产生随机数字并在数据池中共享入口。

IP 别名判断（IP Aliasing）

在VU测试脚本中使用HTTP协议，该选项设置是否使用IP别名判断。

IP别名判断需要每个虚拟测试者有一个不同来源的IP地址。这是有意义的，如果你只是在执行HTTP的测试脚本，和你的系统管理员已经设置了你的测试机使用IP别名判断。

有关设置IP别名判断的信息，参阅331页*Configuring Local and Agent Computers*的内容。

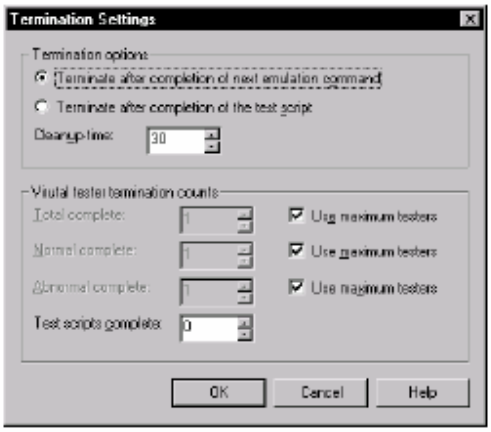
对如何终止一组 suite 的控制（Controlling How a Suite Terminates）

你可以设置状态强制一组suite以停止执行。例如，如果大量的虚拟测试者是不规则地完成，那么你可能要停止一组suite，指明在运行时出现的错误。

要对如何终止一组suite进行控制：

1 点击**File > Open Suite**，并选择一组suite。

2 点击**Suite > Edit Termination**。



一组 suite 的执行（Running a Suite）

在你执行一组suite时，你提供运行期－特定（runtime-specific）指南。每个执行分派给它的suite条目的虚拟测试者在这些指南的范围中执行。

TestManager在执行和编译任何不编译或过期的测试脚本之前检查该suite。

TetsManager保存suite的执行结果到测试日志中。在你执行该suite之后，执行报告来分析保存在这些日志中的数据。你可以用曲线图表和图解的形式来展现这些信息，或是以更常规的报告的形式。

当你执行一组suite时，你说明：

●虚拟测试者的数量，如果你是执行一组性能测试的suite的话。

如果一组suite既包含固定不变的又包含可按比例增减的用户组，那么固定不变的用户组是首先被分配的。如此，举例来说，如果你的suite包含一个固定分配了10个虚拟测试者的用户组，并且你要执行100个虚拟测试者，那么这10个虚拟测试者被分配到这个固定不变的用户组中，剩下的那90个虚拟测试者被分布在可按比例增减的用户组之间。

注意事项：可用的虚拟测试者的数量依赖于你所拥有的许可的类型。如果你的许可不能支持你说明的虚拟测试者的数量，你可以查看错误信息。

●测试机的数量，以执行suite的和可用测试机的列表中的测试机数量。

如果你是执行一组功能测试的suite并且没有指明资源的话。

●日志信息，包括build数量，日志文件夹，和日志文件名。

通过默认方式，日志文件夹的名称是基于suite的，日志名称是基于虚拟测试者数量和你已经执行suite的次数的。例如，如果你执行了实例suite3次，分别包含了10虚拟测试者，15个虚拟测试者，和20个虚拟测试者，那么全部的三个测试日志将被放在这个实例suite中。而这些日志名称将会是Users 10 #01，Users 15 #02，和Users 20 #03。因此，日志名Users 20 #03指明了这已经是你第三次执行该suite，而且这个suite执行时包含了20和虚拟测试者。

你可以在Options对话框的**Run**标签中更改这些设置。更多信息，参阅TestManager的帮助。

●资源监控。

在你监控一组suite时，TestManager记录测试机资源是如何被利用的，并且在你分析你的结果时，用图表表示覆盖于对应虚拟测试者的响应时间的这些使用数据。指定监控视点的时间间隔被更新；更低的时间间隔，更快的更新。

●是否要忽略关联的配置。

选择 **Ignore configurations for test cases** 检查框以使 TestManager 忽略配置并在任意可用的测试机上执行该测试用例。

TestManager 有三种执行测试用例的方法，如果这个选项被选择的话：

☐如果被选择的测试用例有配置的测试用例和一个实施（例如，一个测试脚本或 suite），那么 TestManager 会在任意可用的测试机上执行被这个选择的测试用例，但是不能执行任意的配置测试用例。

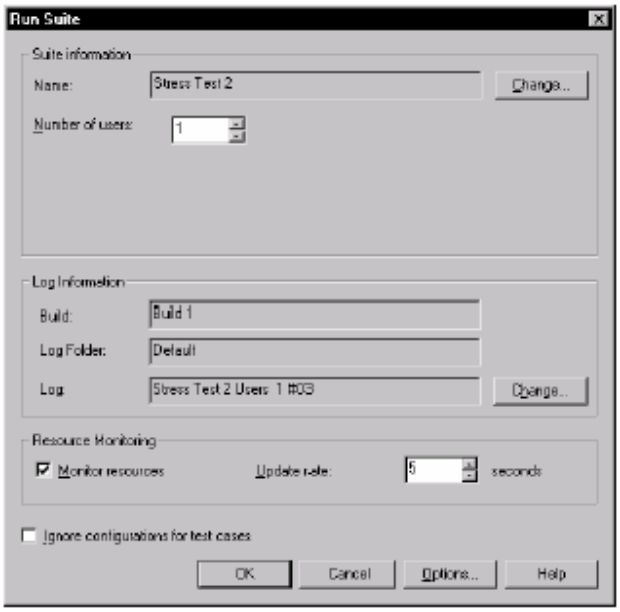
☐如果被选择的测试用例有配置的测试用例但无一个实施，那么 TestManager 不能执行该测试用例或任意的配置测试用例。

☐如果一个单独的配置测试用例被选择，那么 TestManager 会在指定的配置下执行该测试用例。

从 TestManager 中执行一组 suite（Running a Suite from TestManager）

从TestManager中执行一组suite：

点击**File > Run Suite**。



TestManager的该对话框显示的，依赖于你执行的suite的类型。

如果你是执行一组性能测试suite，那么你必须指定虚拟测试者的数量来执行。

如果你是执行一组功能测试suite，而且你没有指定测试机资源，那么你就必须要指定执行该suite的测试机。

从 Command Line 中执行一组 suite（Running a Suite from the Command Line）

你可以从command line中或以批处理的模式执行一组suite。例如，如果你有长时间执行的一组suite，或者你要一次执行几组suites，那么你可以排定它们在任何时候以方便的批处理方式来执行。在你从command line执行一组suite时，TestManager应用程序打开，执行这个suite，并可选择地，关闭。

The following syntax shows how to run a suite from the command line:

下面的语法展示如何从command line执行一组suite：

(rtmanager.exe suitename /runsuite /user *userid* [/password *password*] /project *.rsp path* [/computers [Local];
[*computer 1; computer 2; ... computer n*]
[/computerlists *computerlists*] /build *buildname*
/logfolder *logfoldername* /log *logname* [/overwritelog] [/numusers *nnn*]
[/ignoreconfiguredtestcases] [/close]**)**

rtmanager.exe suitename /runsuite /user 用户ID[/password 密码] /project .rsp路径[/computers [Local]；[测试机 1；测试机 2；...测试机 n] [/computerlists 测试机列表] /build build名称 /logfolder 日志文件夹名称 /log 日志名称[/overwritelog] [/numusers 数量][/ ignoreconfiguredtestcases] [/close]

If an argument contains spaces, enclose it in quotation marks.

如果一个argumengt中包含空格，封装它到引用标志中。

Syntax Element	Description
suitename	The name of the suite to run.
/runsuite	Runs the suite referenced in suitename.
user *userid*	The user name for logging on to TestManager.
password *password*	An optional password for logging on to TestManager. Do not use this option if there is no password.
/project *.rsp path*	The full path to the project's .rsp file.
/computers [Local]; [*computer 1; computer 2;... computer n*]	The names of the computers to use when running the suite. **Local** is the computer on which TestManager is running. This option is ignored if the specified suite indicates the computers on which to run it. If the suite does not indicate the computers or computer lists to use, this option is required.
/computerlists [*computerlist 1; computerlist 2,... computerlist n*]	The computer lists to use when running the suite. This option is ignored if the specified suite indicates which computers to use to run it, or if you specified computers in the /computers option.
/logfolder *logfoldername*	The name of the log folder in which to create the test log. If the log folder does not exist, TestManager creates it.
/build *buildname*	The build in which to create the log folder. If the specified build does not exist, TestManager creates it.
/log *logname*	The name of the test log in which to record the results of running the suite. If the log exists, you must specify the overwritelog option.
/overwritelog	Overwrite the test log if it exists. If you omit this option and the log exists, a message appears that says the log exists and cannot be overwritten.
/ignoreconfiguredtestcases	Ignore configuration matching (such as test cases that specify computers with specific operating systems or other attributes) when running test cases that are in the specified suite.

Syntax Element	Description
/numusers *nnn*	The target number of virtual testers when executing a suite that contains user groups. This is mandatory for running performance testing suites.
/close	Closes TestManager after running the suite.

语法元素	说明（描述）
suitename	要执行的 suite 的名称
/runsuite	以 suitename 为参考执行 suite

user *userID*	登陆 TestManager 的用户名
password *password*	登陆 TestManager 的一个可选密码。如果没有密码则不用该选项。
/project *.rsp path*	到项目的.rsp 文件的完整路径。
[/computers[Local];[*computer 1; computer 2; ... computer n*]	在执行 suite 时使用的测试机的名称。Local 是运行 TestManager 的测试机。如果被指定的 suite 已说明了执行它的测试机，那该选项可被忽略。如果被指定的 suite 未说明执行它的测试机或测试机列表，那么该选项是必需的。
[/computerslist;[*computerlist 1; computerlist 2; ... computerlist n*]	在执行 suite 时使用的测试机列表。如果被指定的 suite 已说明了执行它的测试机，或者被指定的测试机已在/computers 选项中，那该选项可被忽略。
/logfolder *logfoldername*	创建测试日志的日志文件夹名称。如果日志文件夹不存在，TestManager 将创建它。
/build *buildname*	以创建日志文件夹的 build。如果被指定的 build 不存在，TestManager 将创建它。
/log *logname*	以记录 suite 执行结果的测试日志的名称。如果该日志已存在，那么你必须指定/overwritelog 选项。
/overwritelog	如果该测试日志已存在则覆盖它。如果你遗漏该选项并且日志已存在，那么会有消息提示说日志存在和不能被覆盖。
/ignoreconfiguredtestcases	忽略配置匹配（诸如测试用例，指定了它含有的操作系统或其他属性的测试机）当执行在被指定的 suite 中的测试用例时。
/numusers *nnn*	执行包含用户组的 suite 时，虚拟用户的目标数量。这对于性能测试 suites 的执行是强制的。
/close	执行 suite 之后关闭 TestManager。

例子（Example）

这个例子执行的suite命名为**MySuite**。

```
rtmanager.exe MySuite /runsuite /user admin /project "C:\Sample
Project\Sample.rsp" /computers Local /build "Build 1" /logfolder
Default /log Sample /close
```

在这个例子中，TestManager登陆到项目C:\Sample Project\Sample.rsp作为虚拟测试者admin和在本地测试机上执行**MySuite**。执行**MySuite**的结果被保存在测试日志**Sample**中，它在build **Build 1**中的日志文件夹**Default**中被创建的。在此次执行完成后，TestManager关闭。

要使得suite执行的结果保持在屏幕上，忽略**/close**选项。该结果也会出现在默认的记录中。

Suites 的监控（Monitoring Suites）

当一组suite正在执行时，你可以监控它的进展。你可以：

●确认一组suite正在成功地进行。

●在执行时尽早地发现潜在的问题，因此有必要的话，你可以干预。

●挂起和重起虚拟测试者。

●更改共享变量的值。

●在同步点上释放虚拟测试者等待。

TestManager的监控工具提供给你最新的信息，该信息在suite执行时动态地更新。

该信息包括：

●已成功执行的命令数量和失败的命令数量。

●虚拟测试者的一般状态：是否初始化，连接到一个数据库，存在suite，或执行其他的任务。

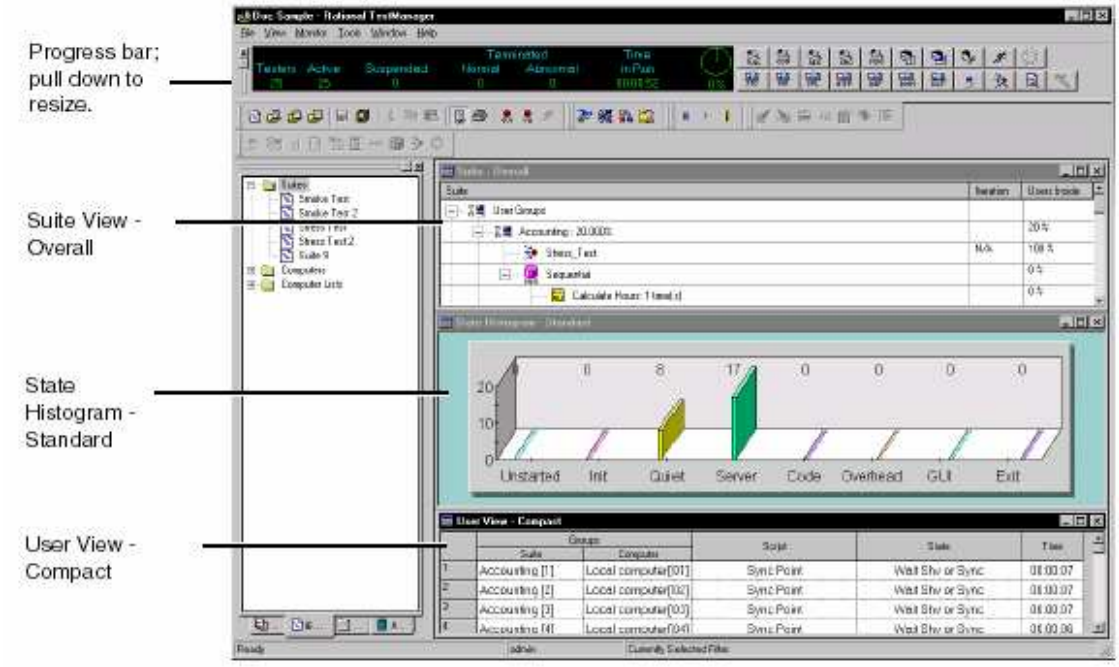
●是否有任意的虚拟测试者不正常地终止。

注意事项：本节的重点在于对suite执行的监控。然而，你也可以用同样的方法监控测试用例和测试脚本的执行。

Progress 栏和默认视图（The Progress Bar and the Default Views）

当你执行一组suite时，TestManager在一个Progress栏中和视图来显示监控信息。Progress栏给你一个执行状态的快速摘要并且不能被变更。你可以改变视图（views），然而，要提供关于每个虚拟测试者的摘要信息和细节信息。

下图展示了Progress栏和默认视图：



通过Progress栏，你可以快速地评估suite如何成功地执行。

Progress栏提供以下信息：

●**测试者（Testers）**——执行中的虚拟测试者的全部数量。

●**活动（Active）**——既不是挂起也不是终止的虚拟测试者的数量。

●**挂起（Suspended）——**处于暂停状态中的虚拟测试者的数量。

●**终止：正常的（Terminated: Normal）——**成功地完成它们的任务的虚拟测试者的数量。

●**终止：反常的（Terminated: Normal）——**未完成它们被指派任务时的终止状态的虚拟测试者的数量。

●**执行时间（Time in Run）——**suite已经执行的时间，表示为小时：分钟：秒。

●**完成率（% Done）——**suite已经完成的近似百分率。

TestManager也展现suite执行的三种视图：

●**Suite——总视图（ Overall view）**，展现有关虚拟测试者的状态的一般信息。更多信息，参阅105页*Displaying the Suite Views*的内容。

●**状态直条图（State Histogram）——标准（Standard）**，是一种直条图，它提供虚拟测试者正在执行的任务的一般信息。例如，一些虚拟测试者可能正在初始化，一些虚拟测试者可能正在运行代码，还有一些虚拟测试者可能在连接数据库。该图表展现了在每种状态下的虚拟测试者的数量。

TetsManager展现状态直条图——默认为标准。然而，如果你正在执行GUI测试脚本，或是

正在测试一个SQL数据库或一个Web服务器，你可以展现一个直条图逐一地连接起那些测试。更多信息，参阅107页*Displaying the State Histograms*的内容。

- **用户视图（User View）－紧凑的（Compact），**或**测试机视图（Computer View）－紧凑的（Compact）**，展现虚拟测试者的当前状态的信息。在该视图中，你可以点击一个特定的虚拟测试者来展现它的附加信息或控制它的操作。更多信息，参阅109页*Displaying the User and Computer Views*的内容。

Suite 视图的展现（Displaying the Suite Views）

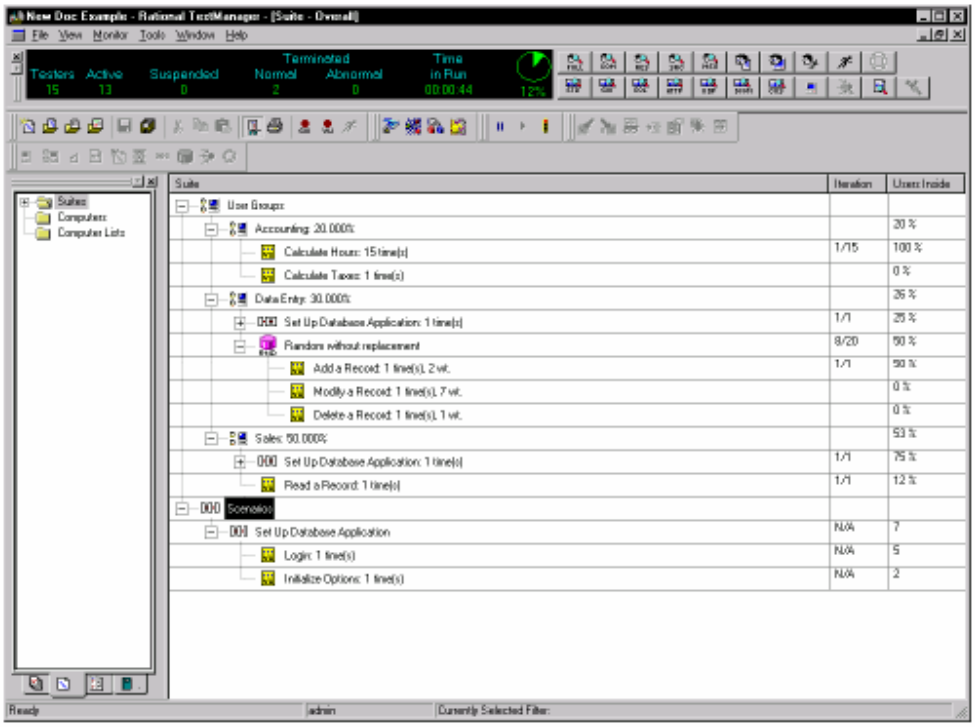
Suite视图非常类似于你已经设计的实际的suite。“栏（Columns）”表示给你被执行的迭代和一个测试脚本中的当前组或一个选择器中的虚拟测试者百分率。

- **Suite －总图（Overall）－－**使用该视图以展现有关suite状态的一般信息。TestManager在你执行一组suite时通过默认方式展现该视图。
- **Suite －用户（Users）** or **Suite －测试机（Computers）－－**使用该视图以展现一个特定的虚拟测试者的确切的suite进展。

Suite－－总视图（The Suite - Overall View）

要展现suite－总视图：

- 在suite执行期间，点击**Monitor > Suite > Overall**。



Suite－总视图是相似于你已经设计的实际的suite。

然而，它包含了两个附加栏（columns）：

Iteration栏表示在suite条目中和进展中有多少的迭代，平均覆盖当前执行的那个suite条目的所有虚拟测试者。

例如，8/20指明了，对于当前执行的那个suite条目的所有虚拟测试者，一般说来，虚拟测试者正在执行全部20个中的第八个迭代。

Users Inside栏表示当前执行suite的每个部分的虚拟测试者的百分率。该百分率贴近用户组表示全部虚拟测试者的百分率，这些虚拟测试者已经被指派到该组并且还没有退出suite。该百分率贴近一个用户组内的条目表示执行那个条目的组内的虚拟测试者的百分率。

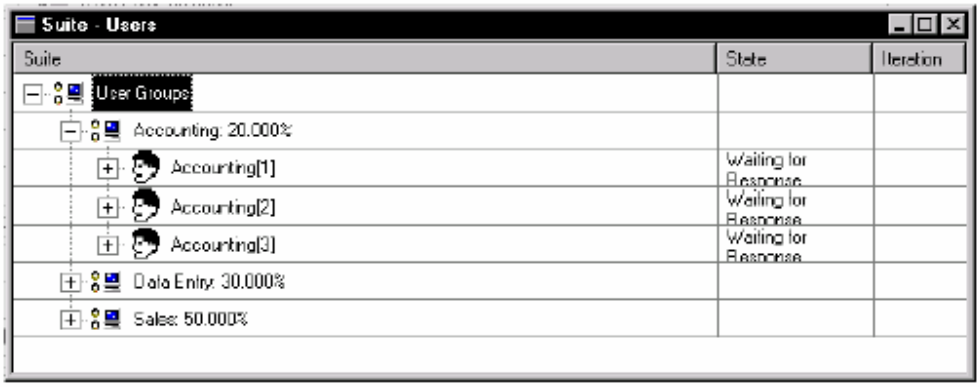
例如，如果Sales用户组包含全部虚拟测试者的百分之50，那么对于这个组在Users Inside栏中为50％。如果在该Sales组中的所有虚拟测试者正在执行Read Record测试脚本，那么对于这个测试脚本在Users Inside栏中为100％。

用户和测试机视图（The Suite - Users and Suite - Computers Views）

要展现suite－用户或suite－测试机视图：

●在suite执行期间，点击**Monitor > Suite > Users**来展现一个suite－用户视图。

点击**Monitor > Suite > Computers**来展现一个suite－测试机视图。



在该视图中的各栏的有关信息，参阅TestManager帮助索引中的*Suite - Users* or *Suite - Computers*。

状态直条图的展现（Displaying the State Histograms）

状态直条图分类虚拟测试者到不同的状态中去，诸如退出和初始化。使用一个直条图来展现在每种状态中分布的虚拟测试者的数量。

要展现一个状态直条图：

●在一组suite执行期间，点击**Monitor > Histogram**，并选择期望的直条图。

状态直条图是：

●**标准（Standard）**——数据以一般的方式被分类聚集。如果你要虚拟测试者状态的一个一般的总视图，那你就选择该视图。该视图中各栏的有关信息，参阅TestManager帮助索引中的*State Histogram - Standard*。

●**GUI**——针对执行GUI测试脚本的测试，数据适当地被分类聚集。该视图中各栏的有关信息，参TestManager帮助索引中的*State Histogram - GUI*。

●**SQL**——针对存取SQL数据库的测试，数据适当地被分类聚集。该视图中各栏的有关信息，参TestManager帮助索引中的*State Histogram - SQL*。

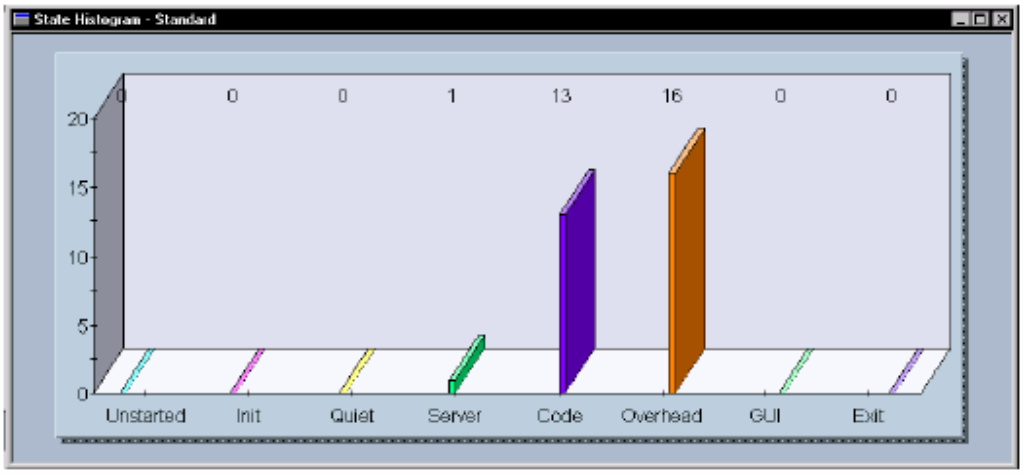
●**HTTP**——针对存取Web服务器的测试，数据适当地被分类聚集。该视图中各栏的有关信息，参TestManager帮助索引中的*State Histogram - HTTP*。

●**IIOP**——针对存取IIOP服务器的测试，数据适当地被分类聚集。该视图中各栏的有关信息，参TestManager帮助索引中的*State Histogram - IIOP*。

●**DCOM**——针对获取DCOM功能的测试，数据适当地被分类聚集。该视图中各栏的有关信息，参TestManager帮助索引中的*State Histogram - DCOM*。

●**Custom**——根据你的需要数据被分类聚集。有关定制一个直条图的信息，参阅122页*Configuring Custom Histograms*的内容。

下图表示了一个标准的状态直条图：



在这副直条图中，一个虚拟测试者在一个Server状态中，13个虚拟测试者在Code状态中，还有16个虚拟测试者在Overhead状态中。

细化直条图状态栏的展现（Zooming In on Histogram Bars）

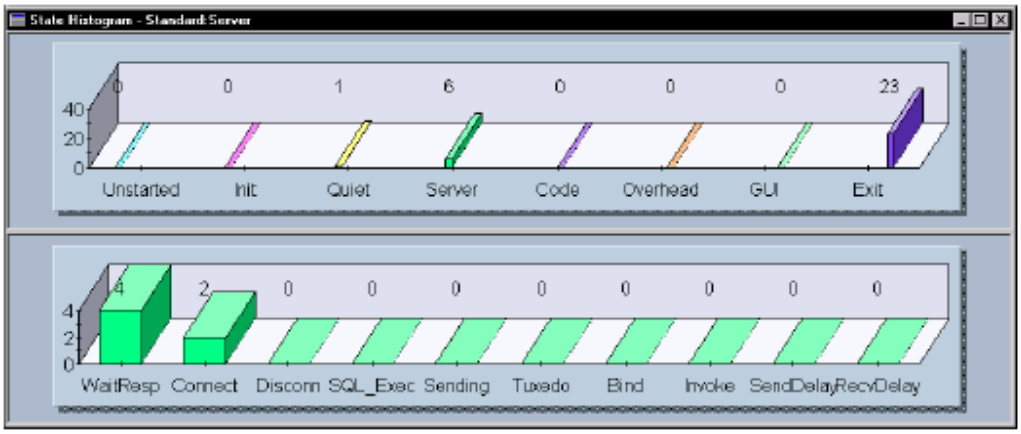
在一个直条图中的每个状态栏只显示了一个概要的状态，而这些状态又包含了独立的状态。你可以细化一个状态栏的展现以查看在每个状态中有多少虚拟测试者出现失败。

要细化一个直条图状态栏的展现：

●双击一个包含了虚拟测试者的状态栏。一个窗口出现，展现独立的状态。

要恢复该窗口到它的原状态中，点击**View > Reset**。

下图显示了一个展开的直条图，在你已经点击了Server状态栏之后：



6 个虚拟测试者作为 Server 状态被分类。展开的直条图显示有 4 个在 WaitResp 状态中，还有 2 个在 Connect 状态中。

用户和测试机视图的展现（Displaying the User and Computer Views）

用户和测试机视图动态地展现了状态和虚拟测试者操作的细节，依赖与虚拟测试者的用户类型。展现一个视图以查看单独的虚拟测试者的状态。

注意事项：TestManager展现一个基于用户或基于测试机的视图，依赖与你正在执行的suite的类型。

To display a view:

要展现一个视图：

●在一组suite执行期间，点击**Monitor > User**或**Monitor > Computer**，并选择一个视图。

User/Computer视图是：

●**User/Computer View – Full**——包含了有关所有虚拟测试者的完整信息。

●**User/Computer View – Compact**——包含了有关所有虚拟测试者的概要信息。在你执行代理测试机时，这是最有效的视图使用。

●**User/Computer View – Results**——包含了有关每个仿真命令（emulation command）的成功和失败率的信息。

●**User/Computer View – Source**——展现排列数量和被执行额源文件的名称。

●**User/Computer View – Message**——相似与User/Computer View – Compact，但也显示来自TSS展示功能的文本的最初的20条文字（letters）。

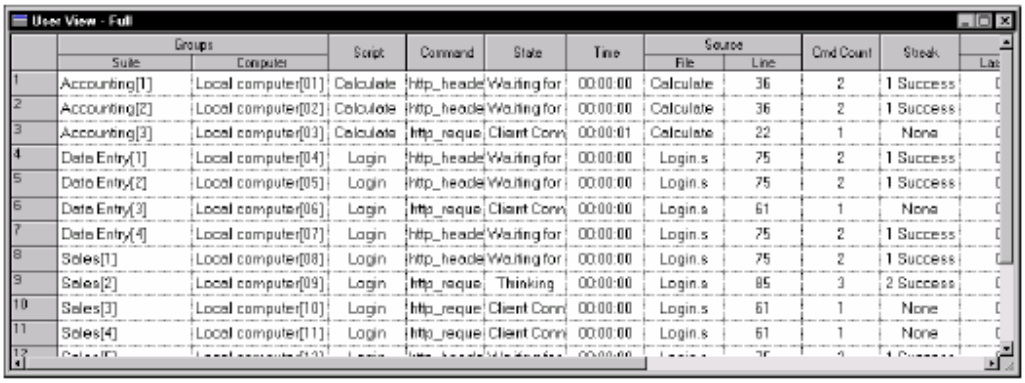
下面的条目从属于所有的用户和测试机视图：

- 要使跟踪某个虚拟测试者变得更容易，你可以变更被显示的虚拟测试者。更多信息，参阅119页*Filtering and Sorting Views*的内容。
- 当你展现一个用户或测试机视图时，你也可以虚拟测试者执行的那个测试脚本。双击第一栏中的数字，相邻与虚拟测试者。TestManager展现该测试脚本。更多信息，参阅114页*Displaying the Test Script View*的内容
- 当一个虚拟测试者不正常终止时，TestManager写一个消息，声明终止的原因到执行Suite的窗口中去。右键点击这个终止的虚拟测试者，然后选择**View Termination Message**的选项。你可以容易地确认一个反常终止的测试虚拟者，因为在视图中的**Exited**状态是红色标识的。

这一节的剩余部分则描述并给出了每种视图的例子。

User/Computer View - Full

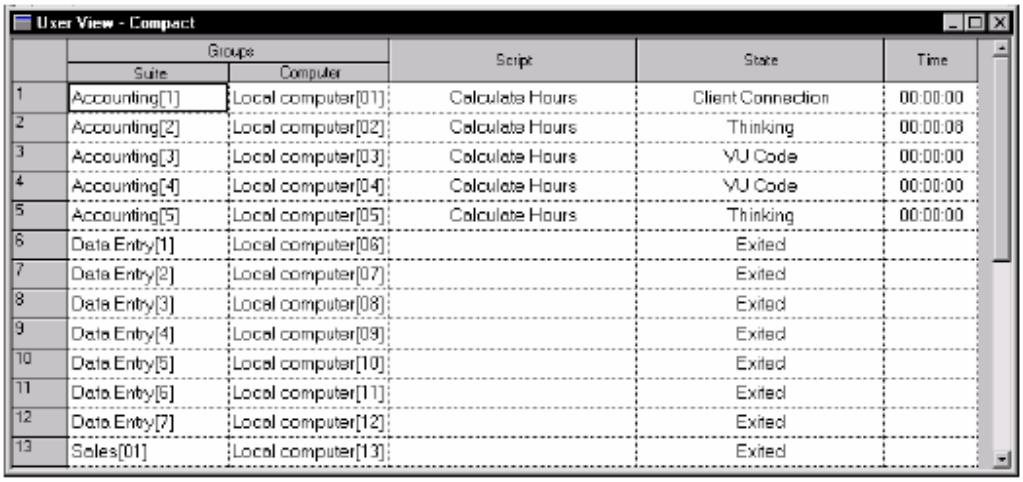
该视图包含了所有虚拟测试者的完整信息，如同下图所示：



有关该视图中各栏的信息，参阅TestManager帮助索引中的*User View - Full* or *Computer View Full*。

User/Computer View - Compact

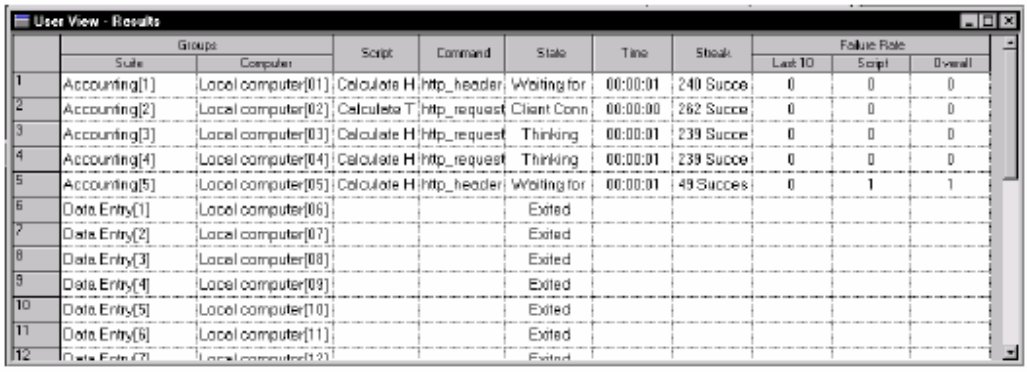
该视图包含了所有虚拟测试者的概要信息，如同下图所示：



有关各栏的信息，参阅TestManager帮助索引中的*User View - Compact* or *Computer View - Compact*。

User/Computer View - Results

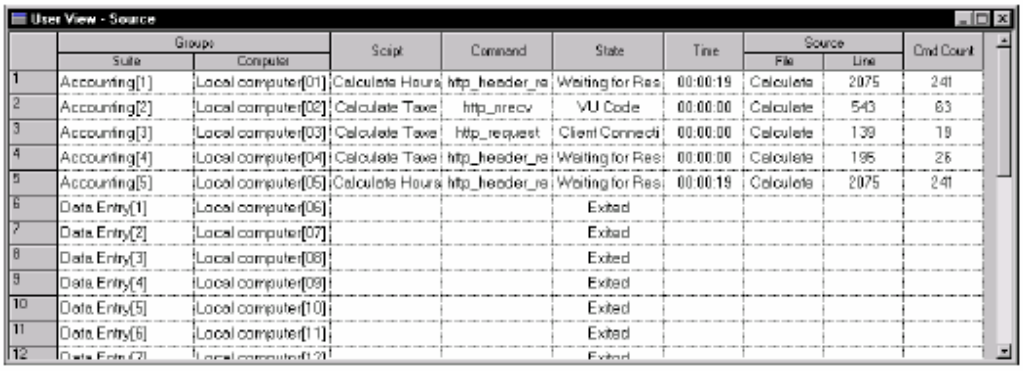
该视图包含了每个仿真命令（emulation command）的成功和失败率信息。如下图所示：



有关各栏的信息，参阅TestManager帮助索引中的*User View - Results* or *Computer View - Results*。

User/Computer View - Source

该视图展现排列数字和被执行的源文件的名称，如下图所示：

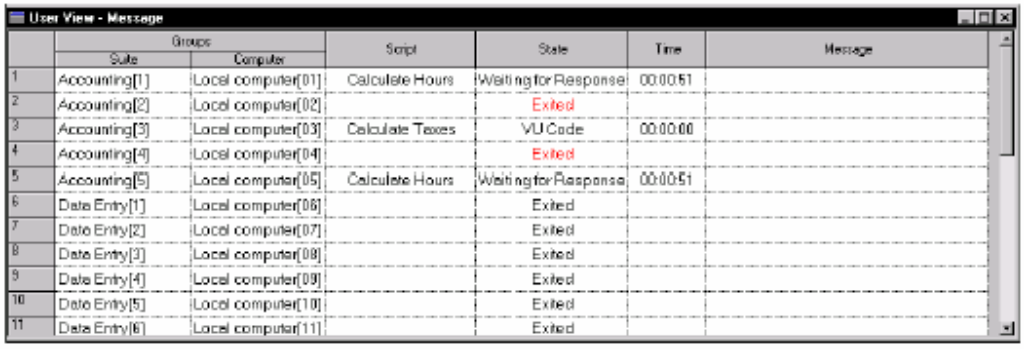


有关各栏的信息，参阅TestManager帮助索引中的*User View - Source* or *Computer View - Source*。

User/Computer View - Message

该视图类似与User/Computer Compact视图，但也显示来自TSS展示功能的文本的最初的20条文字（letters）。如果你已经添加了一个展现常式到一个测试脚本中，那么你可以显示这个视图：

下图显示了一个User View – Message的例子：



有关各栏的信息，参阅TestManager帮助索引中的*User View - Message* or *Computer View - Message*。

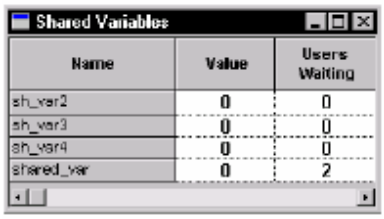
共享变量视图的展现（Displaying the Shared Variables View）

在共享变量视图中，你可以检查你设置在你的suite或测试脚本中的共享变量的值。

要展现共享变量视图：

●在一组suite执行期间，点击**Monitor > Shared Variable**。

下图显示了一个共享变量视图：



该视图显示了每个共享变量的名称，变量值，以及虚拟测试者的数字等待共享变量已达到某个值。

一个共享变量值的变更（Changing the Value of a Shared Variable）

你可以在你监控一组suite时，变更一个共享变量的值。

1 在一组suite执行期间，点击**Monitor > Shared Variable**。

2 双击变量名称，或在视图中右键点击并点击**Change Value**。

3 如果共享变量是只读的，那么可以在**Value**框中输入一个新值。

4 如果共享变量是被动态地更新，那么你不能输入一个新值。

在你读取该值的时候，确定被改变的值，并输入它，一个虚拟测试者可能已经修改该值。如果发生此情况，你的变更是失败的。改为：

在**Operators**之下，选择一个操作。如果你选择减（－）或除（/）的操作，那么针对操作的顺序是：

现存的值－新值

现存的值 / 新值

例如，假设共享变量有一个当前值6。如果你在**Value**框中输入4并点击“－”操作，则该共享变量的新值为2，因为6－4＝2。

5 点击**OK**。

Displaying the Virtual Testers Waiting on a Shared Variable

如果你的测试脚本包含共享变量，那么你可以查看虚拟测试者等待各共享变量。

要展现虚拟测试者等待一个共享变量：

1 在共享变量视图中，双击变量名，或在测试脚本中右键点击。

2 点击**See Users**。

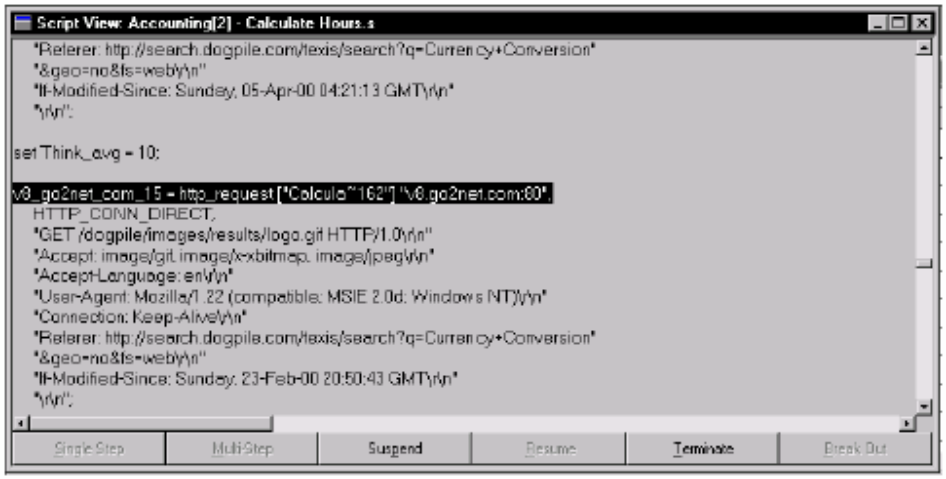
测试脚本视图的展现（Displaying the Test Script View）

测试脚本视图展现测试脚本，而这些测试脚本执行和突出了虚拟测试者执行的代码行。该视图

对于通过一个测试脚本来观察一个虚拟测试者的进展是有利的。

要展现测试脚本视图：

●在一组suite执行期间，点击**Monitor > Test Script**，并点击你要的虚拟测试者以查看它的进展



测试脚本视图窗口显示了正在执行的测试脚本。测试脚本展现的是，一行接一行的，虚拟测试者在做的是什么。

有关该视图中的可用选项，参阅TestManager帮助索引中的内容。

一个测试脚本的调试（Debugging a Test Script）

你可能在你监控一组suite时遇到一些问题。TestManager提供工具使你能够调试一个测试脚本。当你调试一个测试脚本时，执行伴随了刚好一个虚拟测试者的suite是个好方法，纠正测试脚本，然后像通常的情况那样执行该suite。

要调试一个测试脚本：

●在一组suite执行期间，点击**Monitor > Test Script**并选择虚拟测试者执行的你要调试的测试脚本。

Select one of the following the following debugging options:

选择以下的调试选项：

●**Single Step**——Steps通过一个测试脚本一次一个仿真指令，允许你去查看各指令发生了什么。要使用该选项，首先挂起一个虚拟测试者。这对于精确地找到问题是有利的。

●**Multi-step**——Steps通过一个测试脚本一次多个仿真指令。要使用该选项，首先挂起虚拟测

试者。然后你可以选择若干的指令以一次执行。

●**Suspend**——挂起一个虚拟测试者在下一个仿真指令开始时。

●**Resume**——允许一个被挂起的虚拟测试者去通过一个测试脚本以恢复它的进展。

●**Terminate**——一个测试脚本的虚拟测试者执行的结束。

●**Break Out**——将一个虚拟测试者从以下状态中移出：

□等待一个共享变量

□等待一个响应

□TSS延迟功能

同步点视图的展现（Displaying the Sync Points View）

Sync Points视图展现的是有关同步点的信息，这些同步点是你已经设置在suite中或是已经包含在一个测试脚本中。

注意事项：该视图包含从属于性能测试的信息。

要展现Sync Points视图：

●在一组suite执行期间，点击**Monitor > Sync Points**。

有关该视图中的各栏信息，参阅TestManager帮助索引中的*Sync Points view*。

Displaying Virtual Testers Waiting on a Synchronization Point

要展现等待一个同步点的虚拟测试者：

1 在一组suite执行期间，点击**Monitor > Sync Points**。

2 右键点击同步点的名称，然后点击**See Users**。

一个同步点的释放（Releasing a Synchronization Point）

你可能决定要释放一个同步点，即使还没有被达到必需的虚拟测试者的数量。后继的达到同步点的虚拟测试者不被拥有。然而，如果你已经在suite中设置一个重起时间和一个最大时间，那么虚拟测试者将被延迟。例如，如果你释放一个同步点，但是你已经设置了一个1秒的重起时间和一个4秒的最大时间，那么每个要达到同步点的虚拟测试者会有1到4秒的时间延迟。

要释放一个同步点：

1 在一组 suite 执行期间，点击 **Monitor > Sync Points**。

2 右键点击同步点的名称，然后点击 **Release**。

3 一个确认消息出现，询问你是否确认该释放。点击 **Yes** 或 **No**。

测试机是视图的展现（Displaying the Computer View）

使用Computer视图去检查测试机资源在一组suite执行期间的使用情况，以及本地和代理测试机在一次执行的开始和结束时的状态。

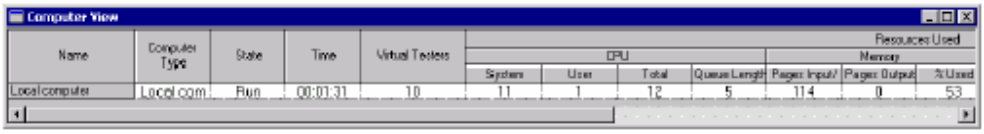
有关该视图中各栏的信息，参阅TestManager帮助索引中的*Computer view*。

Viewing Resource Usage During a Run

当你在一组suite执行期间查看资源的利用时，TestManager展现对于运行的各本地和代理测试机的资源使用情况。

要检查测试机资源在一个执行期间的使用情况：

●在一组suite执行期间，点击**Monitor > Computers**。

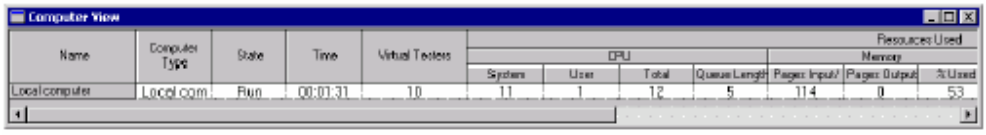


执行期间资源使用情况的图表表示（Graphing Resource Usage During a Run）

你可以图表表示你的测试机在一组suite执行期间的资源使用情况。这样就提供给你资源使用情况的一个可视化的描述。在这个资源图表中，你可以改变图表中一个条目的颜色，从图表中删除一个条目，或删除图表中所有的条目。

要图表表示测试机资源：

●在一组suite执行期间，点击**Monitor > Computers**。



在一次执行的开始和结束时查看测试机（Viewing Computers at the Start or End of a Run）

Computer视图在代理测试机启动时自动出现。当所有的代理启动和执行，Computer视图关闭。当代理开始关闭，Computer视图自动地重新出现，以使你可以观察清除（cleanup）活动，诸如

转移文件到本地测试机。

Computer视图包括了**Progress**消息，它指明了什么时候测试机在创建或初始化过程，转移文件，终止虚拟测试者，和其他等等。

有关该视图中的各栏信息，参阅 TestManager 帮助索引中的 *Computer view*。

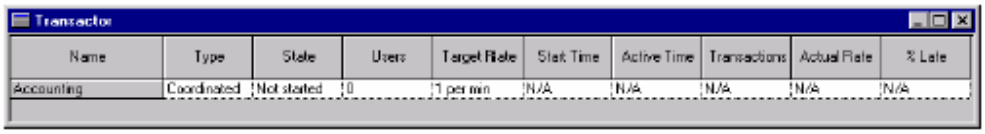
处理者视图的展现（Displaying the Transactor View）

Transactor视图显示你插入到suite中的任意处理者的状态。

注意事项：该视图主要率属于性能测试。

展现一个Transactor视图：

●在一组suite执行期间，点击**Monitor > Transactors**。



有关各栏的信息，参阅TestManager帮助索引的*Transactor view*。

组视图的展现（Displaying the Group Views）

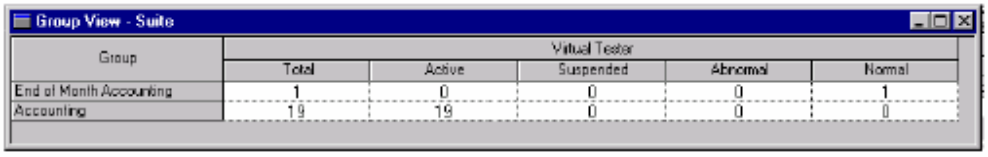
Group视图显示你在suite中定义全部的用户组的状态。Group视图又有Suite视图和Computer视图之分，两者显示相同的信息，但Suite视图根据用户组显示信息，而Computer视图则根据测试机显示信息。

要展现一个Group视图：

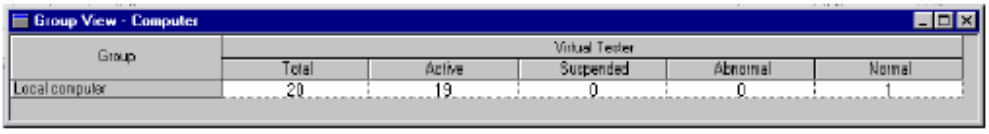
●在一组suite执行期间，点击**Monitor > Groups**。

那两种Group视图是：

●**Group View – Suite**——在一组suite中的一个用户组列表。下图显示了该视图：



●**Group View – Computer**——指派到相同测试机的一个组列表。下图显示了该视图：



各栏的有关信息，参阅TestManager帮助索引中的*Group View - Suite* or *Group View - Computer*。

要展现组内的虚拟测试者，右键点击在左边栏中的名称，然后点击**See Users**。

过滤和分类视图（Filtering and Sorting Views）

这一节讨论如何定制一个视图。例如，你可以用不同的方法分类虚拟测试者，或者你可以过滤虚拟测试者和组，以便于唯一的确定信息的展现。

在用户或测试者视图中分类虚拟测试者（Sorting the Virtual Testers in a User or Computer View）

当你展现一个User 或Computer视图时，你可能要以一个特定的顺序查看虚拟测试者。例如，你可以按字母顺序分类虚拟测试者，或者你可以它们开始的顺序对其进行分类。

要改变虚拟测试者的展现顺序：

●从一个打开的用户或测试机视图中，在**Suite**或**Computer**标题下的一栏中右键点击，可看到一个快捷菜单。

你可以使用以下的顺序对虚拟测试者进行分类：

●**Suite顺序——**在suite中用户组出现的顺序。

●**执行顺序——**虚拟测试者开始的顺序。

●**Suite组——**Suite组的字母顺序列表。

●**测试机组——**测试机组的字母顺序列表。

过滤一个视图（Filtering a View）

你可以在一个用户或测试机视图中过滤虚拟测试者，以使唯一的确定的虚拟测试者出现。如果你的suite包含许多的虚拟测试者并且你的重点在于其中一些虚拟测试者的进展，那么这样做是有效的。

过滤虚拟测试者（Filtering Virtual Testers）

你可以在虚拟测试者身上通过包含或者排除被选择的虚拟测试者来进行过滤：

●**Include（包含）**——展现只是你选择的虚拟测试者。

●**Exclude（排除）**——展现除了你选择之外的所有虚拟测试者。

要过滤虚拟测试者：

1 从一个打开的用户或测试机视图中选择你要过滤的虚拟测试者并右键点击以显示快捷菜单。

2 点击**Filter Virtual Testers**，然后点击**Include**或**Exclude**。

根据值过滤一个虚拟测试者（Filtering a Virtual Tester by Value）

你可以在执行期间保持常量的任意值上过滤虚拟测试者，诸如它的组名，执行的脚本的类型，或是一个虚拟测试者执行的测试机名称。

例如，你可能正在执行一个在Accounting用户组中具有200个虚拟测试者的测试，Data Entry用户组中具有300个虚拟测试者，Sales用户组中具有500个虚拟测试者。你只想要查看Data Entry组中的虚拟测试者。过滤该组以使TestManager只是展现具有“Data Entry”值的组。

要通过值来过滤一个虚拟测试者：

1 右键点击一个打开的Use或Computer视图中在suite、Group、或者Type标题下的任意单元。

2 点击 **Filter Virtual Testers**，然后点击 **By Value**。

过滤一个组视图（Filtering a Group View）

如果Group视图展现了许多栏目，你可以过滤出一些栏目以提供更多的空间去查看你想要查看的栏目。你可以在执行期间保持常量的任意值上过滤一个组，诸如测试机或用户组名称，或测试脚本类型。

要过滤一个Group视图：

1 在一个打开的User或Computer视图的Group或Type标题中使用右键点击。

2 点击**Filter By Value**。

恢复默认视图（Restoring the Default Views）

如果你细化展现一个直条图栏，过滤一个视图，或改变视图中一栏的宽度，你可能想要恢复栏目或者视图到最初的设置。

要恢复一个视图到最初的设置：

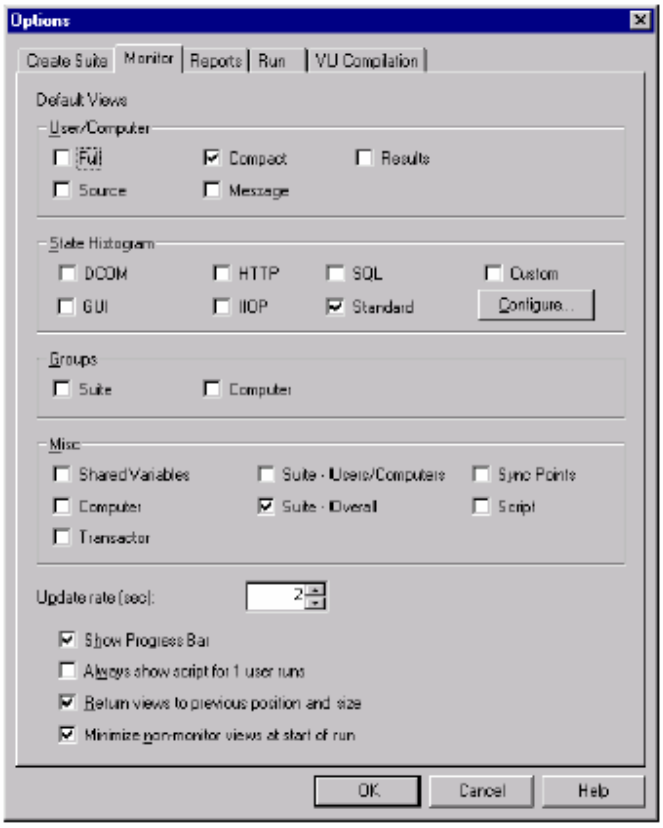
●从你想要恢复的视图中，点击**View > Reset**。

变更监控的默认（Changing Monitor Defaults）

在你监控一组suite时，你可设置视图为自动显现，视图的刷新频率，以及在你执行一组suite时工具栏是否自动显现。你也可以配置Custom直条图，并改变它的颜色，这将在下一节描述。

要变更监控的默认设置：

●点击**Tools > Options**，并点击**Monitor**标签。

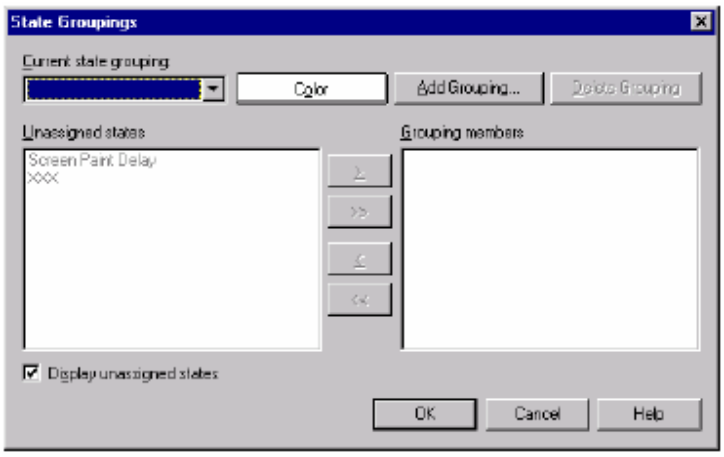


配置常规直条图（Configuring Custom Histograms）

通过默认，State Histogram－Custom与State Histogram –Standard是相同的。然而，不像其他的直条图，你可以配置State Histogram－Custom。你可以配置组，创建新组，并变更以标识组的颜色。

要配置the State Histogram – Custom：

●从Options对话框**Monitor**标签中，在**State Histogram**下，点击**Configure**。



要在一个常规直条图组中指派或移除一个状态：

●在State Groupings对话框中，从**Current State Grouping**区域选择一个分组。

要添加一个完整的分组到常规直条图中：

●在State Groupings对话框中，点击**Add Grouping**。

要从常规直条图中删除一个组：

●在State Groupings对话框中，从**Current State Grouping**框中选择你要删除的组，然后点击**Delete Grouping**。

When you delete a group, all states that were in the group are unassigned.

当你删除一个分组时，在该分组内的所有的状态是被解除指定的。

在执行期间对 suite 进行控制（Controlling the Suite During a Run）

TestManager提供多种方法去帮助你在suite执行时来控制它。

例如，你可以挂起一组suite去变更设置或检查它的进展。

在一组 suite 中挂起和恢复虚拟测试者（Suspending and Resuming Virtual Testers in a Suite）

当一组suite执行时，你可以挂起和恢复所有的虚拟测试者，或者你可以挂起和恢复单独的虚拟测试者。这对于审查在执行期间发出现的问题是有利的。

要挂起或恢复所有的虚拟测试者：

●点击**Monitor > Suspend**或**Monitor > Resume**。

要挂起或恢复单独的虚拟测试者：

●点击**Monitor > Users**或**Monitor** > **Computers**，并选择你想要挂起的虚拟测试者列。

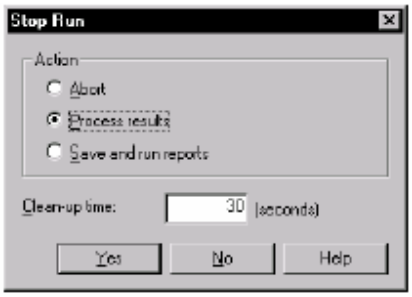
停止 suites（Stopping Suites）

当TestManager检查suite时，取消suite的执行。

●从执行suite的窗口的Messages中，点击**Cancel**。

要停止一组suite在它执行时：

●点击**Monitor > Stop**。



停止一组suite的方法：

●**Abort**——停止执行并且没有保存结果。如果你不打算执行任意的报告或查看任意的虚拟测试者错误或虚拟测试者输出文件，那么点击该选项。

●**Process Results**——停止执行但保存结果以使你可以执行报告，并在Test Log窗口中查看虚拟测试者错误或虚拟测试者的输出文件。

●**Save and Run Reports**——停止执行，保存结果，并产生报告，如果你的执行正常地执行。你也可以指定**Clean-up time**。这是时间的数量允许你从时间起请求终止，直到TestManager强行终止该执行。

注意事项：在你异常终止一个包含了多处理器的本地机或代理机的大suite时，选择60秒或更多的**Clean-up time**以允许虚拟测试者（rtsvui过程）时间，以此使其到时间后自动退出。默认为1秒的**Clean-up time**经常引起本地测试机一次终止许多的处理过程，并且导致剩余物rtsvui过程。虽然这是无害的，但他们会混乱过程表。通过使用Task Manager，它们可被破坏，或一次全部退出。

当suite完成执行－－是否正常地或通过手工终止－－TestManager在Test Log窗口中展现任意的日志数据。有关Test Log窗口的信息，参阅127页*Evaluating Tests*的内容。

测试的评估（Evaluating Tests） 6

本章说明如何使用TestManager中的Test Log窗口去查看日志并说明它们的内容。它也说明如何创建并执行报告来帮助你管理你的测试成果。本章包含下面的标题内容：

●关于测试日志

●查看测试日志结果

●由Rational Robot查看测试脚本结果记录

●结果的报告

注意事项：有关过程的细节，参阅TestManager的帮助。

关于测试日志（About Test Logs）

在你执行一组suite，测试用例，或者测试脚本之后，TestManager写结果到一个测试日志中。你使用Rational TestManager的TestLog窗口查看在你执行了一组suite，测试用例，或测试脚本之后而被创建的测试日志。

一个测试周期可以具有很多单个的针对一个应有程序的特定区域的测试。

复查在TestLog窗口中的测试结果显示是否各测试通过或失败。复查和分析帮助决定你是在软件开发过程的那个阶段，以及是否一个失败是一个缺陷或者一个设计更改。

你可以使用TestLog窗口来：

●打开一个测试日志去查看一个结果。

●过滤一个测试日志的数据去查看仅仅是你需要的信息。

●由在Test Log窗口的Test Case Results标签中的一个未评估的结果来查看所有的测试用例。在性能测试用例施行的评估结果中，这有着显著的作用。你可以通过实际的结果分类测试用

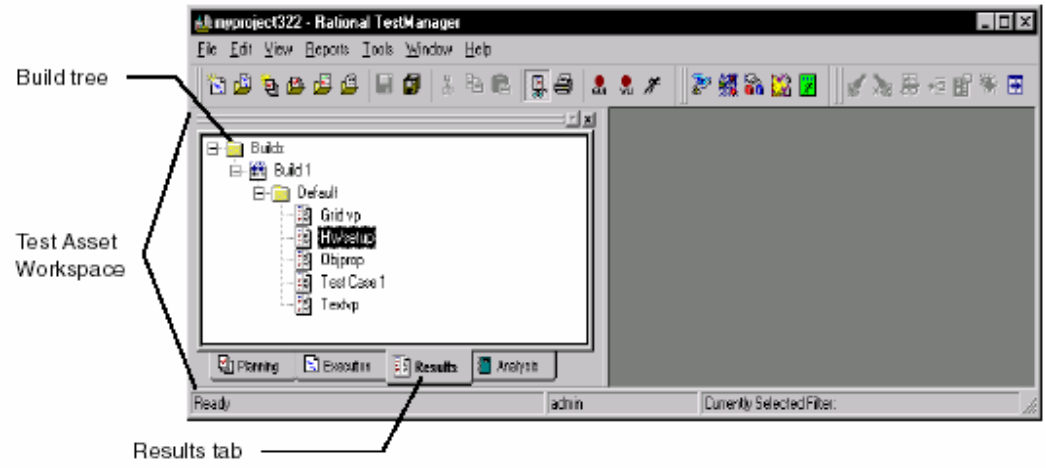
例，并复查和更新所有未评估的测试用例。

- 提交针对一个失败的日志事件的一个缺陷。测试日志自动地填写build，配置，和在Rational ClearQuest缺陷表中的测试脚本信息。有关缺陷提交的信息，参阅138页*Submitting and Modifying Defects*的内容。
- 利用合适的测试脚本开发工具打开一个script-based日志事件的测试脚本。例如，如果你创建一个手工测试脚本，那么Rational ManualTest打开和展现该测试脚本。如果你创建的是一个常规测试脚本类型，那么TestManager打开带有你指定的编辑器的测试脚本。你使用一个Test Script Console Adapter（测试脚本控制台适配器）去指定编辑器打开一个测试脚本和去处理常规测试脚本类型。有关使用一个常规测试脚本类型的信息，参阅13页*Defining Custom Test Script Types*的内容。
- 预览或打印在Test Log窗口的活动测试日志中被展现的数据。
- 如果你使用Rational Robot去记录测试脚本，那么你可以分析在Comparator中的结果来决定一个测试失败的原因。如果你使用Rational Quality Architect从Rose模型中生成测试脚本，那么你使用Grid Comparator来分析结果。有关使用Comparators的信息，参阅181页*About the Four Comparators*的内容。

在 TestManager 中打开一个 Test Log（Opening a Test Log in TestManager）

要手动打开TestLog窗口：

- 点击**File > Open Test Log**。
- 在Test Asset Workspace（测试资产工作区）的**Results**标签中，展开Build树并选择一个日志。



你可以打开多于一个的在TestLog窗口中的测试日志。如果你已经打开了多个测试日志，那么这个当前活动的日志是按照你使用最多的菜单指令的一个。(If you have more than one test log open, the log that is currently active is the one that is acted upon when you use most menu commands.)

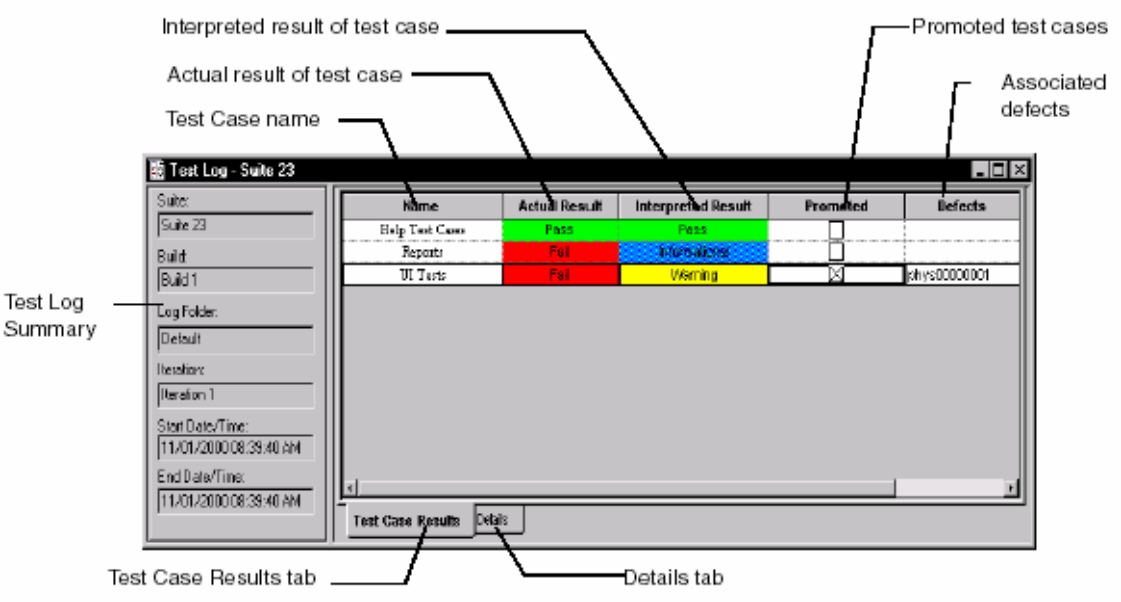
执行一个测试用例，测试脚本，或suite后，打开测试日志进行控制：

● 点击**Tools > Options**并点击**Run**标签。

注意事项: 你也可以从一个RationalTestFactory的application map中的一个被选择的测试脚本开始你的测试日志。有关更多的信息，参阅手册*Using Rational TestFactory*或者Rational TestFactory的帮助。

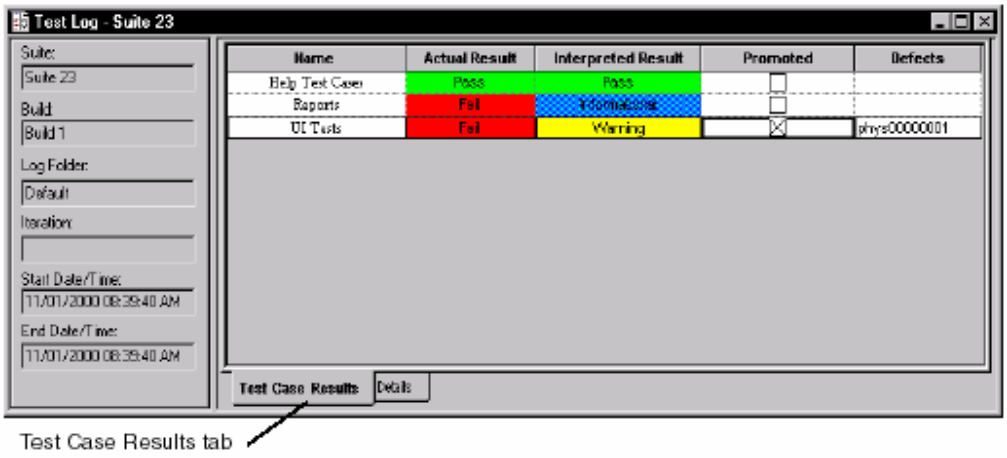
Test Log 的窗口（The Test Log Window）

TestManager中的TestLog窗口包含了测试日志摘要(Test Log Summary)区域，测试用例结果(Test Case Results）标签，和细节（Details）标签。出现在Test Log Summary区域中的迭代是与build相关联的。



测试用例结果标签（Test Case Results Tab）

从Test Log窗口中，你可以点击Test Case Results标签来获得每个测试用例总的结果——是通过还是失败？Test Case Results标签展现一个测试用例的执行结果或者包含了测试用例的一组suite。如果你从TestManager，ManualTest，或者从Robot中执行一个测试脚本，即使测试脚本是一个测试用例的实施，那这个Test Case Results会是空的。你必须执行一个测试用例去获得在Test Case Results标签中的结果。



说明测试用例结果（Interpreting Test Case Results）

在你首次打开一个测试日志并点击**Test Case Results**标签，它针对实际的结果和结果的说明展

现相同的值：pass, fail, informational, 或者warning。

一个结果是测试用例结果在它执行时返回和记录日志的。

该实际结果也出现在说明结果的栏中（Interpreted Result）作为默认的被说明结果。你可能想要*interpret*（说明解释）一个测试用例结果，如果你持有关于它的附加的了解，并且想要纠正它。例如，一个测试用例可能失败，因为这里有一个软件缺陷。在这种情景下，失败是有效的，并且你不需要说明该结果。但是，一个测试用例可能在其他的情况下失败，包含：

●Application-under-test已经变更，并且你意识到你需要去修改相关的测试脚本。

●有一个测试自动化的问题－－例如，一个测试脚本以一个错误的顺序执行。

在其他情况中，失败是误导的，并且你可能想要改变在Test Case Results标签中的Interpreted Result栏的Pass或者Informational。

Promoting Test Case Results

在你*promote*一个测试用例结果时，你指定该结果对你的项目是有用的并使它对其他人是可见的。在你promote一个结果之前，它仅仅出现在测试日志中。

例如，假定需要的测试用例的数量通过你的针对一个build被运送到beta用户的测试标准。你把测试结果作为“正式的”（official）并要将它们包含在Test Case Results Distribution或Test Case Trend报告中。因此，你promote针对每个测试用例的结果对于你的测试标准是重要的。

另一方面，假定TestManager报告一个测试用例是失败的。你发现测试机未插入，因而这个测试用例不能执行－－这就是为什么TestManager报告为失败。一个先前的测试用例在那台测试机上通过，所以你promote这个测试用例，因为你想要将它包含在一个Test Case Results Distribution或Test Case Trend报告中。因此，你不能promote已失败的测试用例的结果，因为这个结果是没有意义的－－并且，实际上，是误导的。

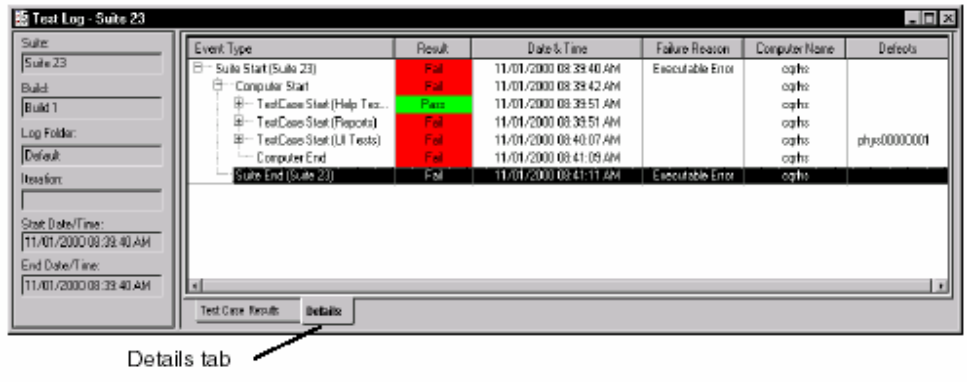
Promotion不影响结果（pass, fail, informational, or warning）。它仅仅指明了这个结果是有意义的，以使结果足以出现在一个Test Case Results Distribution或者Test Case Trend报告中。

注意事项：你也必须在你关闭一个测试日志时保存测试用例结果以使它出现在一个Test Case Results Distribution报告中，当然是在*promoting*它之后。

细节标签（Details Tab）

Test Log 窗口中的**Details**标签包含日志事件，事件是在你执行一个测试脚本，测试用例或者suite时被生成的。

注意事项：在Details标签中的详尽的测试结果是不能被promoted的。



查看测试日志结果（Viewing Test Log Results）

在执行一个测试用例，测试脚本，或者suite之后，你可以在Test Log窗口中快速的评估结果。

查看测试用例结果（Viewing Test Case Results）

全部测试用例的结果出现在Test Log窗口的**Test Case Results**标签中。

在**Test Case Results**标签中你可以：

●分类，可通过名称（name），实际结果（actual result），说明结果（interpreted result），或者promotion状态（promotion status）。

要分类测试用例：

☐ 在**Test Case Results**标签中，点击**View > Sort By**，然后选择你想要如何分类测试用例。

☐ 双击栏头。

●通过下面的标准显示测试用例：

☐ 实际结果，具有pass，fail，warning，或者其他。

☐ 说明，关于pass，fail，warning，或者其他。

☐ 隐藏相等的结果。

要通过某种标准显示测试用例：

☐ 在**Test Case Results**标签中点击**View > Show Test Cases**。然后选择你要查看的标准。

●展现事件细节对于一个在Test Case Results标签中的特殊的测试用例。

要展现事件细节：

☐ 在**Test Case Results**标签中，选择一个测试用例，然后点击**View > Event Details**，

TestManager展现**Details**标签并针对你选择的测试用例设置事件细节。

查看事件细节（Viewing Events Details）

每个事件的详细信息在Test Log窗口的**Details**标签中是可用的。

在**Details**标签中，你可以:

●崩溃（Collapse）和展开事件。

●找出一个特殊的结果，事件类型，协议，失败原因，验证点，或者命令，或者寻找一个特定事件属性的名称和值。

你可以查看一个事件并从Log Event窗口导出所有的失败（**Details**标签的**Result**栏中用红色显示）。

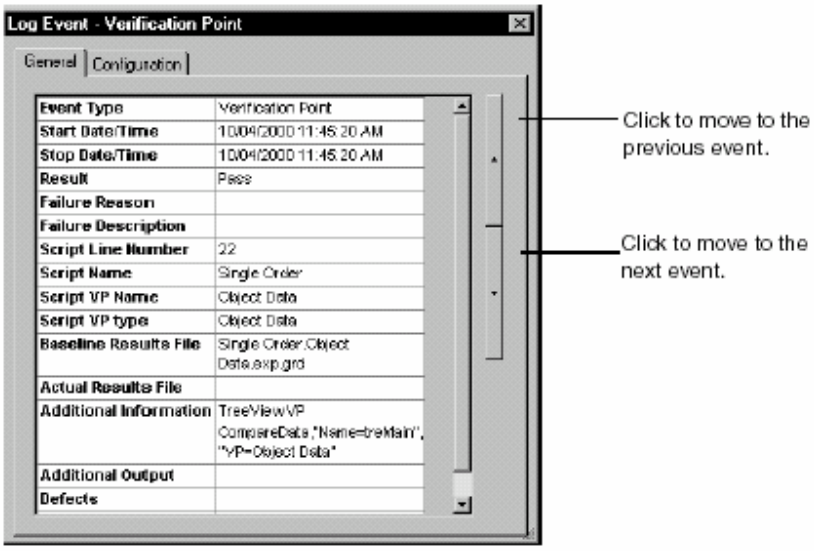
要查看一个特殊的事件：

1 点击Test Log窗口中的**Details**标签。

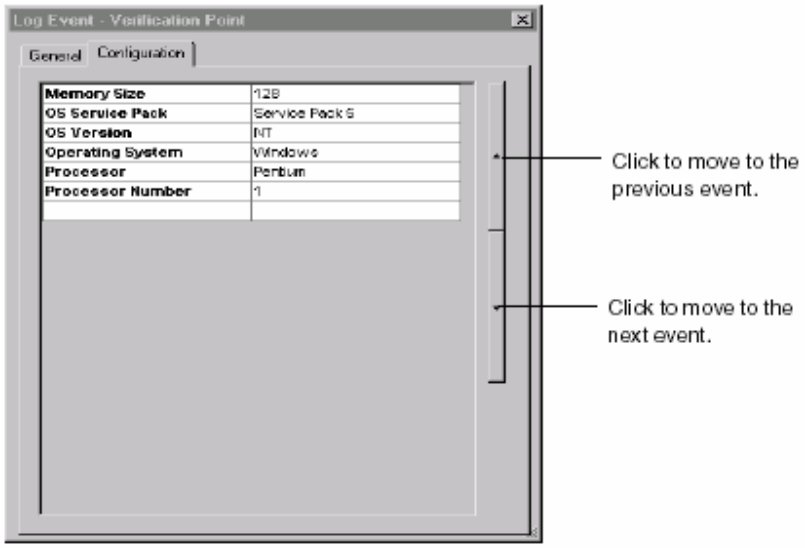
2 点击**View > Properties**。

你可以在你移动Test Log窗口**Details**标签中的每个事件时保持Log Event窗口的打开状态。你也可以缩放和移动窗口。

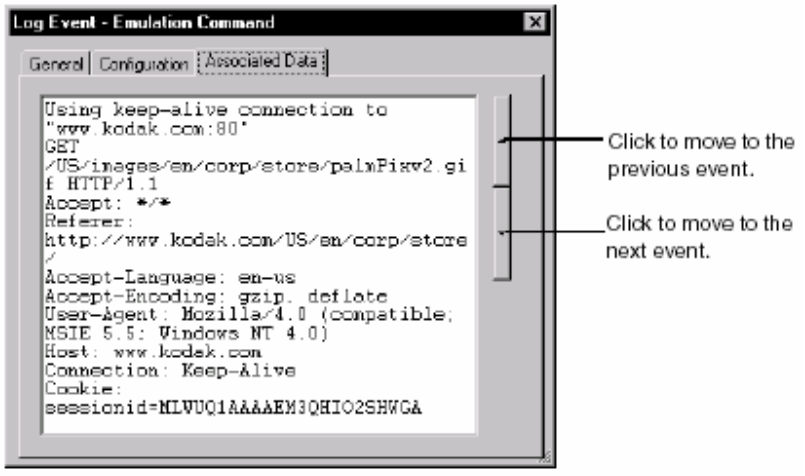
注意事项：如果你先前使用的是Rational Suite PerformanceStudio，那么在跟踪（Trace）和模拟（Analog）报告中发现的信息对于现在的Log Event窗口是可用的。Log Event窗口中的**General**标签展现事件类型，日期和时间事件被记录，测试脚本名称，结果信息（如果任何的），和有关一个日志事件的其他信息。



Log Event窗口的**Configuration**标签展现你执行测试脚本的测试机的相关配置信息。



当你添加一个新的日志事件属性类型时，一个新的标签出现在Log Event窗口中。数据，依赖与如何被定义，展现在Log Event窗口的一个单独的标签中作为文本或一个HTML文件，或者在一个单独的应用中。例如，**Associated Data**标签展现由所有的http_request命令产生的数据，TSS登陆方法（例如VB中的`TSSLog.Message`），以及SQABasic定时声明。



有关日志过滤器（About Log Filters）

你可以创建一个日志过滤器来缩小在Test Log活动窗口中展现的数据数量。例如，你可以创建一个过滤器只展现查证点或测试机启动。一个日志过滤器可以使它容易的查看大的测试日志。

注意事项：TestManager保存日志过滤器在一个项目中。所有的日志过滤器对项目中的所有用户都是可用的，但是，当你对一个活动日志实施一个日志过滤器时，该结果只能在你的本地系统上可见。

你可以：

●创建或者编辑日志过滤器。

●选择一个过滤器来缩小在Test Log活动窗口中展现的数据数量。

●复制，重命名，或者删除一个日志过滤器。复制的特性在你想要创建彼此类似的复合过滤器（时是有用的。在你创建第一个过滤器之后，你可以复制它。然后，你可以编辑这个被复制了的过滤器来使其做必要的修改。你也可以重命名和删除一个日志过滤器。

创建和编辑一个日志编辑器（Creating and Editing a Log Filter）

要创建或者编辑一个日志过滤器：

●点击**Tools > Manage > Log Filters**。点击**New**，或者选择一个过滤器并点击**Edit**。然后在每个标签上选择你要过滤的事件信息的类型。

如果你过滤一个事件类型，那么这个过滤器包含所有的事件类型信息以及事件类型本身。

实施一个测试日志过滤器（Applying a Test Log Filter）

在你创建一个测试日志过滤器后，你需要设置过滤器，使用它缩小在Test Log活动窗口中展现的数据数量。

要实施一个测试日志过滤器：

●打开一个日志，点击**View > Set Test Log Filter**，并选择这个过滤器。

要关掉所有的日志过滤器，点击测试日志过滤器**All**。这个过滤器展现你执行一个测试用例，测试脚本，或者suite时的所有（记录）登陆在Test Log窗口中的信息。

注意事项：在你实施一个测试日志过滤器后，活动的Test Log窗口生成一个新的日志并到那时展现新的被过滤的日志信息，可能会花一些时间。

查看一个测试脚本（Viewing a Test Script）

你可以选择任意的关联与一个测试脚步的日志事件并在你使用的创建测试脚本的工具中查看它。例如，如果你创建一个GUI测试脚本并从Test Log窗口打开该测试脚本，那么这个测试脚本在Robot中打开。如果你创建一个手工的测试脚本并从Test Log窗口代开该测试脚本，那么它在Rational ManualTest中打开。如果你创建一个常规测试脚本类型，TestManager根据你指定的编辑器打开该测试脚本。你使用一个Test Script Console Adapter（测试脚本控制台适配器）来指定打开一个测试脚本的编辑器并去处理常规测试脚本类型。有关使用一个常规测试脚本类型的信息，参阅13页*Defining Custom Test Script Types*的内容。

注意事项：在你双击在Rational Purify, Quantify, 或者PureCoverage之下由Robot生成的一个打开的日志中的事件时，测试脚本在Robot中打开，并且，在诊断工具中打开文件。有关设置诊断工具选项的信息，参阅Robot的帮助。

要查看一个测试脚本：

1 打开一个测试日志。

2 点击**Details**标签。

3 右键点击一个测试脚本的开始或者结束事件，并点击**Open Script**。

与测试日志一起工作（Working with Test Logs）

在你与测试日志一起工作时，你可以：

●打开一个测试日志。

●重命名一个测试日志。

●查看一个测试日志的属性（名称，描述，build，和日志文件夹）。

要做这些工作的任意一个：

1 在Test Asset Workspace（测试资产工作区）中，点击**Results**标签。

2 右键点击一个测试日志，然后选择一个在快捷菜单中的条目。

注意事项：你也可以打印在Test Log窗口中展现的数据。有关信息，参阅141页*Printing a Test Log*的内容。

有关测试日志（About Test Logs）

测试日志记录一个测试脚本、测试用例或者suite执行期间发生的任何事，从它们的开始到结束。除非日志被特别地关闭，每个虚拟测试者的活动，系统调用，查证点，和结果都是包含在日志过程中的。你可以从Test Log窗口中查看每个事件的属性，并且你也可以在一个完整的事件中查看某个测试日志。

Suite 日志（Suite Log）

Suite日志包含所有关联与一组suite执行的信息。 这是当你执行一组suite时与你在Messages窗口中看到的信息是相同的。该日志包含了build，日志文件夹和日志名称信息，和有关suite检查的信息，编译测试脚本，以及任何关联与该suite的警告或错误。

要查看一个suite日志：

●右键点击在TestLog窗口中的一组suite的开始事件，并点击View Suite Log。

要打印一个suite日志：

●从一个打开哦suite日志中，点击File > Print。

虚拟测试者错误文件（Virtual Tester Error File）

错误文件包含关联与一个指定的虚拟测试者的任意执行期间的错误信息。

注意事项：错误文件不能一直存在。如果一个错误是在录制回放期间遇到的，那么TestManager会在这个文件中记录该错误。如果没有错误出现，那么在这个文件中没有数据，并且，TestManager自动地删除该文件。有关信息，参阅*Using Rational Robot*手册，或者Rational Test API文档适合于的脚本语言。

要查看一个错误文件：

●右键点击在TestLog窗口中一个虚拟测试者的开始事件，并点击**View Virtual Tester Error File**。

要打印一个错误文件：

●从一个打开的错误文件，点击**File > Print**。

虚拟测试者的输出文件（Virtual Tester Output File）

虚拟测试者的输出文件包含一个虚拟测试者特别地从测试脚本输出写入的任意信息。这可能有关与任何事件。例如，这个文件可以记载（log）SQL命令。

注意事项：输出文件不能一直存在。如果输出是在录制回放期间生成的，那么TestManager在这个文件中记录该输出。如果没有输出被生成，那么在这个文件中没有数据，并且，TestManager自动地删除该文件。有关信息，参阅*Using Rational Robot*手册，或者Rational Test API文档适合于的脚本语言。

查看一个输出文件：

●右键点击在Test Log窗口中一个虚拟测试者的开始事件，然后点击**View Virtual Tester Output File**。

要打印一个输出文件：

●从一个打开的输出文件中，点击**File > Print**。

有关缺陷的提交和修改（About Submitting and Modifying Defects）

一个缺陷可能是从一个请求来的任何事，针对一个新的特征，在测试之下的应用中向着一个实际bug的发现。缺陷跟踪是软件测试工作的重要部分。

你可以使用TestManager的Test Log窗口来提交针对在一个被记录的测试脚本录制回放期间失败的任意查证点的缺陷。当你从测试日志中提交一个缺陷时，TestManager打开一个特殊的缺陷表，TestStudio缺陷表，并由来自日志的信息填充该表。如果你将一个测试脚本关联与一个测试输入，那么该测试输入信息自动地显示在该缺陷表中。（当你提交缺陷时，TestManager不是实际地启动ClearQuest；它打开缺陷表，这是ClearQuest的一部分。）你也可以使用ClearQuest来手动地提交缺陷，但是没有区域将被自动地填充。

注意事项：要使用ClearQuest来保存缺陷，一个管理员必须首先设置ClearQuest图表，然后再创建或者附上一个ClearQuest用户数据库作为一个Rational项目的部分。有关信息，参阅*Administering Rational ClearQuest*手册。如果你使用ClearQuest Multisite，你必须具有ClearQuest Multisite权限以提交或者修改在Test Log窗口中的一个缺陷表。如果你不具有ClearQuest

Multisite，你可以使ClearQuest手工地修改缺陷。

为了你的便利，一个专门地设计图表，TestStudio图表，是为你的缺陷跟踪而包含在你的软件中的。在ClearQuest中，术语*schema*涉及到关联与一个变化-请求（change-request）数据库的全部属性。这包含区域定义，区域行为，状态转变表，活动，和表格。有关ClearQuest图表的更多信息，参阅Rational ClearQuest的帮助。

关于Rational TestStudio 图表（About the Rational TestStudio Schema）

TestStudio图表包含两个TestStudio缺陷表格：一个是为了提交新的缺陷，一个是为了修改和跟踪缺陷信息。

注意事项：要使用TestStudio图表，你必须在你创建一个ClearQuest用户数据库作为一个Rational项目的部分时选择它。你也可以选择对一个项目的一个现存的ClearQuest用户数据库，并使用TestStudio图表。如果你想要附上一个现存的ClearQuest数据库，它与TestStudio图表相比有不同的图表，你需要升级该图表。有关信息，参阅*Using the Rational Administrator*手册。

你可以使用TestStudio缺陷表来跟踪许多的或者少数的有关于一个你要的缺陷的细节信息。

注意事项：要展现缺陷表中各个条目的信息，右键点击条目并点击帮助。

如何提交和修改一个缺陷（How to Submit and Modify a Defect）

你可以从Test Log窗口或ClearQuest中提交或者修改缺陷表。如果你打开TestManager中的一个测试日志，TestManager缺陷表中的多数区域。如果你使用ClearQuest，那么你必须手动输入。

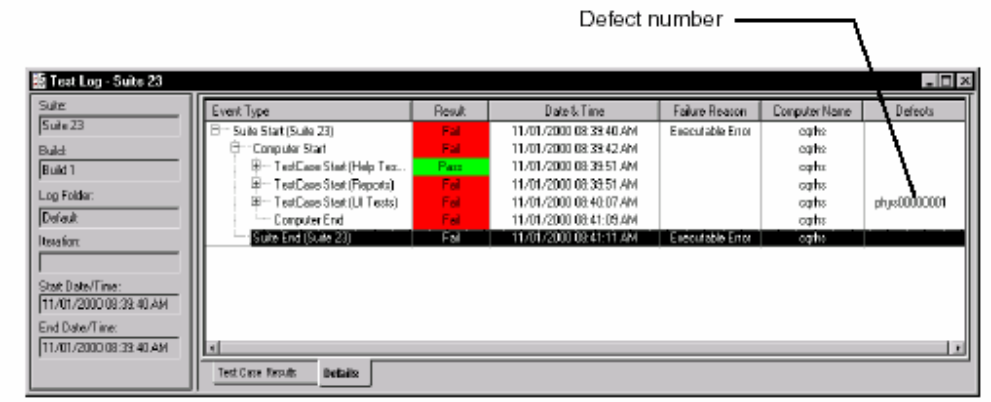
要从TestManager中提交或者修改一个缺陷：

●右键点击**Event Type**栏中失败的事件，点击**Submit Defect**。

●点击**Edit > Submit Defect**。

注意事项：TestManager视图去链接ClearQuest使用你的用户名和密码。然而，如果TestManager依然不能够链接到ClearQuest数据库，那么Login对话框会出现。在这个用例中，输入你的ClearQuest用户名和密码。选择你想要提交缺陷的数据库。

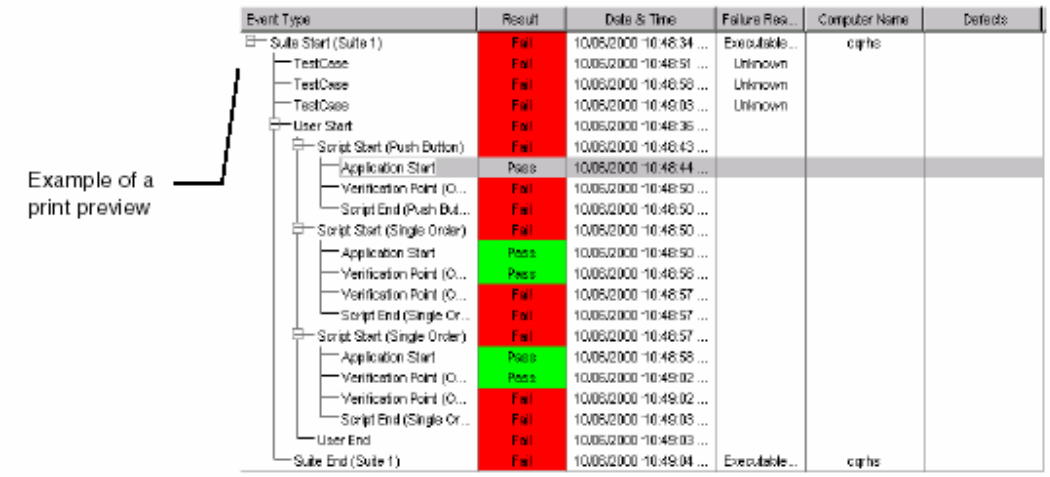
如果你从测试日志提交一个缺陷，那么新的缺陷的数量会显示在测试日志的**Defect**栏中。



注意事项：你也可以由SiteCheck提交缺陷，在你录制回放了一个Web查证点之后。从测试日志中，右键点击失败的Web查证点，并点击**Submit Defect**。在SiteCheck中，点击**Tools > Enter a Defect**。

打印一个测试日志（Printing a Test Log）

你可以预览或者打印在活动测试日志中展现的信息，以分析测试结果，就像下面展示的这个例子：



当你打印一个活动测试日志时，你打印的文档将匹配与你在屏幕上看到的内容，如此确定要展现在你的屏幕上的细节的水平。要在报告中获得更多的细节，点击**Event Type**栏中的（＋）号。要减少细节数量，点击**Event Type**栏中的（－）号。

要打印一个活动的测试日志：

1 展现你要在Test Log窗口中打印的日志。

2 点击**File > Print**。

日志事件属性类型的管理（Managing Log Event Property Types）

你可以管理日志事件属性去注册附加的日志事件属性以展现在TestManager中。在你添加一个新的日志事件属性类型－－依赖与它如何被定义－－它展现在Log Event窗口的一个单独的标签中，作为文本或者一个HTML文件，或者在一个单独的应用中。更多有关属性类型的信息，参阅133页*Viewing Events Details*的内容。

注意事项：日志事件属性类型关系到TestManager的扩展性。当整合一个新的测试脚本类型时，你可以定义附加的日志事件属性。更多信息，参阅*Rational Test Extensibility Reference*手册。

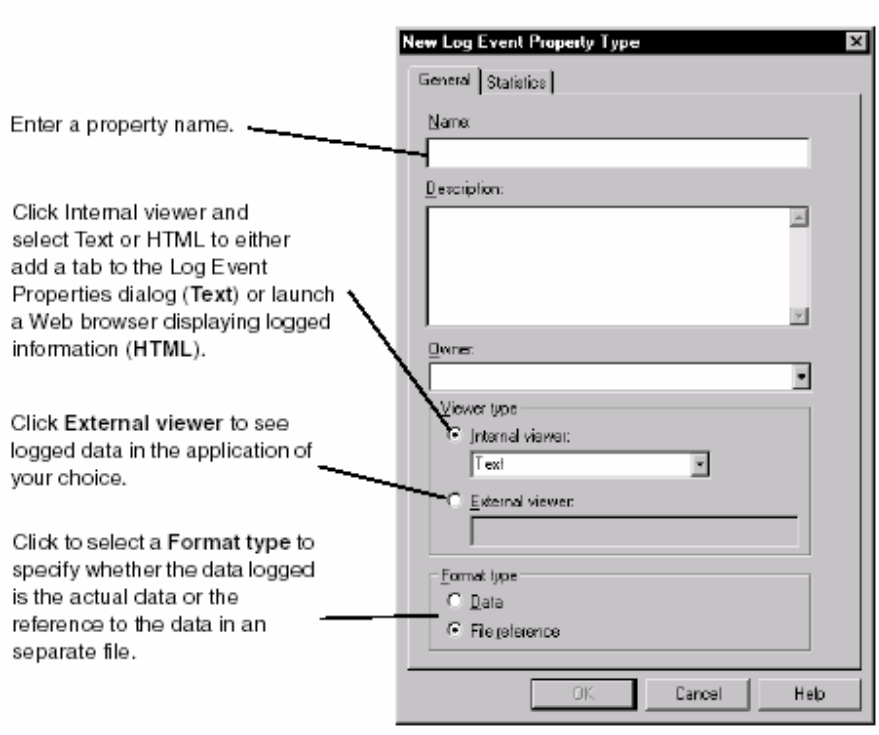
TestManager提供一个默认的日志属性类型**Virtual Tester Associated Data**。

这不仅仅适用于通过http_request指令，TSS登陆方法（诸如VB中的`TSSLog.Message`）产生的数据，也适用与SQABasic定时声明。

例如，如果你正在执行一个倚赖Web服务器的测试，那么你可以设置一个事件类型，它明确地登陆HTTP请求在该Web服务器上。

要创建或者编辑一个日志事件属性类型：

● 点击**Tools > Manage > Log Event Property Types**。点击**New**。



由 Rational Robot 记录的测试脚本结果的查看（ Viewing Test Script Results Recorded with Rational Robot）

你可以使用Rational Robot来记录包含了查证点的测试脚本。在你录制回放该测试脚本之后，Robot写结果到一个日志中。某个查证点也具有被保存的基线数据文件（baseline data files）。你可以使用适当的比较器（Comparators）来查看实际的数据或者图象文件，以及查看和编辑需要的基线文件。

除了使用Test Log窗口去查看查证点的录制回放结果外，你也可以使用它去查看程序上的失败，异常，和任意的附加的回放信息。

一个测试周期可以具有多个针对一个应用程序的特定区域部分的测试。

复查在Test Log窗口中的测试结果展现是否是通过或者失败。分析一个比较器中的结果帮助确定一个测试为什么会失败。复查和分析帮助确定你所处的软件开发的位置，以及是否一个失败是一个缺陷或者一个设计变更。

比较器中的一个查证点的查看（Viewing a Verification Point in the Comparators）

在Test Log窗口的**Details**页中，失败的事件在**Result**栏中被指明为红色。如果该事件是一个测试脚本的一个失败的查证点，使用Robot产生该脚本，那么你可以使用比较器中的一个来分析这个失败。

要查看一个比较器中的一个查证点：

1 打开一个测试日志。

2 点击**Details**标签。

3 右键点击一个查证点并点击**View Verification Point**。

打开合适的比较器是基于查证点的类型，像下表所展示的。你可以到那时分析结果去确定是否是一个缺陷引起的失败或者是一个应用中的有意变更。

Comparator	Verification points
Text Comparator	Alphanumeric
Grid Comparator	Object Data Menu Clipboard
Image Comparator	Window Image Region Image
Object Properties Comparator	Object Properties

比较器	**查证点**
文本比较器（Text Comparator）	字母数字（Alphanumeric）
表格比较器（Grid Comparator）	对象数据（Object Data） 菜单（Menu） 写字板（Clipboard）
图象比较器（Image Comparator）	窗口图象（Window Image） 区域图象（Region Image）
对象属性比较器（Object Properties Comparator）	对象属性（Object Properties）

注意事项：Rational QualityArchitect使用表格比较器（Grid Comparator）来展现查证点信息。更多有关4个比较器的信息，参阅181页*Using the Comparators*的内容。

在测试日志中失败的说明并不意味着测试之下的应用（application-under-test）是失败的。你需要评估每个查证点的失败伴随着合适的比较器来确定他是否是一个实际的缺陷，一个录制回放环境的差异，或者一个有意的设计变更生成一个测试之下的应用（application-under-test）的新的build。

录制回放/环境的差异（Playback/Environmental Differences）

在记录环境和录制回放环境之间的差异可以产生失败说明，但它不能代表软件中的一个实际的缺陷。

（This can happen if there are applications or open windows in the recorded environment that are not in the environment, or vice versa。）

这可以发生，如果应用或者打开的窗口在被记录的环境中，不是该环境，或者反之也可以。

例如，如果你创建一个文件使用被记录的环境中的Notepad，那么当你在录制回放该测试脚本时，该文件已经存在，并且测试日志展示与你实际测试的软件无关的一个失败。

你应当分析这些伴随着合适的比较器的失败说明来确定，在测试脚本录制回放或者一个无关的窗口期间，是否Robot不能够被发现的窗口是作为一个应用窗口而已经打开。

对一个 Application Build 的有意更改（Intentional Changes to an Application Build）

对在测试之下应用的修正可以生成失败说明，在测试脚本和使用一个先前的build开发的查证点作为基线。这是特别地准确，如果该用户接口已经改变。

例如，Window Image查证点将一个从被记录的基线图象文件中象素为象素（pixel-for-pixel）的位图与测试之下的应用的版本相比较。如果用户接口变更，那么Window Image查证点将会失败。当有意的应用变更导致失败的时候，你可以容易地使用Image Comparator来更新基线文件去符合新的接口。在其他区域有意的变更，也可以使用其他的比较器来更新。

For information about updating the baseline, see *Using the Comparators* on page 181.

更多有关更新基线的信息，参阅181页*Using the Comparators*的内容。

结果的报告 Reporting Results

TestManager提供给你一套报告的标准，你可以使用它来分析性能和测试用例的结果。

TestManager提供3种报告类型来帮助你进行你的测试工作。

●测试用例报告（Test case reports）－－使用它来跟踪计划实施的过程，以及测试用例的执行。

●列表报告（Listing reports）－－使用它来展现保存在一个Rational项目中的测试资产。

●性能测试报告（Performance testing reports）－－使用它来分析在指定条件下的一个服务器的性能。

关于测试用例的报告（About Test Case Reports）

测试用例报告帮助你跟踪计划实施的过程，以及测试用例的执行。这些报告有多种展现格式，包括栏（bar）图，堆（stack）图，面积（area）图，线（line）图，饼（pie）图，以及树（tree）图。

这是测试用例报告的3种类型：

●测试用例分布（**Test Case Distribution**）报告可以在计划期间和一个项目的实施阶段被起到作用。一个测试用例分布报告可以帮助你确定：

□计划的测试用例的数量。

□由测试脚本实施的测试用例的数量。

□未被实施的测试用例的数量。

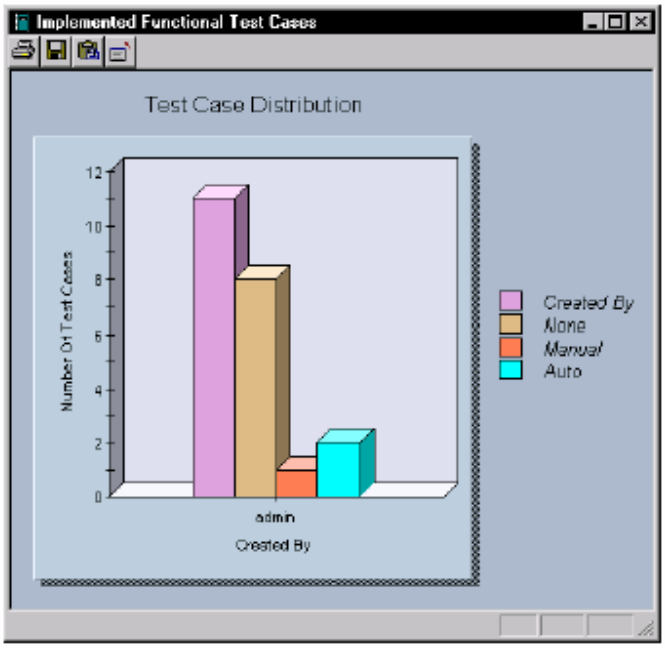
□由手工测试脚本或者自动测试脚本实施的测试用例的数量。

测试分布报告来自于为你产生的两个覆盖率报告。这些测试覆盖率报告，帮助你跟踪你的计划测试的过程，测试开发，以及测试执行的工作。

□计划测试的覆盖率（**Test Planning Coverage**）－－一个测试用例分布报告展现计划的测试输入的百分率和计划的测试用例的数量。

□测试开发的覆盖率（**Test Development Coverage**）－－一个测试用例分布报告有助于编制测试计划，通过展现给你测试输入的数量和百分率，具有的由测试脚本实施的准备去执行的测试用例。这个报告展现给你不仅仅是针对每个测试输入，你是否已经计划了一个测试脚本，也是该测试脚本是否已经被实际记录。

下面的测试用例分布报告展现了通过一定数量的手工或者自动测试脚本实施的测试用例：

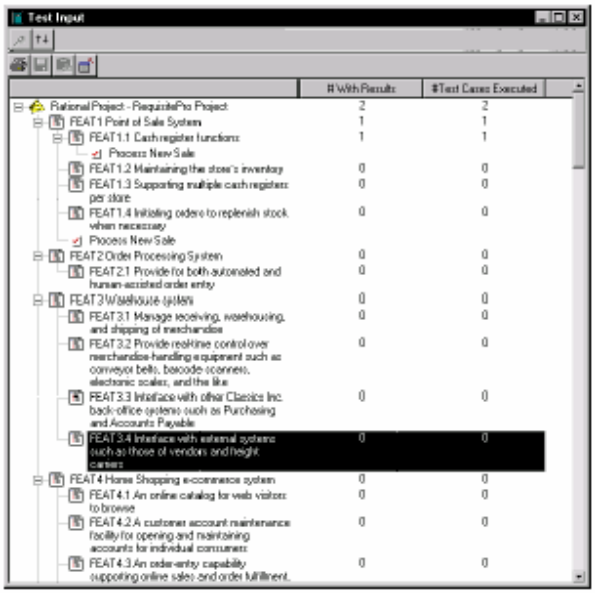


●测试结果分布（**Test Case Results**）报告可以在一个项目的执行阶段使用。这些报告提供有关执行一个测试脚本、测试用例或者测试suite的关键性信息。通过结果信息，你可以评估一个特定build的质量，以及这个build的测试过程。

测试用例结果分布报告来自于已经为你创建的一个覆盖率报告。这个测试覆盖率报告帮助你跟踪你的测试执行工作的过程。

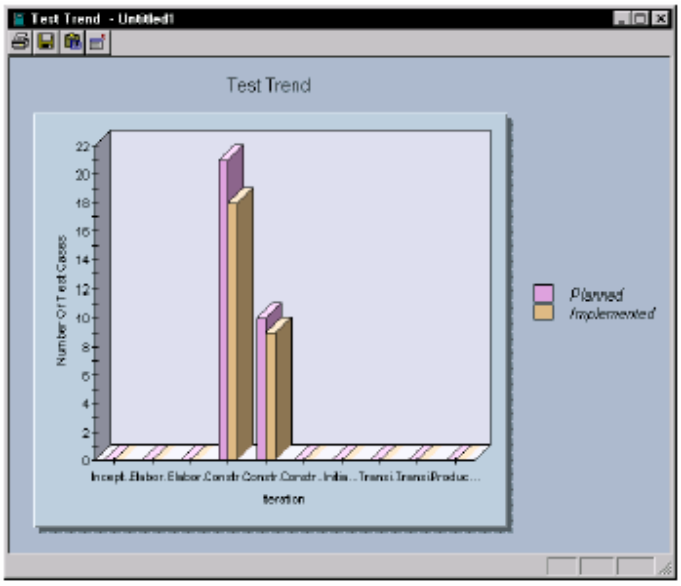
□测试执行的覆盖率（**Test Execution Coverage**）——一个测试用例结果分布报告展现计划的测试用例的数目，执行的测试用例的数目和百分比，通过的测试用例的数目，以及失败的测试用例的数目。

下面的测试用例的结果分布报告展现了有结果的测试用例的数目和针对每个测试输入已执行的测试用例的数目。



●测试用例的趋向报告提供有关信息，如测试的数量，已经计划的、开发的、执行的测试用例的数量，或者是在一段时期内符合测试标准的测试用例的数量，伴随通过builds、迭代或者日期的间隔描述和指定。

下面的测试用例趋向报告展现了遍及所有迭代的计划的和实施的测试用例的数目。



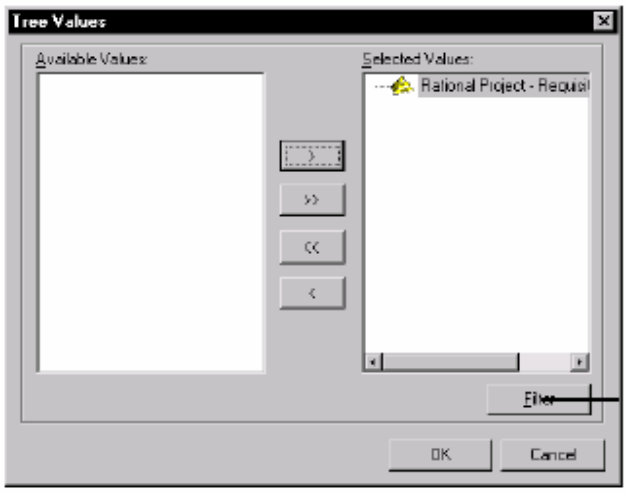
过滤测试输入的来源数据（Filtering Test Input Source Information）

有两种方法可以让你在测试用例分布报告或者测试用例结果分布报告中过滤测试输入的来源数据。

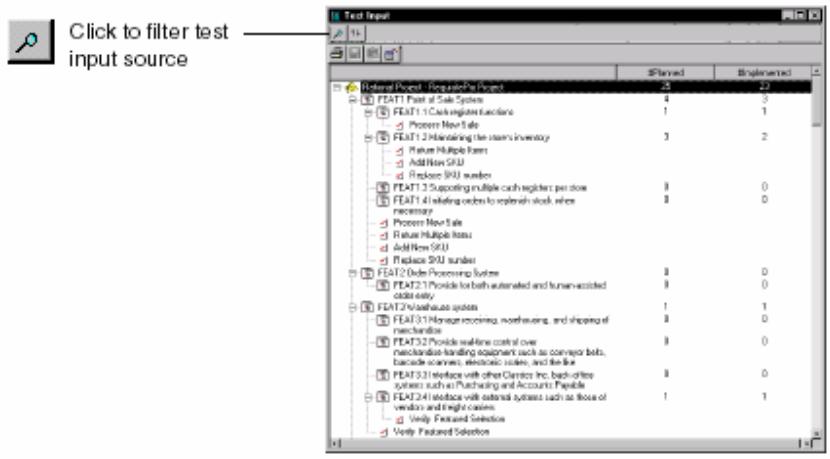
注意事项：当你过滤测试输入的来源数据时，它可以为TestManager花一些时间来生成一个新的

报告。

●在你执行一个报告之前过滤——你可以过滤掉不必要的数据信息，以便缩小展现在一个测试用例报告中的数据数量。

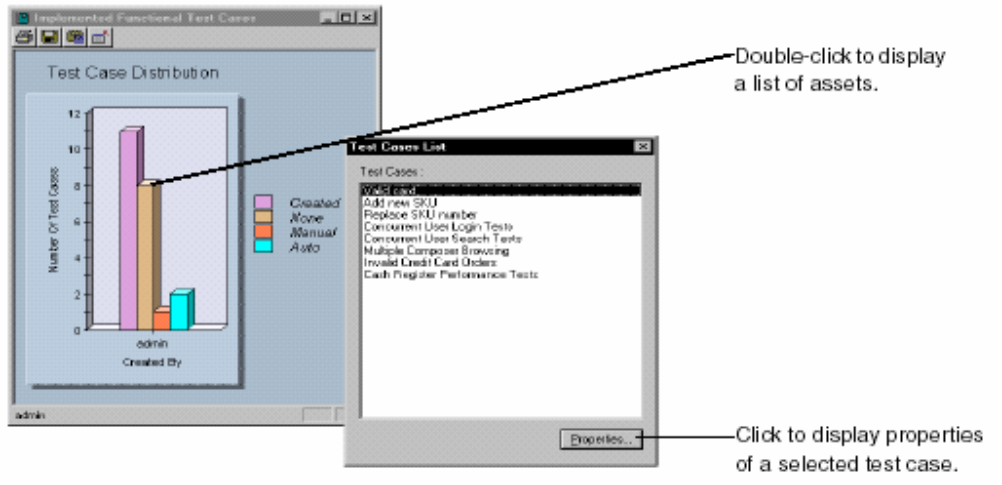


●在你执行了一个报告之后过滤——执行一个报告之后，如果你依然需要去排除不必要的信息和数据，那么你可以过滤掉这些不必要的数据信息。



在一个测试用例报告中查看资产属性（Viewing Properties of Assets in a Test Case Report）

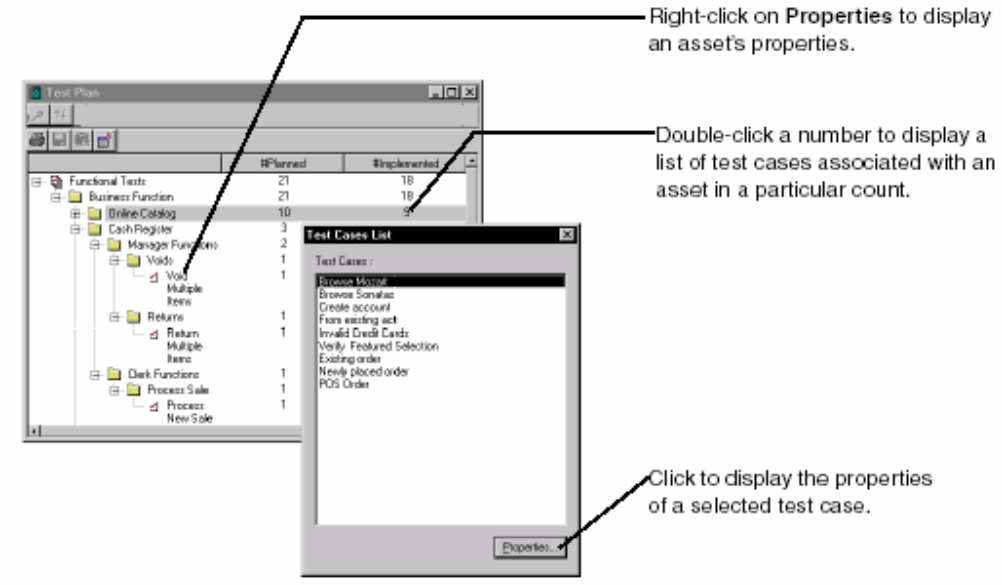
你可以在任意类型的测试用例报告中查看一个资产的属性。双击在一个报告中的资产来展现关联与特殊资产的一个测试用例列表，如下图所示：



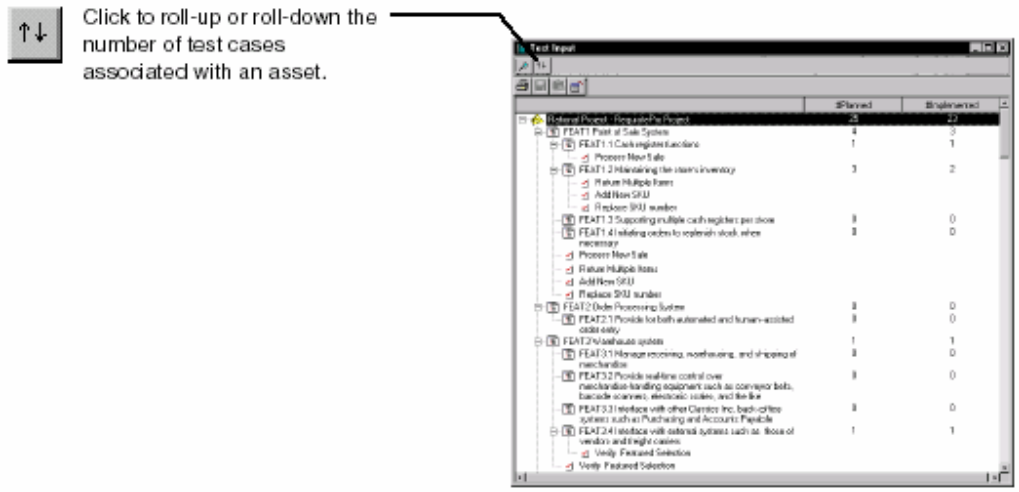
注意事项：当你保存一个报告时，它创建一个JPEG格式的图。你不能双击一个JPEG格式的图来展现一个资产列表。

当你双击测试用例报告的一个树图类型中的一个被计数资产的数目时（一个测试用例分布报告或者测试用例结果报告在你分布在一个测试输入或者测试计划时被创建），该资产在那个特定的count中展现。

例如，在下面的图中，当你双击实施栏中的9时，一个含有9个已实施的测试用例的列表出现在测试用例列表窗口中。要查看任意资产的属性，选择一个资产，并右键点击**Properties**。



你可以在测试用例报告的一个树图类型中，查看全部的关联与一个资产的测试用例的数目。当你“卷起”（roll-up）一个报告时，全部的关联与一个资产的测试用例的数目出现。当你“卷下”（roll-down）一个报告时，关联与一个资产的测试用例的数目出现在每个资产的旁边。



关于列表的报告（About Listing Reports）

列表报告展现保存在一个Rational项目中的不同测试资产的列表。

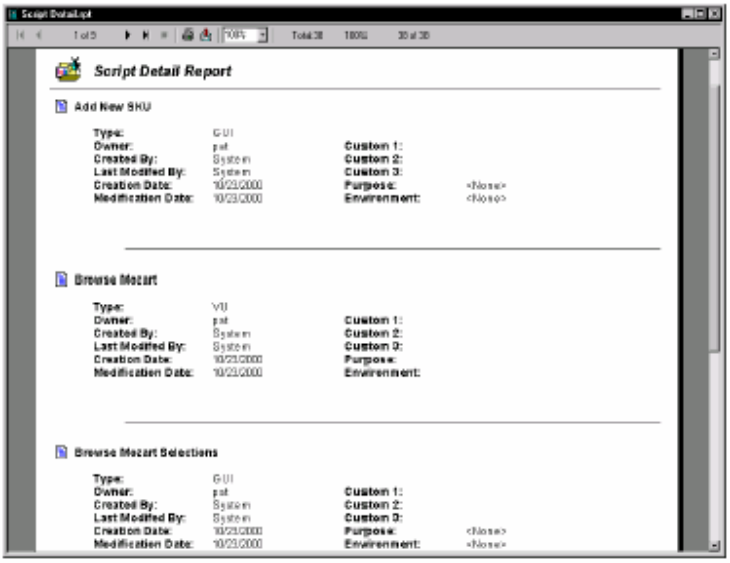
TestManager包含针对builds，测试机，测试机列表，配置，迭代，sessions，suites，测试日志，测试计划，测试脚本，以及用户的列表报告。

每个列表报告来自于一个或者更多的你可以使用的设计布局。一个设计布局定义每个报告的外观和包含在一个列表报告中的说明信息。你也可以定制设计布局或者使用Crystal Reports创建新的设计布局。有关信息，参阅153页*Customizing Design Layouts for Listing Reports*的内容。

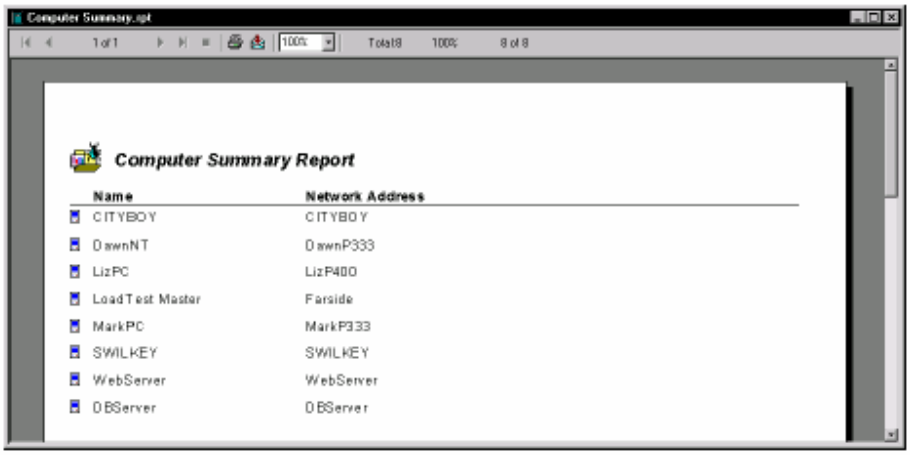
通过使用不同的布局组合以及列表报告，你可以创建准备去执行（ready-to-run）的报告的一个广泛的种类。

例如，使用可用的设计布局和列表报告，你可以创建一个测试脚本的列表报告：

●列出你项目中的所有测试脚本的细节。



●总结概述一个项目中的所有测试机。



你也可以创建一个查询来说明包含在一个列表报告中的数据。有关创建一个查询的信息，参阅的Crystal Reports帮助。

为列表报告定制设计布局（Customizing Design Layouts for Listing Reports）

一个设计布局定义每个列表报告的外观和包含在一个列表报告中的说明信息。要定制已存在的设计布局，或者创建一个新的设计布局，你必须安装Crystal Reports 8.0 Professional Edition（专业版）。Crystal Reports软件在你的Rational软件工具包的一个单独的CD-ROM中。

当你在TestManager中创建一个新的列表报告时，你可以选择性地创建新的或者定制现存的设计布局。Crystal Reports使用资产字典和保存在Rational Test数据库中的属性。这些字典链接各种各样的资产一起使用数据库图解。

更多有关使用Crystal Reports来创建新的或者定制已存在的设计布局的信息，参阅Crystal Reports

帮助。

关于性能测试的报告（About Performance Testing Reports）

性能测试报告帮助你分析在指定条件下，一个服务器的性能。例如，你可以决定一个虚拟测试者多长时间来执行一个命令，以及在不同的suite执行下的响应时间的变化。

你也可以定制报告。

性能测试报告包括：

●性能（Performance）报告。

●比较性能（Compare Performance）报告。

●响应时间（Response vs. Time）报告。

●命令状态（Command Status）报告。

●命令使用（Command Usage）报告。

有关性能测试报告的细节信息，参阅291页*Reporting Performance Testing Results*的内容。

选择要使用的报告（Selecting Which Reports to Use）

下面的表总结了TestManager报告的类型。

To	Use this report	For information, see
Categorize test cases by a particular property. (For example, you can view how many test cases are in each iteration or how many test cases were created by people in a particular testing group.)	Test Case Distribution	*About Test Case Reports* on page 145
Determine the number of test cases that meet your test criteria.	Test Case Results Distribution	*About Test Case Reports* on page 145
Determine the percentage of test cases planned, implemented, or executed for several builds, iterations, or dates: to view the percentage of test inputs tested, not tested, satisfied, or not satisfied for several builds, iterations, or dates.	Test Case Trend	*About Test Case Reports* on page 145
List the builds in your project.	Build Listing	*About Listing Reports* on page 152
List the computers in your project.	Computer Listing	*About Listing Reports* on page 152
List the computer lists in your project.	Computer List Listing	*About Listing Reports* on page 152
List the configurations in your project.	Configuration Listing	*About Listing Reports* on page 152
List the iterations in your project.	Iteration Listing	*About Listing Reports* on page 152
List the sessions in your project.	Session Listing	*About Listing Reports* on page 152

List the suites in your project.	Suite Listing	*About Listing Reports* on page 152
List the test logs in your project.	Test Log Listing	*About Listing Reports* on page 152
List the test plans in your project.	Test Plan listing	*About Listing Reports* on page 152
List the test scripts in your project.	Test Script Listing	*About Listing Reports* on page 152
List the users in your project.	User Listing	*About Listing Reports* on page 152
Display the response times, and calculate the mean, standard deviation, and percentiles for each command in a suite.	Performance	*Performance Reports* on page 308
Compare the response times measured by several Performance reports.	Compare Performance	*Compare Performance Reports* on page 312
Display individual response times and whether a response has passed or failed.	Response vs. Time	*Response vs. Time Reports* on page 317
Obtain a quick summary of which commands passed or failed.	Command Status	*Command Status Reports* on page 320
View cumulative response time and summary statistics, as well as throughput information for emulation commands for all test scripts, and for the suite run as a whole.	Command Usage	*Command Usage Reports* on page 322

To	**使用的报告**	**有关信息，参阅**
利用一个特殊的属性分类测试用例。（例如，你可以查看在每个迭代中有多少的测试用例或者在一个特殊的测试组中通过人员创建的测试用例的数量。）	测试用例分布报告	145页的About Test Case
确定符合你的测试标准的测试用例的数量。	测试用例结果分布	145页的About Test Case
确定为一些builds，迭代，或者日期计划的，实施的，或者执行的测试用例的百分比；查看针对一些builds已测试的，未测试的，满	测试用例趋向	145页的About Test Case

意的，或者不满意的测试输入的百分比。		
编列你项目中的builds	Build列表	152页的About Listing Report
编列你项目中的测试机	测试机列表	152页的About Listing Report
编列你项目中的测试机列表	测试机表的列表	152页的About Listing Report
编列你项目中的配置	配置的列表	152页的About Listing Report
编列你项目中的迭代	迭代的列表	152页的About Listing Report
编列你项目中的session	Session的列表	152页的About Listing Report
编列你项目中的suites	Suite的列表	152页的About Listing Report
编列你项目中的测试日志	测试日志的列表	152页的About Listing Report
编列你项目中的测试计划	测试计划的列表	152页的About Listing Report
编列你项目中的测试脚本	测试脚本的列表	152页的About Listing Report
编列你项目中的用户	用户的列表	152页的About Listing Report
展现响应时间，和估计平均数、标准偏差和一个suite中的针对每个命令的百分比	性能	308页的Performance Report
比较由多个性能报告估量的响应时间	比较性能	312页的Compare Performance Report
展现单独的响应时间以及是否一个响应已经通过或者失败	响应时间	317页的Respone Vs. Time Report

获得命令通过或者失败的一个快速的摘要	命令状态	320页的Commond Status Report
查看累计响应时间和摘要统计，以及针对所有测试脚本的仿真命令，和针对一个suite执行作为整体的“吞吐量”的信息	命令使用	322页的Commond Usege Report

设计你自己的报告（Designing Your Own Reports）

如果你是一个有经验的Crystal Reports用户，你可以创建除列表报告之外你自己的常规报告来符合你的测试团队的需要。有关信息，参阅Crystal Reports帮助。

附加报告（Additional Reports）

附加报告在Rational ClearQuest和Rational SoDA中是可用的。

你可以使用ClearQuest报告，和设计布局，查询，和图表来帮助你管理缺陷数据库。这些报告和其他的条款是为你在创建一个关联ClearQuest数据库的项目时自动创建的。有关使用这些缺陷报告的信息，参阅ClearQuest的帮助。

有关创建一个项目的信息，参阅*Using the Rational Administrator*手册。

你也可以使用SoDA创建报告。Rational SoDA是一个报告生成工具，以支持报告和规范的文档需求。通过SoDA，你可以从不同的信息来源中取回信息，诸如Rational Rose 和Rational RequisitePro，去创建一个单一的文档或者报告。有关使用Rational SoDA创建报告的信息，参阅SoDA的帮助。要使用SoDA，点击**Reports > SoDA Report**。

创建报告（Creating Reports）

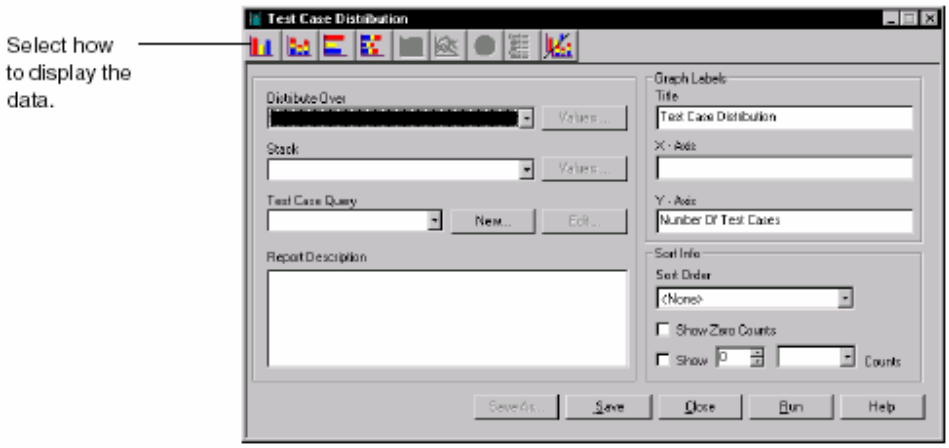
要创建一个报告：

● 点击**Reports > New**，并选择你想要创建的报告类型。

创建一个测试用例分布报告（Creating a Test Case Distribution Report）

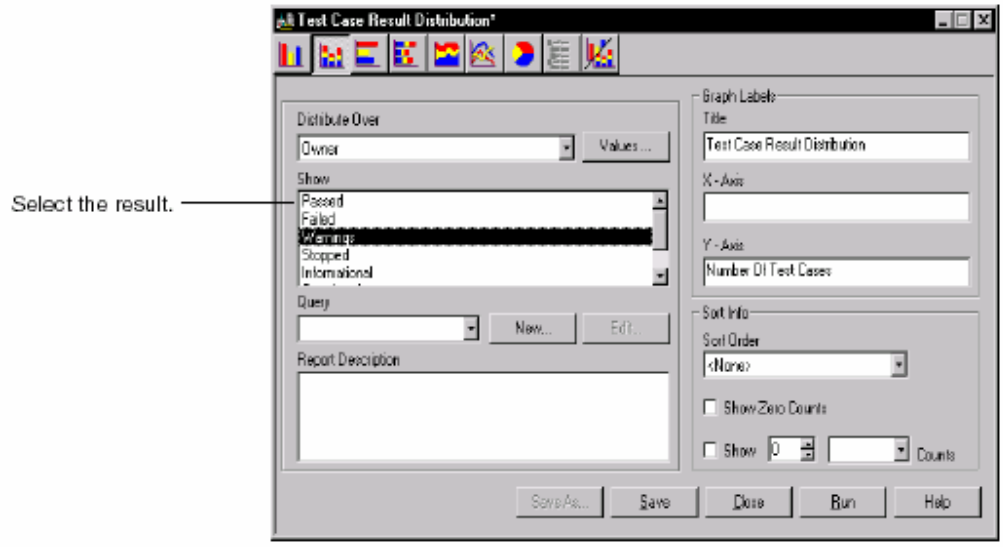
当你创建一个测试用例分布报告时，你可以选择数据如何出现：

在栏（bar）中，堆（stack）中，行（line）中，饼（pie）图中，或者树型报告，这依赖与你选择的报告类型。



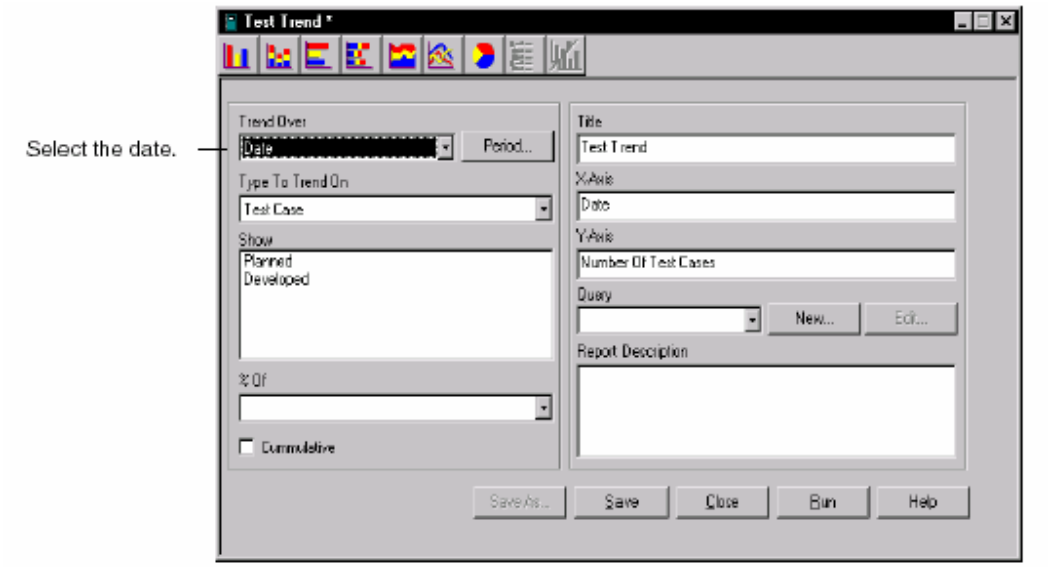
创建一个测试用例结果分布报告（Creating a Test Case Results Distribution Report）

当你创建一个测试用例结果分布报告时，你选择报告中测试用例结果：



创建一个测试用例趋向报告（Creating a Test Case Trend Report）

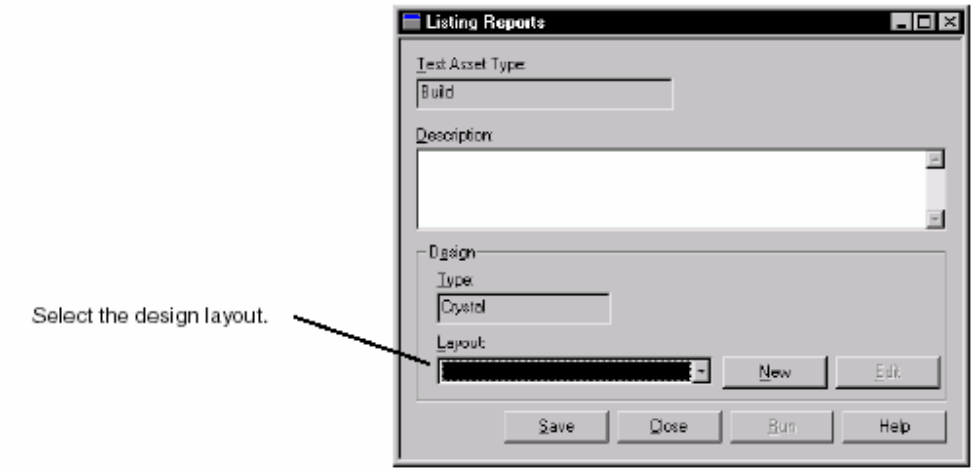
当你创建一个测试用例趋向报告时，你选择有关测试用例和测试输入的数据信息，它们越过你要显示在一个报告中的一些builds，迭代，或者日期。



创建一个列表报告（Creating a Listing Report）

当你创建一个列表报告时，你确定你如何通过一个Crystal Reports设计布局来展现你想要的信息。你可以创建新的或者定制现存的Crystal Reports设计布局。有关信息，参阅153页*Customizing Design*

*Layouts for Listing Reports*的内容。



创建性能测试报告（Creating Performance Testing Reports）

当你创建性能测试报告时，你可以说明执行报告的日志数据和如何操作这些日志数据以便你刚好看到你需要的信息。有关创建性能测试报告的细节信息，参阅291页的*Reporting Performance Testing Results*内容。

打开一个报告（Opening a Report）

在创建和保存一个报告之后，你可以打开它，如果必要的话，对该报告可做更改。

要打开或者变更一个报告，可依照下面的一种方法来做：

●点击**Reports > Open**，从列表中选择一个报告，并点击**OK**。

●在Test Asset Workspace（测试资产工作区）的**Analysis**标签中，选择你要打开的报告类型。选择你要打开或者修改的特定的报告。

●从Report栏（只有性能测试报告）中打开一个报告。

有关打开一个报告的细节信息，参阅TestManager的帮助。

报告的执行（Running Reports）

你可以从以下的地方来执行报告：

●在测试资产工作区（Test Asset Workspace）。

●报告（Report）菜单。

●报告(Report)栏(只有性能测试报告)。有关信息，参阅294页的*Running a Report from the Report Bar*内容。

从测试资产工作区执行一个报告（Running a Report from the Test Asset Workspace）

要从测试资产工作区（Test Asset Workspace）执行一个报告：

●在测试资产工作区（Test Asset Workspace）的**Analysis**标签中，扩展执行的报告的类型。右键点击特定的报告，并点击**Run**。

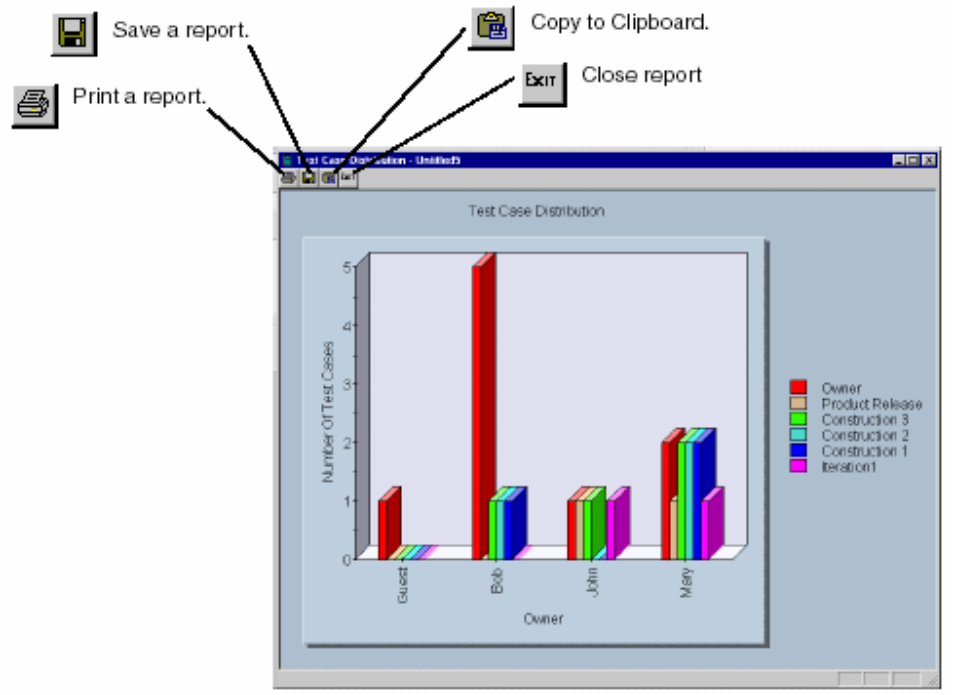
从菜单执行一个报告（Running a Report from the Menu）

从菜单执行一个报告：

●点击，并选择要执行的报告类型。选择一个特定报告，并点击**OK**。

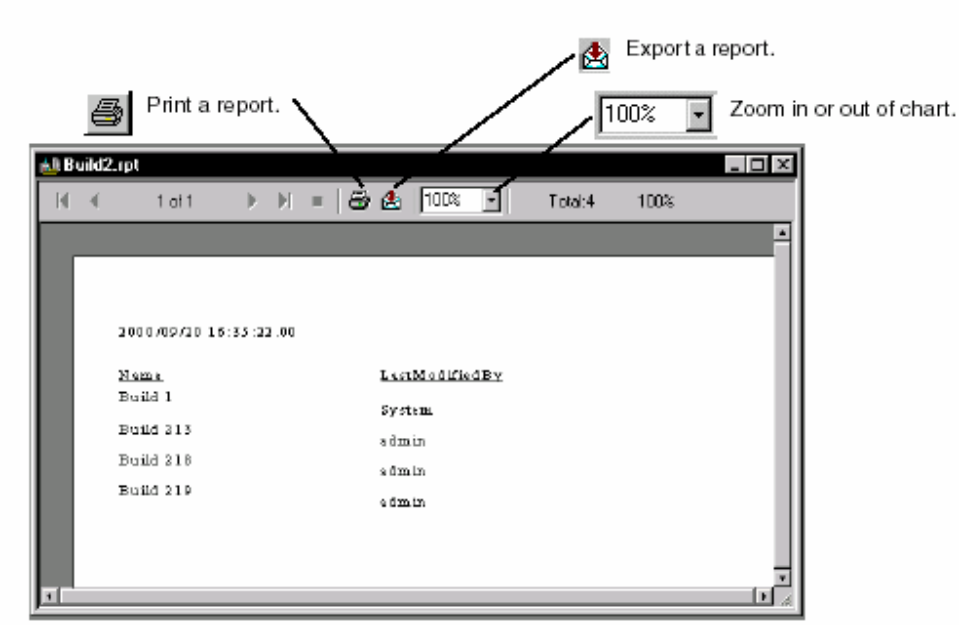
打印，保存，或者复制一个测试用例报告（Printing, Saving, or Copying a Test Case Report）

在你执行一个测试用例报告之后，你可以打印，保存或者复制它到剪贴板（Clipboard）中。



在一个列表报告上打印，输出，或者缩放（Printing, Exporting, or Zooming in on a Listing Report）

在你执行一个列表报告之后，你可以打印它或者输出它到一个不同的文件格式中，并保存它在你的测试机上。你可以输出一个完成的文件到一些大众的spreadsheet，文字处理格式，和到HTML，ODBC，以及普通的数据交换格式。这样使得分布信息的进行变得容易。例如，你可能要使用在一个spreadsheet中的项目趋向报告或者要发信给你的测试团队中的其他成员。



复制报告到一个新项目中（Copying Reports to a New Project）

如果你创建一个报告或者一个新的设计布局，并且要在一个新的项目中使用它，那么可以使用Rational Administrator，在你创建一个新项目时复制它们。Rational Administrator复制任意已保存的列表报告和列表设计布局到这个新的项目中。

有关信息，参阅Administrator帮助。

创建一个查询（Creating a Query）

一个查询是针对从一个Rational Test数据库中说明信息的一个请求。你可以为TestManager报告中的每个类型创建一个查询。

针对测试用例分布，测试用例趋向，和性能测试报告的查询（Queries for Test Case Distribution, Test Case Trend, and Performance Testing）

Report

TestManager提供预定义（pre-defined）的查询来缩小测试用例分布，测试用例结果分布，测试用例趋向，和性能测试报告中的数据。你可以编辑现存的查询和为这些报告创建你自己的查询。

要创建一个查询，依照下面的一种方法：

●创建或者打开一个报告，并点击**New**按钮到**Query**区域。

● 点击**Tools > Manage > Queries > Test Case**。

针对列表报告的查询（Queries for Listing Reports）

要创建一个针对列表报告的查询，你必须安装Crystal Reports 8.0 Professional Edition（专业版）。Crystal Reports软件可在你的Rational软件工具包中的一个单独的CD-ROM中获得。有关为列表报告创建一个查询的信息，参阅Crystal Reports帮助。

Part 2: Functional Testing with Rational TestManager

功能测试 suites 的创建（Creating FunctionalTesting Suites） 7

本章描述如何创建功能测试的suites。它包含了下面的标题内容：

●关于suites

●向suite中插入一个测试机组

●向suite中插入一个测试脚本

●向suite中插入一个测试用例

●向suite中插入一个suite

●在一个测试脚本、测试用例、或者suite上设置一个前置条件

●向suite中插入一个选择器

●向suite中插入其他项

●在一台指定的测试机上执行测试

●在不同的测试机上分布测试

●执行suites

关于 Suites（About Suites）

一个suite显示一个分层的你要测试的任务表示。它展现一些项，如执行测试的测试机。执行的测试脚本，以及每个测试脚本执行的次数。

Through a suite, you can:

通过一个suite，你可以：

●分配测试用例到测试机，在没有再分配它们的情况下返回这些测试用例。

●在下一台可用的测试机上执行测试脚本和测试用例，从而加快你的测试过程。

●在一个suite中的项上设置前置条件，它是在suite中执行下一个项之前需要它们成功地完成。

●同步化虚拟测试者。

注意事项：在本章地suites中包含了GUI测试脚本，它一般被用来进行功能测试。一个suite，当然，也可以包含VU测试脚本，VB测试脚本，或者其他地“用户定义”的测试脚本类型。

向 suite 中插入一个测试机组（Inserting a Computer Group into a Suite）

当你为功能测试创建一个suite时，你首先要设置**computer groups**（测试机组）。一个测试机组包含该suite执行的测试脚本，并声明测试机是可用与该suite的。

你的测试可执行在你在TestManager中已经定义的任意Agent（代理）测试机上。如果你还没有定义任意的测试机，TestManager会在Local（本地）测试机上执行你的测试。有关定义测试机的信息，参阅69页*Defining Agent Computers and Computer Lists*的内容。

如果你只是插入一个测试机组并接受默认方式，那么TestManager创建一个测试机组。当你想使用multiple computer groups（并联的测试机组）：

●分配某些项来执行在某个测试机集合上。这些项分配到一个组将仅仅使用被分配到那个组的测试机。

●在一个suite中混和GUI和VU测试脚本。GUI和VU测试脚本必须在不同的测试机组中。

当你插入一个测试机组到suite中时，你必须决定你何时分配这些测试机。你使用的这个方法适用与完整的suite。你可以：

●在你插入测试机组时，分配指定的测试机。该suite将执行，当然，如果这些测试机在执行期是可用的话。

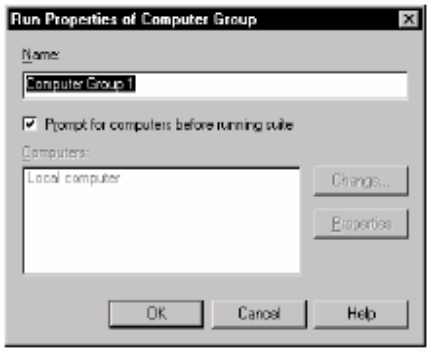
●等待，直到执行期时分配指定的测试机。当你执行suite时，TestManager促使你为使用该测试机。在执行期间，该suite将会执行在任意存在的这些可用的测试机上。

当你在执行期间分配测试机时，你要限制该suite到一个测试机组。

●在任意空闲的以运行一个测试脚本的测试机上执行测试。这被称作分布式功能测试（*distributed functional testing*）。有关分布式功能测试的信息，参阅178页的*Distributing Tests Among Different Computers*内容。

要插入一个测试机组到一个suite中去：

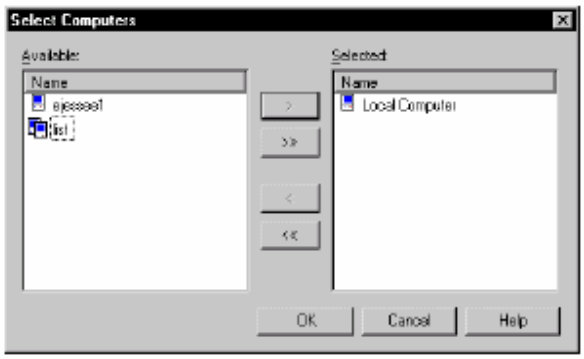
●点击**Suite > Insert > Computer Group**。



此时，该测试机组中的虚拟测试者被分配到本地（Local computer）测试机上。

要分配代理测试机（Agent computer）给组内的虚拟测试者：

●清除**Prompt for computers before running suite**，点击**Change**。

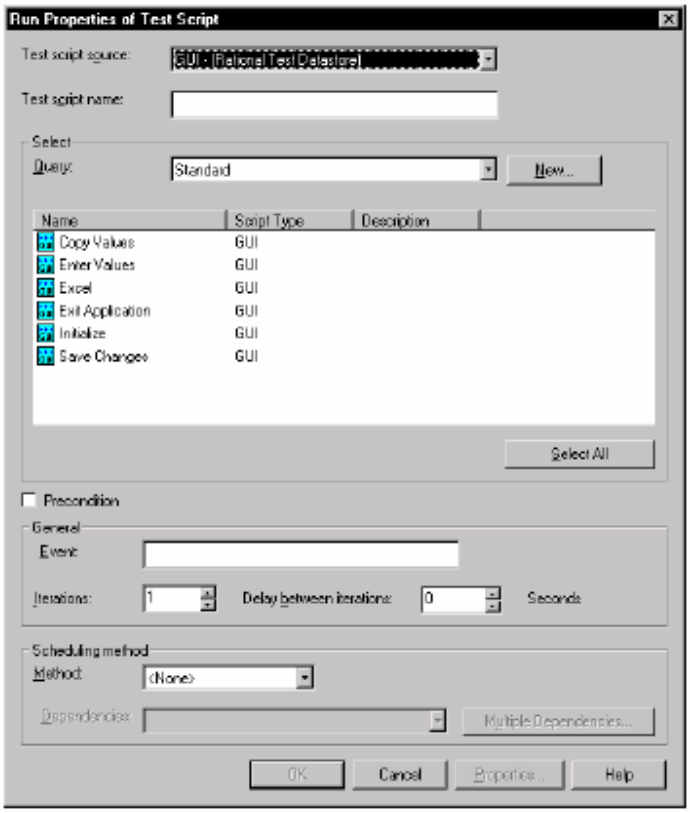


向 suite 中插入一个测试脚本（Inserting a Test Script into a Suite）

在你插入一个测试机组到一个suite中后，你可以插入测试脚本，该测试机组将会执行。一个测试脚本一次运行在一台测试机上。

要插入一个测试脚本到一个suite中去：

●从一个打开的suite中，选择测试机组以执行测试脚本，然后点击**Suite > Insert > Test Script**。



你可以设置一个关于测试脚本的前置条件。当你设置一个前置条件时，测试脚本必须成功地完成，以使其他的具有相同“父类”的suite项执行。例如，一个测试脚本可能确立了软件中的某种状态。你可以执行该测试脚本以确立该状态，然后再执行依赖与系统状态的一系列的测试。有关前置条件的信息，参阅174页的*Setting a Precondition on a Test Script, Test Case,* 或者*Suite*内容。

向 suite 中插入测试用例（Inserting a Test Case into a Suite）

测试用例使你：

- 在没有考虑它的实施的情况下定义一个测试。久而久之，实施可以被改变，但是测试用例仍然是一样的。它的好处是你可以创建一个具有一个测试用例的suite，并在没有更新或者保持该suite的情况下改变实施（测试脚本）。
- 插入测试用例到suite中，以使你可以一次执行并联的测试用例，并保存在一起执行的测试用

例集合。

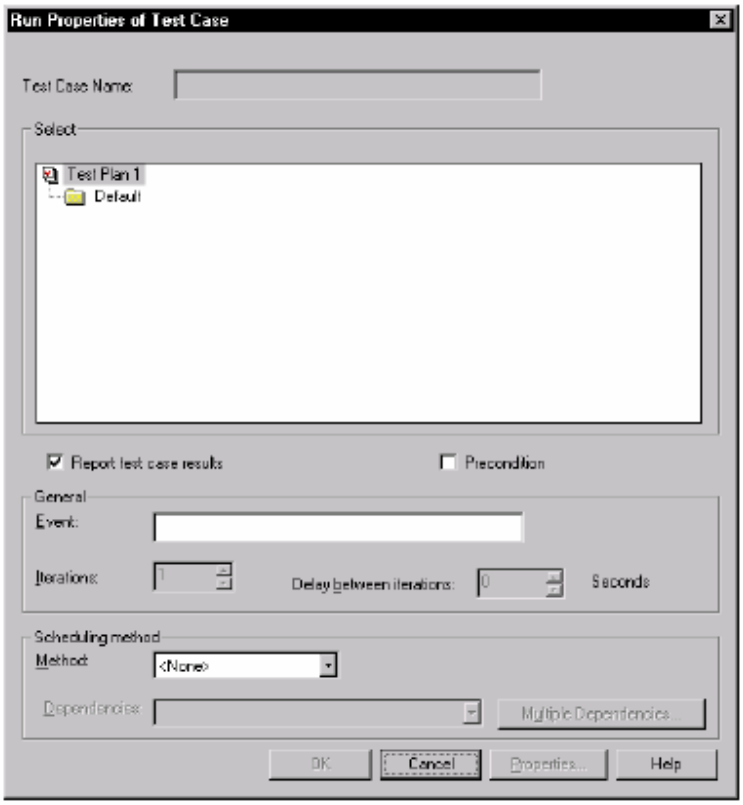
●插入配置的测试用例以查证一个测试用例在多样的不同环境中顺利进行。

To insert a test case into a suite:

. Click **Suite > Insert > Test Case**.

要插入一个测试用例到suite中：

●点击**Suite > Insert > Test Case**。



你可以设置一个关于测试用例的前置条件。当你设置一个前置条件时，测试用例必须成功地完成，以使其他的具有相同“父类”的suite项执行。

有关前置条件的信息，参阅174页的*Setting a Precondition on a Test Script, Test Case,* 或者*Suite*内容。

要设置一个关于测试用例的前置条件：

●右键点击测试用例来实施前置条件，并选择**Run Properties**。

插入 suites 和 Scenarios 到 suites 中（Inserting Suites and Scenarios into Suites）

插入suites或者scenarios到一个suite中能够使你保持suite项的一个层次结构。当你插入一个suite或者scenario到一个suite中时，你可以：

●重用suite的项，在没有将他们重复到一个suite的复合区域中的情况下。

●将suite的项聚集在一起，以使它们可以被一个以上的测试机组所共享。

●更加容易地保持你的suite。这是尤其准确的，如果你有一个复杂的使用了很多测试脚本的suite的话。在一个suite或者scenario下聚集这些suite项，有一个优点，就是让你的suite更容易的保持和阅读。

在功能测试中，你有代表性地插入一个suite到另一个suite中，因为插入一个suite将提供更大的灵活性。一般而言，插入一个suite到另一个suite中，是在：

●你希望在复合的suites中重用一系列的项。你可以插入一个suite到不同的suites中去。

●你希望的任何变更，你将一个suite被复制与那个suite的每一个实例中。

插入一个scenario到一个suite中时：

●你希望重用一个suite中的一系列项。这时，你不能插入一个scenario到不同的suites中去。

●你希望在你打开一个suite时看到这个suite所有项的层次。这时，Scenario可以让你看到这个结构。如果你插入一个suite到另一个suite中，那么你只有打开这个子suite来查看它的项。

例如，你可能创建了三个suites，每一个测试一个帐目结算应用程序的一个不同的方面：

●一个suite打开并编辑数据表。

●一个suite测试所有的菜单。

●一个suite测试数据表中的那些复杂的公式。

全部的这三个suites都需要虚拟测试者以打开这个帐目结算的应用程序。然而，在每个suite中，唯一的任务需要被重复。你可以创建一个单独的suite，打开这个应用程序，并将这个suite插入到上面那三个suites的每一个中去。

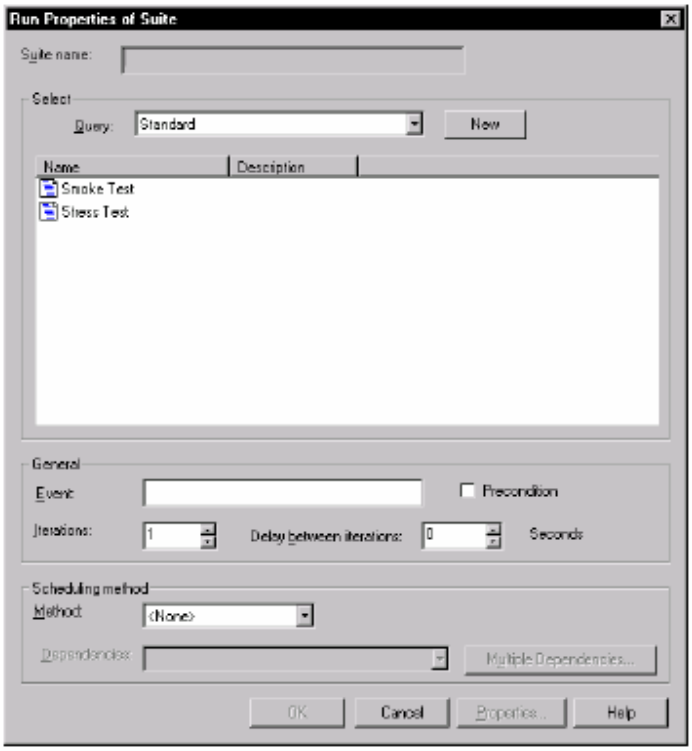
插入suite到另一个suite中（Inserting a Suite into a Suite）

如果一个suite包含测试机组，那么你可以将其插入到其他的suites中去。当你正在创建一个复杂的测试，或是创建执行双重功能的多重测试时，插入一个suite到另一个suite中去是有利的。你可以创建并核对一个suite，然后再将其插入到更大一些的suite中。这样的话，不必再为每

一个suite定义相同的测试资产了，这让你节省了时间。任何的变化，将使一个suite被复制在那个suite的每个实例中。

插入一个suite到另一个suite中：

●点击Suite > Insert > Suite。



你可以设置suite的前提条件。在你设置一个前提条件时，这个suite必须成功地完成，以使其他的suite所有项伴随这相同的父类执行。有关前提条件的信息，参阅174页*Setting a Precondition on a Test Script, Test Case, or Suite*的内容。

设置关于一个suite的前提条件：

●右键点击该suite以设置前提条件，然后选择Run Properties。

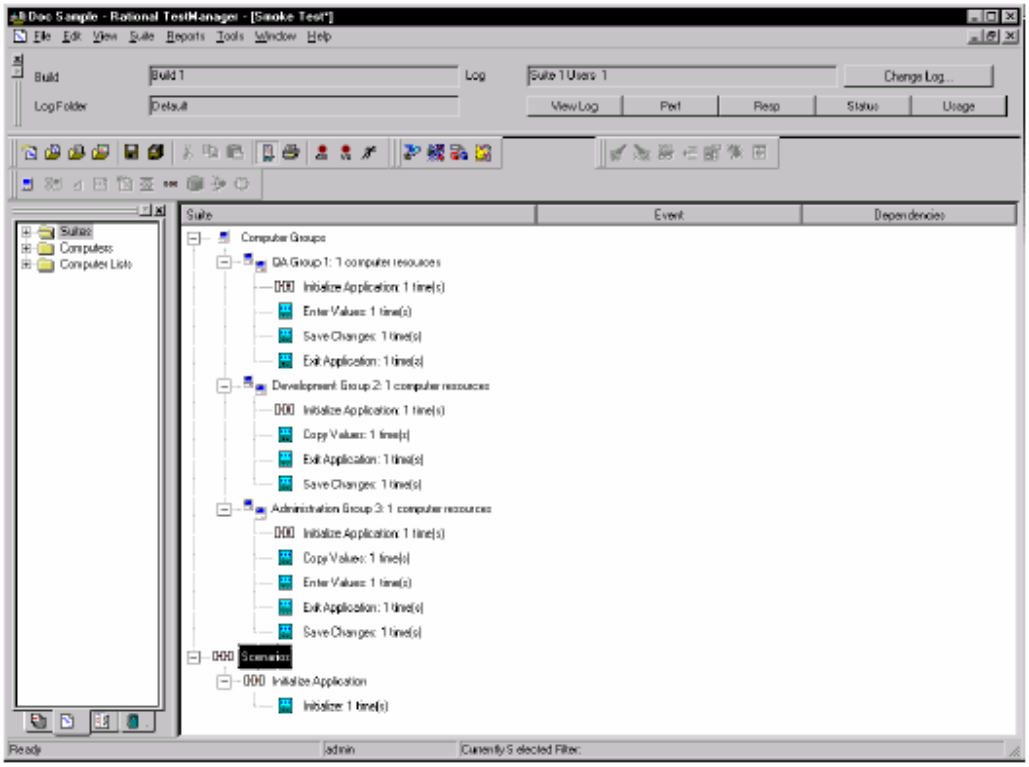
插入Scenario （Inserting a Scenario）

通过插入一个scenario和在suite里插入项，你可以在该suite的Scenarios部分中定义一个scenario。

创建一个测试机组来执行scenario，那么你需要在测试机组中插入这个scenario的名称。否则，这个scenario将不被执行。

以下的suite中，在测试一个应用程序的各部分前，执行测试教本的所有这三个测试机组需要初始化该应用程序。通过在scenario中保存这个必须的初始化教本，你可以将这个suite简单化。

该suite表示测试脚本的Initialize被当作Initialize Application scenario的一部分。在测试脚本执行后，延迟可能被添加到这个scenario中，而那些变更将过滤Initialize Application scenario的所有实例。



创建新的scenrio：

●从该suite的**Scenarios**部分，点击**Suite > Insert > Scenario**。

插入scenario到一suite中：

●点击你希望放置scenario的区域，然后再点击**Suite > Insert > Scenario**。

在你创建了scenario和执行该scenario测试机组之后，组装（populate）这个scenario是个不

错的主意。Scenario仅仅需要去执行测试脚本。然而，像测试机组，一个真正的scenario也可以包含测试用例，suites，和选择器。

设置测试脚本，测试用例或者suite的前提条件（Setting a Precondition on a Test Script, Test Case, or Suite）

在你插入测试脚本，测试用例，或者suite到另一个suite中时，你可以说明，对于该suite的剩余项来说，那个项的成功完成是一个（*precondition*）前提条件。对于在相同级别下执行的剩余suite项来说，该项必须通过。

例如，假设一个suite包含两个suites，而它们中的每一个都包含了初始化的测试脚本和若干的测试用例。如果你设置了这个初始化的测试教本的前提条件，并且这个测试脚本执行失败了，那么TestManager仅仅在那个suite中浏览所有剩余的测试用例。在第二个suite开始时恢复该suite的执行。

设置前提条件：

●右键点击该测试脚本，suite，或者测试用例以设置前提条件，并选择**Run Properties**。

前提条件仅仅适用于测试脚本，测试用例，或者suite实例的说明。

例如，如果你多次地插入一个测试脚本，并且你希望设置该测试脚本所有实例的前提条件，那么你就必须为每一个测试脚本设置前提条件。

插入选择器到 suite 中（Inserting a Selector into a Suite）

TestManager允许你去设置suite项，通过设置选择器，以在不同的序列中执行。比起suite中连续项的一个简单的序列，一个选择器提供了更加复杂的控制。选择器告诉TestManager将要执行的项，以及所处的序列。例如，你可能希望从一组测试脚本中随机的重复地选择一个测试用例。选择器帮助你可以这样做。

下面的列表说明被用于功能测试的选择器类型，以及，其他被用于性能测试的选择器类型。

●**Sequential（顺序的）**－按顺序执行出现在这个suite中的每一个suite项。当然，这是默认的。

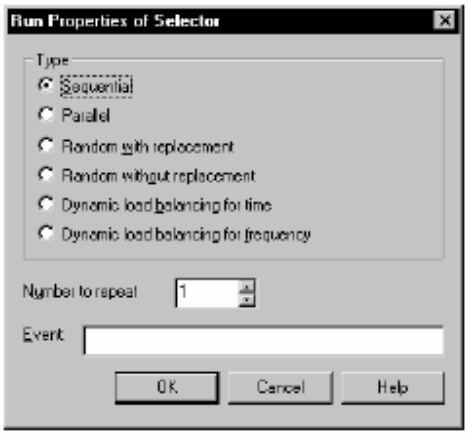
●**Parallel（并行的）**－分布每一个suite项到任意可用的测试机。这个选择器常被用于分布式的功能测试。Suite项被整齐地分割出来，基于可用的测试机来执行另一个测试脚本。一旦有一个项执行，它不会再被执行。

在不把它当做是迭代的情况下，一个并行的选因择器分布在每个测试脚本中。例如，假设脚

本A对于10个迭代执行，而脚本B只对于一个迭代执行。迭代的数量将不影响脚本分布的方法。

插入选择器到suite中：

●选择要包含选择器的测试机组或scenario，然后，点击Suite > Insert > Selector。



插入其他项到 suite 中（Inserting Other Items into a Suite）

分布在以下部分的项一般被用来在性能测试中使用。

当然，你有时也可以在功能测试中使用这些项。

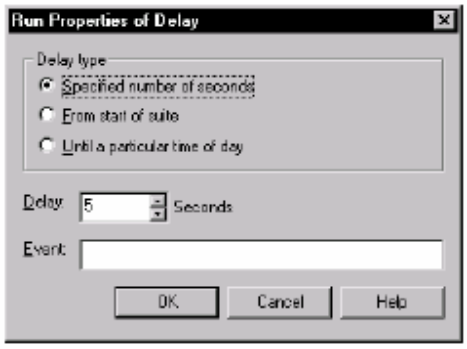
插入延迟（Inserting a Delay）

一个delay（延迟）告诉TestManager，该suite中，在它执行下一个项之前，需要多长时间的暂停。

在功能测试中，执行之前，你可以使用delays（延迟）来引起测试脚本的等待。例如，如果一个虚拟测试者更新记录，那么你可以插入一个delay（延迟），给予应用程序时间以处理和显示正确的信息。通过提供一个delay（延迟），你可以确保应用程序有足够的时间去完成一个任务，因为（in case）另一个虚拟测试者执行一个动作，以作为那项任务的结果。

插入一个延迟到suite中：

点击要添加delay（延迟）的计算机组，scenario，或选择器，然后点击Suite > Insert > Delay。



插入同步点（Inserting a Synchronization Point）

Synchronization point（同步点）通过在特殊点上暂停每个虚拟测试者的执行，使你能够调整许多的虚拟测试者的活动。Synchronization points（同步点）主要使用在进行性能测试的suites中。当然，你可以在进行功能测试的suite中使用Synchronization points（同步点），来测试两个虚拟测试者同时存取一个文件的时将发生的情况。

同步点一般在下面的时间发生时有效：

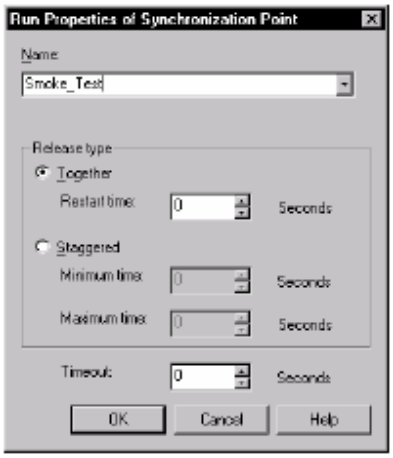
●所有关联了synchronization point（同步点）的虚拟测试者到达该synchronization point（同步点）。

●在所有的虚拟测试者到达该synchronization point(同步点)之前，达到一个超时期(timeout period）。

●在监控suite时，你手工地释放虚拟测试者。

插入同步点到suite中：

●点击**Suite > Insert > Synchronization Point**。



有关synchronization points（同步点）如何工作的信息，参阅249页*How Synchronization*

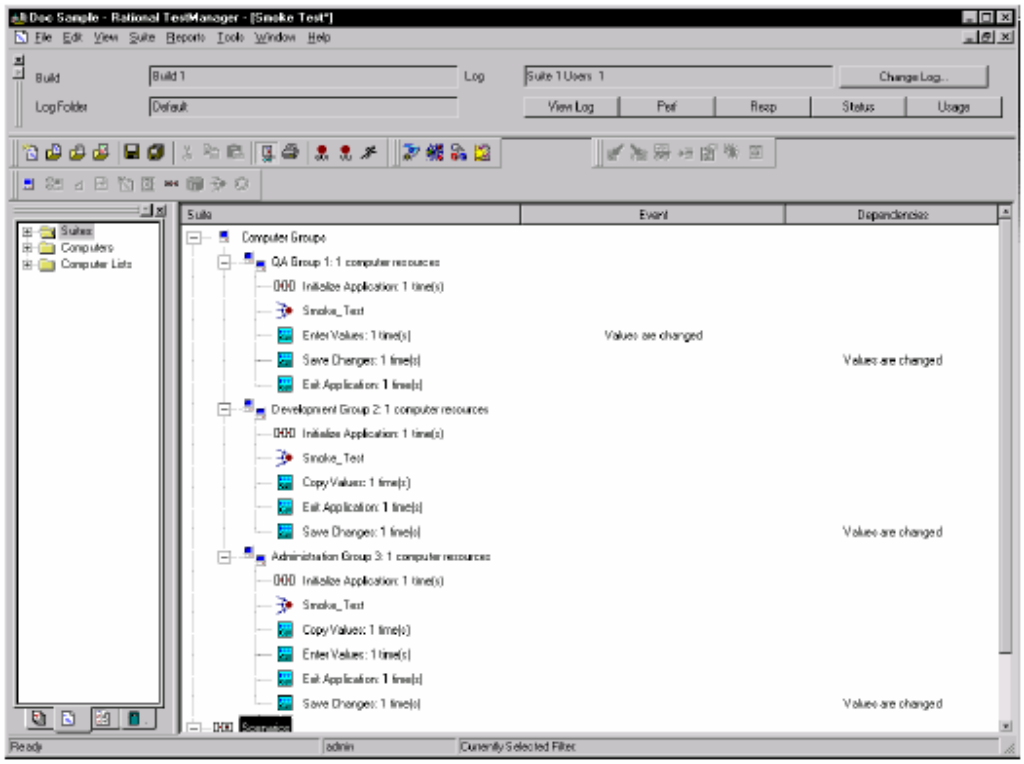
*Points Work*的内容。

使用时间和依赖去调整执行（Using Events and Dependencies to Coordinate Execution）

一个事件是一个机制，调整在suite中执行项的方法。例如，你不能够测试是否应用程序会保存已形成的某个值的变更，除非那些值确实已被变更。你在测试脚本上设置一个（dependency）依赖，保存变更，这样直到事件（这些变更确实已形成）发生才阻碍虚拟测试者。

你可以在suite中具有多重事件。一个suite中，当有且仅有一个项设置一个事件时，这其中的许多项可以依赖这一个事件。

下图说明了虚拟测试者的等待，直到第一个虚拟测试者改变了值时：



在该suite中的第二列里，列出事件，第三列中列出dependencies（依赖）。

添加设置事件的测试脚本，或者添加依赖与事件的测试脚本：

点击**Suite > Insert > Test Script**。

注意事项：这个例子说明如何添加设置事件的测试脚本和依赖与事件的另一个测试脚本。当然，scenario和delays（延迟）也可以设置事件。

在不同的测试机中分布测试（Distributing Tests Among Different Computers）

你可能希望在不同的测试机中分布你的测试脚本。例如，你的测试可能设计到明确的测试机。或者你可能希望在一组测试上执行你的测试，以便它们尽可能快的完成。利用TestManager，你可以并发地执行多个测试机，并且在这些测试机中分布你的测试。这样可以使你加速测试的进程。

在不同的测试机中分布你的测试，可按照这些步骤：

- 当你插入测试机组到suite中时，点击**Change**并添加你的测试机到出现的测试机列表中去。有关设置测试脚本在不同的测试机上执行的更多信息，参阅166页*Inserting a Computer Group into a Suite*的内容。
- 在你已经插入你的测试机组之后，可插入一个并行的选择器。你插入在该选择器下的测试脚本将被不断地发送到下一个可利用的测试机。当然，该测试脚本必须被设计成自包含型（self-contained），并且不相互依赖。有关并行选择器的更多信息，参阅174页*Inserting a Selector into a Suite*的内容。

一个分布式功能测试的例子（Example of a Distributed Functional Test）

在下面的例子中，假设你希望测试你的记帐软件。

你希望分布你的测试到不同的测试机上，以使他们尽可能快的执行。

The following table summarizes how you set up this test.

下表概述如何设置此测试。

Test Scripts	Suite	Reports
A script to log virtual tester in. A modular script for each virtual tester task. A test script to perform any cleanup work and then shut down the application.	One computer group that logs the users in. One computer group that contains a Parallel selector and modular scripts that run on any computer. One computer group that shuts down the application.	Test log report to show whether all virtual testers in the suite successfully ran to completion.

测试脚本	Suite	记录

该表说明执行一个分布式功能测试的方法。有许多使用TestManager的其他方法，构建并执行有效的分布式的测试。

要注意，最重要的是所有的测试脚本都应当被模块化（modular）。

Suits 的执行（Executing Suites）

在你已经创建并保存了你的suite之后，你可以：

- 检查该suite的是否出错。可以这样做，打开suite，然后点击**Suite > Check Suite**。
- 检查代理（Agent）测试机的状态。可以这样做，打开 suite，点击 **Suite > Check Agents**。
- 控制该 suite 的运行时间信息。打开 suite，然后点击 **Suite > Edit Runtime**。
- 控制如何终止该suite。打开suite，然后点击**Suite >Edit Termination**。
- 执行该 suite。打开 suite，点击 **File > Run Suite**。
- 监控该 suite 的执行过程。有关 suites 的信息，参阅 103 页 *Monitoring Suites* 的内容。

Comparators 的使用（Using the Comparators） 8

这一章描述在Rational Robot测试脚本或Rational QualityArchitect中使用查证点时，如何使用Comparators来比较和查看捕获的数据。本章包含下面的标题内容：

●关于基本的四种比较器Comparators

●启动Comparators

●对象属性比较器（Object Properties Comparator）的使用

●文本比较器（Text Comparator）的使用

●表格比较器（Grid Comparator）的使用

●图像比较器（Image Comparator）的使用

注意事项：有关的细节过程，参阅TestManager的帮助。

关于四种基本的 Comparators （About the Four Comparators）

在你录制回放了一个测试用例，测试脚本，或者suite之后，TestManager将结果写入测试日志中，并展现在TestManager的TestLog窗口中。测试日志会告诉你，是否每个测试用例，测试脚本，或者suite都通过或失败。

你可以通过点击测试日志中的Details标签来获得更多的有关细节。当你双击测试日志中的一个已失败的查证点时，会出现一个适合于该查证点的Comparator（比较器）。你可以使用这个Comparators查看和比较捕获的数据，以确认查证点失败的确切原因。

注意事项：你可以仅对包含查证点的测试脚本使用Comparators。

当你纪录包含查证点的测试脚本时，Robot创建一个包含捕获数据的Baseline file（基线文件）。

当你录制回放测试脚本时，Robot将Baseline file（基线文件）中的属性与“测试之下的应用程序（application-under-test）”的属性相比较和对照。如果比较失败，Robot保存引起此次失败的数据到一个Actual file（实际文件）中。该查证点的结果展现在测试日志中。

四种基本Comparators的如下

- Object Properties Comparator（对象属性比较器）--当你使用了Object Properties（对象属性）查证点时，可以使用Object Properties Comparator来查看和对照捕获的属性。
- Text Comparator（文本比较器）--当你使用了Alphanumeric（字符）查证点时，可以使用Text Comparator（文本比较器）来查看和比较捕获的字符（alphanumeric）数据。
- Grid Comparator（表格比较器）--当你使用了以下的查证点时：Object Data（对象数据），Menu（菜单），或者Clipboard（剪贴板），可以使用Grid Comparator（表格比较器）来查看和比较捕获的数据。

  Rational Quality Architect使用表格比较器来展现查证点信息。
- Image Comparator（图像比较器）--当你使用了以下查证点时：Region Image（区域图像）或者Window Image（窗口图像），可以使用Image Comparator（图像比较器）来查看和编辑位图。你也可以查看Unexpected Active Windows（异常活动窗口）。

启动比较器（Starting a Comparator）

从TestManager中启动一个Comparator：

1 点击**File > Open Test Log**。

2 展开包含日志的Build，双击该日志。

  为了在TestManager的Test Log窗口中打开一个Comparator，该日志必须包含针对这个特定Comparator的查证点。

3 点击Test Log窗口底部的**Details**标签。

4 在**Event Type**列中，点击加号（+），展开一个测试脚本并查看所有的查证点。

5 右键点击一个查证点并点击**View Verification Point**。

  打开针对这个特定查证点的Comparator，并展现该查证点。

如果查证点失败，Comparator 将伴随基线（Baseline）和显示的 Actual（实际）文件一起打开。从 Robot 中启动选择器，参阅 *Using Rational Robot* 手册。

对象属性比较器的使用（Using the Object Properties Comparator）

在一个Rational Robot测试脚本中使用Object Properties（对象属性）查证点时，使用对象属性比较器（Object Properties Comparator）来查看和比较捕获的属性。

你可以使用对象属性比较器（Object Properties Comparator）：

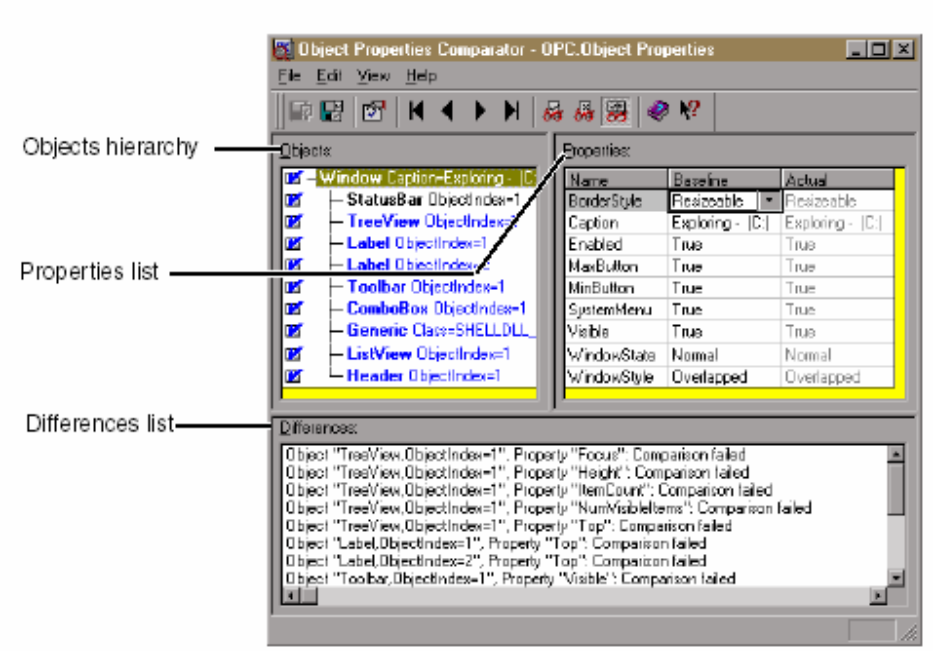
●细查，比较，和分析基线文件（Baseline file）与实际文件（Actual file）之间的差异。

●查看或者编辑针对一个对象属性（Object Properties）查证点的基线文件（Baseline file）。

从测试日志窗口中启动对象属性比较器（Object Properties Comparator），参阅182页*Starting a Comparator*的内容。

主要窗口（The Main Window）

对象属性比较器（Object Properties Comparator）的主要窗口包含对象（Objects）层次，属性（Properties）列表，和差异（Differences）列表。



对象层次（*Objects hierarchy*）包含了全部对象的列表，这些对象由Robot记录在对象属性（Object Properties）查证点中。属性列表（*Properties list*）包含了那些对象的属性的列表。当你选择了处在左边的对象时，它的属性会出现在右边。你可以控制窗口部分，将对象和属性两者都显示

出来，这就要通过**View**命令的使用。

差异列表（*Differences list*）展示了那些在基线与实际文件（Actual files）之间具有差异的对象。如果你在列表中的一个对象，那么这个对象会在对象层次与属性列表中被显著的标明。如果你正在查看一个未失败的文件，这部分将不被展现。展现或者隐藏这部分，可以点击**View > Show Difference List**。

对象层次和属性列表（The Objects Hierarchy and the Properties List）

当对象属性比较器被打开时，对象层次和属性列表如下展现：

●对象层次展现在窗口左边的窗格中。它展现了含有全部对象的列表，这些对象被Robot记录，它们使用对象属性查证点兵保存在基线文件中。

●属性列表展现在窗口右边的窗格中。它展现了被选择对象的属性列表，以及在基线文件和实际文件（如果存在差异的话）中的属性值。

如果查证点是通过的，Comparator仅仅将基线列和对象层次，以及属性列表一起展现。

如果查证点是失败的，Comparator会将基线列和实际列一起同对象层次，属性列表一起展现，这样，你就可以比较它们了。

注意事项：如果查证点仅包含一个对象，那么对象层次就不会展现了。当然，要展现它的话，你可以点击**View > Objects**或者**View > Objects and Properties**。

改变窗口Focus（Changing the Window Focus）

在对象层次已属性列表之间改变focus，可照以下的任意一种方法做：

●在该部分点击鼠标。

●按TAB键。

●按ALT+O键设置focus到对象层次。

●按ALT+P键设置focus到属性列表。

在对象层次中的操作（Working Within the Objects Hierarchy）

展现对象层次：

点击**View > Objects**或者**View > Objects and Properties**。

对象列表是层次化的。通过选择顶层对象，并使用**View > Expand**和**View > Collapse**命令，你可以展开或者折叠对象视图。

当你选择一个对象时，该对象的属性将展现在属性列表中。

每个对象根据它的对象类型被罗列，并以黑体标识。在对象命名之后，这里有一些诸如对象类或者索引的信息，它们常被用来指明对象。

如果对象是红色标识的，那么它在基线和实际文件中有不同的属性值。如果对象是蓝色标识的，那么它只存在于基线文件中。

你可以在对象层次中进行下面的任意一种操作。对象层次必须具有窗口focus。

●按HOME, END, PAGEUP, PAGEDOWN, UP ARROW, 和DOWN ARROW键来进行对象之间的移动。

●点击那些处在前面的每个对象的选择框，可针对测试进行选择和删除的操作。这些所有的具有复选框的对象都是被测试的。

●选择一个处在前面且具有复选标记的对象，在属性列表中展现它的属性。

●选择一个对象，按INSERT键，出现一个对话框，可以针对那个对象从属性列表中进行添加或者移除的操作。

●双击一个父类对象，可展开或者折叠它的子对象。

●按加号（＋）键，被显著标明的对象可展开一级，按减号（－）键折叠该对象。按星号（*）键,可展开所有的对象。

●右键点击在此层次中的一个对象，可展现该对象的快捷菜单。

●双击一个被标签为**Unknown**的对象，可定义该对象。有关在记录期间定义未知对象的信息，参阅*Using Rational Robot*手册。

在属性列表中的操作（Working Within the Properties List）

展现属性列表：

●点击**View > Properties**或者**View > Objects and Properties**。

名称列显示属性的名称。基线和实际列展现属性值。在基线列中的值表现的是从对象属性查证点的原记录而来的属性。

实际列中的值表现的是在最新的回放版本中的属性状态。按缺省，如果在基线和实际列中有差异，那么两者都会被展现。

使用**View**命令来控制在属性列表中的要展现的列。

如果属性是红色标识的，那么它在基线和实际文件中具有不同的值。如果属性是蓝色标识的，那么它仅存在于基线文件中。如果值单元为空，那么该属性具有一个空值。

你可以在属性列表中进行下面的任意一种操作。属性列表必须具有窗口focus。

●键入一个属性名称的第一个字母，移动那个属性或者第一个属性以该字母为开始。

●按HOME, END, PAGEUP, PAGEDOWN, UP ARROW, 和 DOWN ARROW键来显著标明一个属性。

●按INSERT键，出现一个对话框，可以从属性列表中进行属性添加和移除的操作。

●选择一个属性并按DELETE键，可从列表中移除它。

●右键点击一个属性的值单元，可编辑该值。

●定位列标题单元之间在垂直边界上的指针。向左或向右拖动该指针，可改变列宽。

●指向一个属性并点击鼠标右键，可展现该属性的快捷菜单。

当前基线的装载（Loading the Current Baseline）

装载当前基线文件：

●点击**File > Load Current Baseline**。

如果当前的基线文件已经被展现，这个命令将变得不可用。为了编辑一个基线，你必须查看这个当前基线文件。编辑操作可以包括创建一个mask，剪切，复制，粘贴，成倍复制，移动，或者删除masks，或者使用自动Mask特征。

当前基线是最新保存的基线文件，常被用来作为查证点比较的期望结果。通过Robot打开比较器时，在这个比较器中看到的就是当前基线文件。然而，一个更普遍的方法，当比较器是通过TestManager的Test Log窗口打开时，这个比较器可能展现的是历史的基线和实际文件。由于只有当前基线文件可以被编辑，如果你有历史的基线或者任意其他记录的基线显示，那么你将不能够使用任意一种编辑命令－－它们将不可用。你可以手工地强行当前基线文件通过使用这命令来被装载。

定位和比较差异（Locating and Comparing Differences）

对象属性比较器开始它的比较伴随着对象层次中的第一个对象和属性列表中的属性。

在基线和实际列表之间包含差异的对象是红色标识的。对象只存在于基线列表中，那么它是蓝色标识的。

在基线数据与实际数据之间定位第一个差异：

●点击**View > First Difference**。

当差异被定位时，失败是显著标明的。差异列表表明了失败的数量，并提供有关失败的原因。

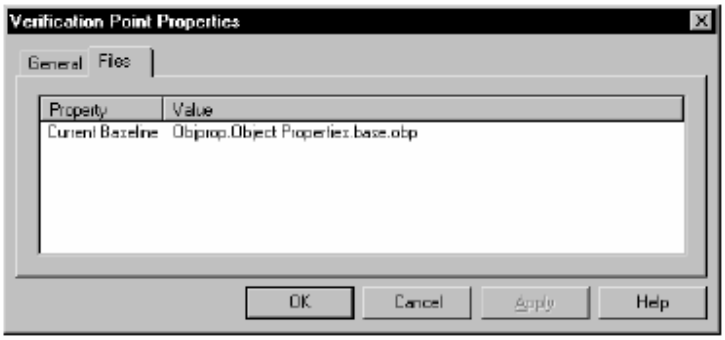
在差异之间进行导航，可使用**View**命令。

你也可以在差异列表中选择一个描述以显著标明那个在属性列表中的失败。

查证点属性的查看（Viewing Verification Point Properties）

查看查证点的属性：

查证点属性对话框展示了查证点的属性，基线文件的名称，以及实际文件的名称。



属性的添加和移除（Adding and Removing Properties）

当你首次创建一个对象属性查证点时，通过从属性列表中添加和移除它们，你可以说明该属性要进行测试。你也可以在你查看对象属性比较器的数据文件时从列表中添加和移除属性。这将使你在测试被创建和回放之后提炼它。

例如，如果对于一个查证点的属性列表包含了一个Height属性，你决定你不去测试，那么你可以移除比较器中的属性。你也可以为这个查证点，将列表中的属性应用于所有的具有相同类型的对象，并且为每一个对象定义默认属性的一个列表。

添加一个属性到属性列表：

●点击**Edit > Edit Property List**。

移除属性是指从属性列表中移除它，而非从查证点的基线文件从移除它。移除属性意味着它在今后的回放中不再被测试。一旦移除，属性可以在以后添加回来。

从属性列表中移除属性：

●点击**Edit > Remove Property**。

●如果你移除属性，那么你将它在晚些时候添加回属性列表，可通过使用**Edit > Edit Property List**命令。

基线文件的编辑（Editing the Baseline File）

当存在有意的变更到测试之下的应用程序时，你可能需要修改基线文件，以随着应用程序的开发保持它的更新。

编辑基线文件时，你可以：

●编辑属性列表中的值。

●剪切，复制，并粘贴值。

●从实际复制值到基线文件。

●变更查证的方法。

●变更确认的方法。

●替换基线文件。

注意事项：你不能编辑实际文件。

执行这些任务每个步骤的指令，可在对象属性比较器的帮助中搜索任务。

基线文件的保存（Saving the Baseline File）

保存变更到基线文件：

●点击 **File > Save Baseline**。

该命令仅当你有变更到基线文件时才有效。

文本比较器的使用（Using the Text Comparator）

使用文本比较器查看和比较捕获的字母数字数据，这是当你在一个Rational Robot测试脚本中使用字母数值查证点（Alphanumeric verification point）的情况下。

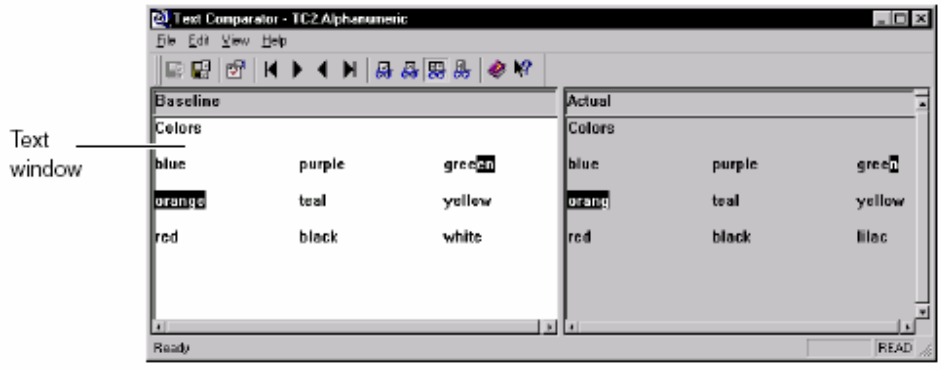
你可以使用文本比较器（Text Comparator）：

●复查，比较，和分析基线文件（Baseline file）与实际文件（Actual file）之间的差异。

●针对一个字母数字查证点，查看或者编辑基线文件。

主要窗口（The Main Window）

文本比较器的主要窗口包含了文本窗口。



文本窗口（The Text Window）

文本窗口有两个窗格：基线和实际。基线窗格显示的数据文件，为了比较而当作一个基线文件。实际窗格显示来自当前回放的数据。

文本窗口使用典型的文本编辑器格式。一般，你使用相同的规则和方法来输入，选择，和删除，这些你希望在标准的文本编辑器（例如笔记本）中使用的操作。

基线窗格具有白色的背景，实际窗格具有灰色的背景。当你使用一个定位命令来标明失败的数据时，它在基线文件和实际文件之间的比较以相反的颜色出现。

在文本窗口中，你可以：

●卷起文本窗口

●变更文本窗格的宽度

●使用字回绕

定位和比较差异（Locating and Comparing Differences）

在基线数据和实际数据之间定位首差异：

●点击**View > First Difference**。

要在差异之间导航，使用**View**命令。

比较开始在窗格较高的左边角。比较器扫描差异，按顺序通过文本的每一行，因为它会在文本编辑器中。

当比较器使用View命令发现发现差异时，在基线文件和实际文件之间，差异以相反的颜色显现。字母数字查证点保存指定的查证方法作为测试脚本命令的一部分。对于通过字母数字查证点创建的数据文件，比较器假定一个case-sensitive的比较，而不管它是如何被记录的。对于数字数据，比较器假定数字等价作为查证方法。

查证点属性的查看（Viewing Verification Point Properties）

要查看查证点的属性：

●点击**File > Verification Point Properties**。

查证点属性对话框显示了查证点类型，基线文件名称，以及实际文件名称。

基线文件的编辑（Editing the Baseline File）

当存在有意的变更到测试之下的应用程序时，你可能需要修改基线文件，以随着应用程序的开发保持它的更新。

编辑基线文件时，你可以：

●编辑数据。

●剪切，复制，和粘贴数据。

●从实际文件复制数据到基线文件。

●替换基线文件。

注意事项：你不能编辑实际文件。

执行这些任务每个步骤的指令，可在对象属性比较器的帮助中搜索任务。

基线文件的保存（Saving the Baseline File）

保存变更到基线文件：

●点击 **File > Save Baseline**。

该命令仅当你有变更到基线文件时才有效。

表格比较器的使用（Using the Grid Comparator）

使用表格比较器（Grid Comparator）来查看和比较捕获的数据，当你在Rational Robot测试脚本中使用下面的查证点时：

●对象数据

●菜单

●粘贴板

Rational Quality Architect也使用表格比较器来展现查证点信息。

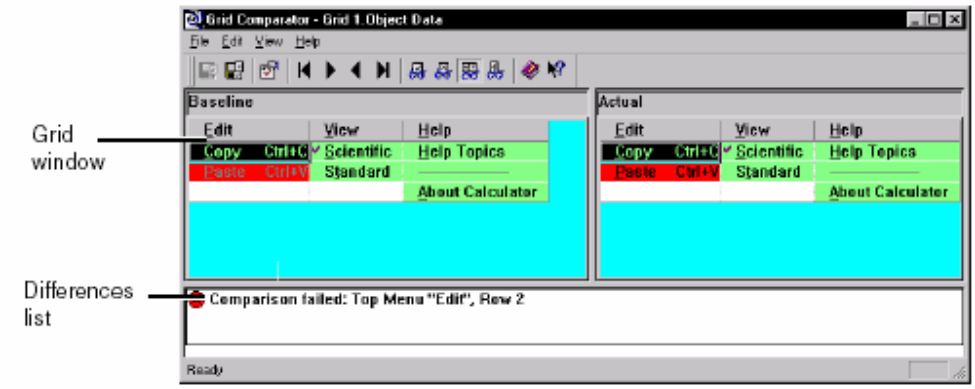
你可以使用表格比较器：

●复查，比较，和分析基线文件与实际文件之间的差异。

●查看或者编辑针对一个查证点的基线文件。

主要窗口（The Main Window）

表格比较器的主要窗口包含了表格窗口和差异列表。表格窗口包含了在对象数据，菜单，或者粘贴板查证点中记录的数据表格。差异列表展现了在回放期间失败的任意一项的描述。



差异列表（Differences List）

差异列表展现了在回放期间失败的实际项。这个列表显示查证点失败的原因，并展现图标以图形说明失败的类型。如果你点击列表中的一项，该项在表格中被标明。如果你在查看一个不具有差异的文件，那么这部分不会显示。

下面的这些图标会出现在差异列表中：

Icon	Meaning
	No differences found
	Comparison failed
	Item not found
	Different sizes
	Key not found

图标	含义
	没有发现差异
	失败的比较
	项未发现
	差异的大小
	键未发现

在差异列表中的操作：

●使用垂直滚动条来滚动描述的列表。

●选择差异列表中的描述，来标明基线和实际文件中的失败。

设置显示选项（Setting Display Options）

你可以在表格比较器中设置下面的显示选项：

●改变列宽。

●转置表格数据。

●使滚动条同步。

●使游标同步。

每个步骤的指令，可搜索在表格比较器帮助中的任务。

定位和比较差异（Locating and Comparing Differences）

定位首差异在基线数据和实际数据之间：

●点击**View > First Difference**。

在差异之间导航，可使用**View**命令。

在表格窗格中，比较以表格中的第一个数据单元（该单元在左上角处）开始。比较器扫描差异。在列的末端，比较器将从第二列的顶部开始，等等。

当一个差异被定位时，比较器标明差异的区域使用相反的颜色，并在差异列表中标明有关的描述。你也可以在差异列表中选择描述来标明基线和实际文件中的描述。

具有完整的被选择行和列的查证点在每个单元中比较数据，和行或者列中的单元数字是一样的。如果单元的数字不同，比较器标明行或者列，并用斜体字表示数字或者文本的标题。

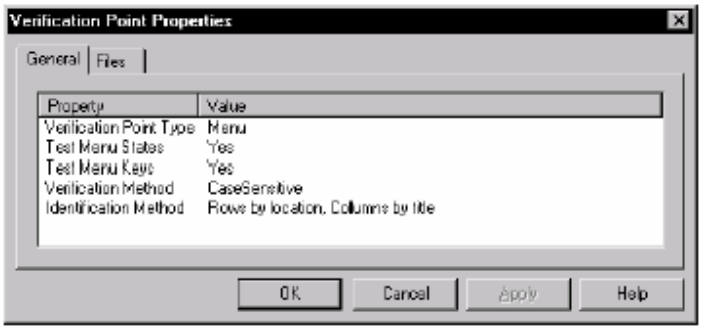
如果在表格中的数据展现比窗口大，那么你可以使用滚动条来查看其他的数据区域，或者你可以调整窗口的大小。

注意事项：如果差异在基线文件中被标明，并且在差异列表中的描述表示为“未发现项”（*Item cannot be found*），那么，它的意思是在实际文件中没有差异以标明，因此，该项在这里是不存在的。

查证点属性的查看（Viewing Verification Point Properties）

要查看查证点属性：

●点击**File > Verification Point Properties**。



使用键来比较数据文件（Using Keys to Compare Data Files）

当你创建对象数据或者创建在Robot中的查证点剪贴板时，你可以使用键/值确认的方法。

通过键/值确认的方法，对于具有行的查证点，你可以使用表格比较器来添加或者变更基线文件中键。在关系数据库中，键被使用，以唯一确认比较的行。

你可以添加或者变更键来确定在查证点中的重要的比较，并且可以变更一个失败的查证点到一个已通过的中去。

如果一个键列中的数据值发生变更，Robot将不会定位该记录，并且该查证点会失败。你可能希望改变比较器中的键，以获得更多的该查证点失败的原因查看。

如果你还没有指定唯一的键，那么测试会失败，因为Robot可能将选择的记录与包含了相似值的记录相比较，但是，这不是你希望测试的记录。你可以试验通过在比较器中变更键来增加查证点的可预言性。

如果数据库表发生变更，你可以改变比较器中的键来确定新的且唯一的列。

使用键来比较数据文件：

1 点击基线文件中一列的名称。

2 点击鼠标右键，或者按CTRL+K来添加或移除键。

基线和实际文件中的数据将自动地再次进行比较。

在这个时候，你可以估计新键的放置点。

如果基线文件中的键列有与实际文件不同的数据，那么差异列表显示为“未发现行”（Row not found）：行x并包含来自基线的键列的值（`Row x` and includes the value from the Baseline key column）。

如果基线中没有键列和行数据，并且实际文件不匹配，那么差异列表显示为“未发现行”（Row not found）：`Row where x` and includes each column name and value from the Baseline file。

基线文件的编辑（Editing the Baseline File）

当存在有意的变更到测试之下的应用程序时，你可能需要修改基线文件，以随着应用程序的开发保持它的更新。

编辑基线文件时，你可以：

●编辑数据。

●编辑一个菜单项。

●剪切，复制，和粘贴数据。

●从实际文件复制数据到基线文件。

●保存基线文件。

注意事项：你不能编辑实际文件。

执行这些任务每个步骤的指令，可在对象属性比较器的帮助中搜索任务。

保存基线文件（Saving the Baseline File）

保存变更到基线文件中：

点击**File > Save Baseline**。

该命令仅当你有变更到基线文件时才有效。

图象比较器的使用（Using the Image Comparator）

使用图象比较器来打开和查看捕获的位图，当你在Rational Robot测试脚本中使用下面的查证点时：

●区域图象

●窗口图象

你可以使用图象编辑器：

●复查并分析基线图象和实际图象之间的差异。

●通过在图象上创建masks编辑区域图象或者窗口图象的查证点。

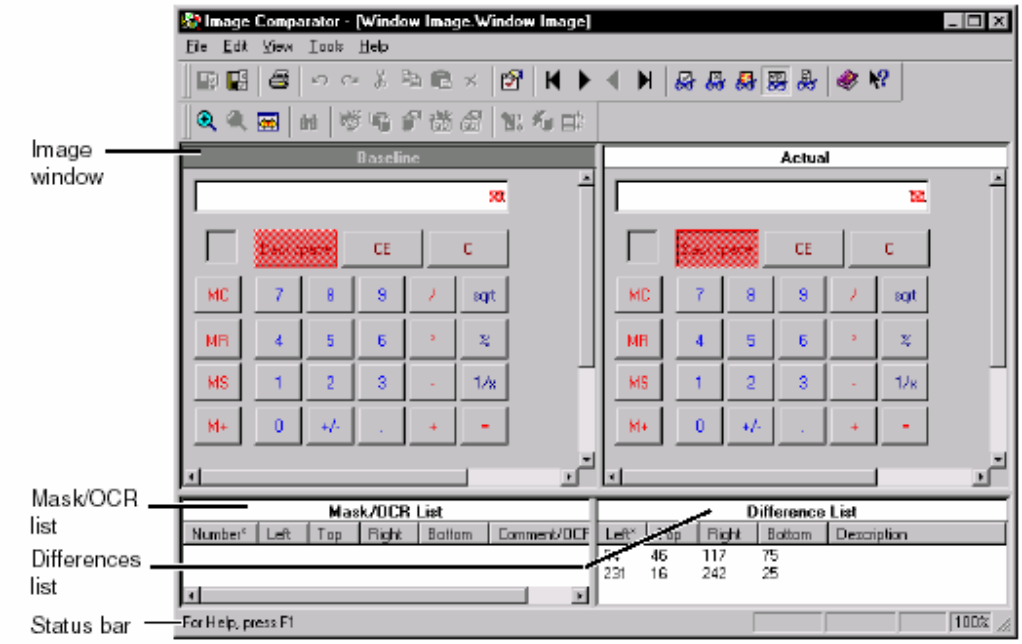
●创建OCR区域来读一个区域中的文本。

●查看异常活动窗口中的图象，它在一个测试脚本回放期间引发失败。

要启动图象比较器，参阅182页*Starting a Comparator*的内容。

主要窗口（The Main Window）

图象比较器的主要窗口包含了图象窗口，Mask/OCR列表，差异列表，以及状态条。



图象窗口（The Image Window）

图象窗口中有两个窗格：基线和实际。基线窗格显示的图象文件当作一个比较的期望文件。实际窗格显示当前回放的图象。通过使用View命令，你可以控制两个窗格的展现。

已通过的图象的部分，基线文件与实际文件之间的比较，当它们被记录时有完全地展现。已失败的图象部分，比较（差异）被显示为红色区域

你可以在窗格中的移动图象，并缩放该图象。有关信息，参阅202页*Moving and Zooming An Image*的内容。

差异列表（Differences List）

差异列表展现了在回放期间失败项的列表。“左”，“右”，“顶”和“底”的列说明了差异区域的大小，象素的数量。“左”列的数字说明从左边距到差异区域的左边界所具有的象素数目。“右”列的数字说明从左边距到差异区域的右边界所具有的象素数目。以相同的方式，“顶”列和“底”列定义了从顶边到差异区域的顶和底边界所具有的象素数目。

在差异列表中的操作：

使用垂直滚动条以滚动描述列表。

选择差异列表中的描述，标明基线和实际文件中的失败。

双击列表中的一项，将其放置，以使区域在视图的中央位置。它会有简短的闪动，并变为选择的。

差异列表是以列来分类的。当前的分类列是以星号标明的。要以不同的列分类，点击列标题。

该列表按选择列的升序进行分类。

Mask/OCR 列表（Mask/OCR List）

Masks用来隐藏来自比较的基本掩饰区域，当测试脚本回放时。包含mask的任意图象区域将不被比较，当你回放一个包含了图象查证点的测试脚本时。

Robot使用OCR区域来读设计区域中的文本，以及在随后的测试脚本回放中来比较它。

在主要窗口左下窗格中的Mask/OCR列表列出在查证点中使用的任意msaks和OCR区域。当你选择列表中的一个mask或者OCR区域时，它在基线和实际文件中被标明。该列表的操作与差异列表的操作具有相同的方法，作为前一部分的描述，差异列表。图象比较器的Mask/OCR列表部分，如果你不具有针对查证点定义的masks或者OCR区域，那么它是空的。

“左”，“右”，“顶”和“底”的列说明了masks或者OCR区域象素的数量的大小。

这些度量操作与差异列表中的操作有相同的方法。对于masks的注释列包含了可选的注释，你可以通过选择一个mask并点击**Edit > Mask Properties**来进行添加。OCR区域的OCR文本列包含了将被测试的区域中的文本。

状态栏（The Status Bar）

在主要窗口底部的状态栏提供你在比较器中操作的有用信息。要显示或者隐藏状态栏，选择**View >**

Status Bar。

状态栏左边部分的信息区域展现了菜单的命令描述以及操作信息，诸如在选择器为差异扫描图象时的更新过程。

状态栏的右边，有四个小窗格，以放置说明信息：

ReadOnly – 指明为一个只读状态。如果当前基线没有展现，由于当前基线是仅有的你可以编辑的文件，在这种情况下，该状态发生。

Load CBL – 指明当前基线不被展现。如果你希望开始编辑，点击**File > Load Current Baseline**来展现当前的基线。

BLINK – 指明闪烁特征被打开。

<zoom percentage> – 指明窗口的缩放百分比。如果你具有原型或者正常的视图，缩放百分比为100％。如果你缩放正常的视图，那么以该百分比显示视图。如果该图象符合该窗口，显示为*FITTED*（合适的）。

定位和比较差异（Locating and Comparing Differences）

●在基线和实际图象中展现差异：

点击**View > Show Differences**。

定位首差异：

●点击**View > First Difference**。

当一个差异被定位时，比较器对其有短暂地闪烁，把差异放在窗格的中央位置，然后在两个窗格中选择它。

在差异之间导航，使用**View**命令。

变更的差异如何被确定（Changing How Differences are Determined）

每个差异区域表现不同象素的一个逻辑集合—一串相异的象素一起关闭。依赖与你的首选设置，选择器确定该区域是否关闭，其中最后的一个属于相同的或者一个不同的差异区域。每次选择器定义新的区域围绕一个相异的象素，选择器确定该区域是否关闭，足够多的任意其他先前被定义的区域。

如果这样，比较器会合并这两个矩形区域。否则，该区域变为一个新的差异区域。

变更的差异如何被确定：

1 点击**Tools > Options**。

使用此设置来说明，当一个新的相异的象素被发现时，如何关闭。

2 在**Difference Regions**下变更设置。移动滑线来确定是否有更多的或更少的差异区域被创建。

当你移动该滑线时，紧邻着滑线的图片是那个选择的一个表示。

Masks，OCR区域，或差异的颜色变更（Changing the Color of Masks, OCR Regions, or Differences）

在图象窗口中变更masks，OCR区域，或者差异的颜色：

1 点击**Tools > Options**。

2 变更**Colors**的设置。

Masks – 在图象中为masks选择亮度颜色。Masks在基线和实际文件中被展现为该颜色块。默认颜色颜色是亮绿色。点击**Change**来选择不同的颜色。

Differences – 在图象中为差异选择亮度颜色。差异区域在基线和实际文件中被展现为该颜色块。默认颜色为亮红色。点击**Change**来选择不同的颜色。

OCR regions –在图象中为OCR区域选择亮度颜色。该区域在基线和实际文件中被展现为该颜色块。默认颜色为亮蓝色。点击**Change**来选择不同的颜色。

移动，缩放一个图象（Moving and Zooming An Image）

这里有几种方法，以在基线和实际窗格内移动图象：

●使用平行和垂直滚动条。如果你在查看两个文件，那么滚动是同步的。

●使用活动的手型指针。如果你在图象的任意地方（非mask或差异区域）压住鼠标左键，鼠标指针转变为手型。你可以使用它围绕这窗口移动图象。

注意事项：你可以使用缩放命令来转动图象。你可以拉近，拉远，按比例缩放，将图象与窗口完全匹配，或者返回到正常图象的大小。

●拉近图象，点击**View > Zoom > Zoom In**。

拉近图象通过两个因素。如果当你使用缩放命令时，你有一个选择的mask，OCR区域，或者差异区域，那么缩放集中于那个区域。如果你没有选择的区域，那么缩放集中在真个图象。你可以使用命令来反复地保持图象中的缩放。

●从图象拉远，点击**View > Zoom > Zoom Out**。

●通过百分比缩放图象的正常展现，点击**View > Zoom > Zoom Special**和百分比。

●恢复图象到它的原有尺寸，点击**View > Zoom > Normal Size**。

●将图象全尺寸符合窗格的大小，点击**View > Zoom > Fit To Window**。

缩放因素经常保留图象的高宽比，以确保文本和图象不出现变形。**Fit To Window**表示最大的缩放因素，在保持图象的高宽比的情况下，可以在窗口中展现完全的图象。

查看图象属性（Viewing Image Properties）

查看一个图象的属性：

●点击**File > Properties**。

图象属性对话框显示有关图象的信息，包括它的比例，颜色，尺寸和文件的创建数据。

有关Masks的操作（Working with Masks）

你可以使用**Edit > New Mask**命令创建图象比较器中的masks。

Masks在测试脚本回放时，被用来隐藏来自比较的基本的masked区域。任意包含mask的图象图象区域不作比较，在你回放一个包含了图象查证点的测试脚本时。

使用masks确保某个区域不进行测试。例如，如果你的应用程序有日期字段，你可能希望mask隐藏它以使其在每次测试脚本回放时不产生失败，你也可以应用masks来隐藏你决定的差异，通过有意的改变到应用程序中，以使它们在未来的测试中不引起失败。

因此你只能编辑基线文件，你不能在实际文件中执行下面的过程。然而，当你在一个基线文件中选择一个mask时，该mask也在实际文件中被选择。你不能修改实际文件中的mask—它只是为了方便而已。

你可以对mask做下面的操作：

●展现masks。

●创建masks。

●移动和调整masks。

●剪切，复制，和粘贴masks。

●加倍复制masks。

●删除masks。

●自动地mask（隐藏）差异。

每步指令，搜索图象比较器帮助中的每个任务。

有关OCR 区域的操作（Working with OCR Regions）

Robot使用Optical Character Recognition（光符识别）区域来读设计区域中的文本，并在随后的测试脚本回放中比较它。

你可以使用ORC区域来查证一个应用程序的正确操作，在窗口区域动态地画入文本，或者实际文本地获得是困难的。

OCR区域在一些的情况下也是有效的，比如一个文本字符串的字体或深度可能发生异常变更，但未发现使用了传统的查证方法。要取得正确的查证，你可以在现存的或新近捕获的查证点上定义OCR区域。

你可以对OCR区域进行下面的操作：

●创建一个OCR区域。

●移动和调整OCR区域。

●剪切，复制，和粘贴OCR区域。

●成倍复制OCR区域。

●删除OCR区域。

每步指令，搜索图象比较器帮助中的每个任务。

保存基线文件（Saving the Baseline File）

保存变更到基线文件中：

●点击**File > Save Baseline**。

该命令仅当你有变更到基线文件时才有效。

查看异常活动窗口（Viewing Unexpected Active Window）

Robot在测试脚本回放期间，设计为响应到异常活动窗口的。一个异常活动窗口，在测试脚本回

放期间，中断回放序列，并阻止产生活动的预期窗口，是任意非脚本窗口的出现。一个UAW的例子，一个错误消息产生，或者一个e-mail通知消息窗口。

你可以查看图象比较器中的异常窗口，仅当你已经在Robot中进行了选项设置。Robot的GUI回放对话框中，点击**Unexpected Active Window**标签。确认**Detect unexpected active windows**和**Capture screen image**选项都被选中。（有关设置异常活动窗口选项的信息，参阅*Using the Rational Robot*手册。）

在选择器中打开UAW以查看：

1 启动TestManager，打开含有一个UAW的日志文件。

2 在 **Event Type** 列中可进行下面的一个操作：

●双击一个异常活动窗口的事件。

●选择一个异常活动窗口事件，并点击 UAW。

图象比较器打开，这个 UAW 出现。

Part 3:

Performance Testing with Rational TestManager

性能测试的计划（Planning Performance Tests） 9

本章介绍使用 Rational TestManager 进行性能测试。它包括了下面的标题内容：

- 有关性能测试
- Rational TestManager 与性能测试
- 性能测试的计划
- 性能测试的实施
- 性能测试实例
- 性能测试的结果分析

有关性能测试（About Performance Testing）

性能测试帮助你确定，一个多客户（multi-client）系统在不同的工作负载和配置下是否执行到定义的标准。

性能测试是实施和执行的测试的一个类，来表征和评估被测试服务器如下的性能特征：

- 时间轮廓

- 吞吐量
- 响应时间
- 操作可靠性和界限

不同的性能测试类型，有其各自关注的一个不同的测试目的，它们实施于整个的软件开发生命周期。

在开发的生命周期初期—在加工迭代中—性能测试关注于确认和排除关联架构的性能瓶颈。

开发周期的后期—在构造迭代中—性能测试调试软件和环境（优化响应时间和资源），并查证应用程序和系统可接受地高工作负载的处理，以及压力状况，比如大量的事务处理，客户，或者大量的数据。性能测试的不同类型匹配与么个迭代。

性能测试包含了与多个虚拟测试者密切相关的服务器负载。例如，当有1000个其它的虚拟测试者在同一时间向相同的服务器发送请求时，你可能有一个关联了一个虚拟测试者的计时器，来查清在这种情况下服务器获得一个请求需要多长的时间。

你也可以使用虚拟测试者来衡量客户端的响应时间，获得一个客户端从发送请求到接受服务器响应时止需要等待的时间长度。当有效的客户端处理或者屏幕刷新时间关联与你要测量的一个真实用户的活动时，这里更多的表示为该用户的经验。

测试的类型（Types of Tests）

性能测试包含以下的测试类型：

- **基准测试（Benchmark tests）**--将一个新的或未知的服务器性能与已知的参照标准相比较，比如现存的软件或度量。
- **配置测试（Configuration tests）--**当操作的条件剩余常量时，查证服务器在不同配置下的可接受性能。
- **负载测试（Load tests）**--当操作的条件剩余常量时，查证服务器在不同工作负载下的可接受性能。
- **压力测试（Stress tests）--**当遇到反常或极端的状况时，比如降低资源或有最大用户数，查证服务器的可接受性能。
- **争用测试（Contention tests）**--查证服务器处理多个用户争用相同的系统资源（数据记录或内存）。

基准测试（Benchmark Tests）

基准测试可以提供关于服务器如何在给定的条件下执行的信息的一个基线。一个初始的基准度量可以提供来自于被评估的其他性能细节的一个参考点。

基准测试可以简单地确定在给定的时间总量中，完成一个操作的虚拟测试者的百分比，或者帮助你计算在这个相同的时间总量中，没有完成该操作的剩余虚拟测试者的百分比。

如果一个性能的基准对于给定的应用程序是确定的，那么你可以相比较这个基线来度量配置，负载，压力，以及争用测试的结果，以评估相对的性能。

配置测试（Configuration Tests）

现在，有着各种各样的C/S环境，每个用户的计算机都有着不同的硬件和软件配置，这样就会出现风险，应用软件将只能运行在其中的一些计算机上，而另外一些则不能。这时，你可以使用配置测试来确保你的产品将能在多种平台上运行。

TestManager让你计划和安排相同的测试以使应用程序在不同的代理计算机上运行，

可以依次使你：

- 兼容性测试的发布。
- 确定应用程序运行的最低和最适合的软，硬件配置。
- 确定应用程序在每台测试机上是如何执行的。

通常情况下，你只是和功能测试一起执行配置测试，当然，你也可以伴随性能测试负载你的C/S系统，以测试在工作负载上配置改变的效果。

负载测试（Load Tests）

负载测试是针对客户端或者服务器端在不同工作负载下的响应时间的测试。负载测试也被用来计算一个服务器在一个给定时间段内可以处理的最大事务处理数。此外，当一个C/S系统使用工作负载平衡或一个分布式结构时，负载测试可以帮助你确认作为设计的负载平衡或分布式方法的工作。

压力测试（Stress Tests）

压力测试是客户端应用程序在极端的条件下执行以查看客户端或服务器是否在“疲劳”的状态下会

发生档机的现象。压力测试的实例包括：

- 连续不断地将客户端的应用程序运行数小时。
- 执行大量的事务处理。
- 上百个虚拟测试者同时执行相同的操作或一个明确的操作组合。

其他的在一个系统上的压力测试类型包括了内存不足，无效的服务或硬件，或者在服务器端减少共享资源。

压力测试帮助你确认客户端应用程序或服务器端能处理的产品条件，无效的计算机资源管理可

能导致系统崩溃。

争用测试（Contention Tests）

争用测试包含虚拟测试者在一台或更多的测试机上执行，以模拟一个真实的用户环境。例如，你可能有虚拟测试者进入到相同的数据库，以发现诸如锁定，死锁条件，以及并发操作的问题。争用测试执行起来比较困难，原因是它需要在虚拟测试者之间进行准确的协调。利用TestManager，你可以进行多重测试机的测试，虚拟测试者在它们继续执行之前，等待条件以满足与其搭配的，或其他的测试机。例如，你可能在一台测试机上有一个虚拟测试者添加一个记录到数据库中，并且在另一台测试机上有第二个虚拟测试者，直到第一个虚拟测试者完成记录的添加方才暂停。第二个虚拟测试者可以读该记录。

本地和代理测试机（Local and Agent Computers）

你需要在单个的一台运行TestManager的Windows NT测试机上协调你所有的测试脚本活动。这就是所谓的*Local computer*（本地测试机）。从本地测试机上，你可以创建，执行，以及监控suites。在一个测试执行期间，你可以在本地测试机或你设计为*Agent computers*（代理测试机）的测试机上录制回放测试脚本。在下面的测试情况下，你使用一台代理测试机：

- **添加工作负载到服务器（Adding workload to the server）**—如果你执行一个具有大量虚拟测试者的测试，那么你可以使用代理测试机来添加工作负载到服务器。
- **在一台或多于一台的测试机上执行测试脚本（Running test scripts on more than one computer）**--如果你正在执行一个功能测试，你可以通过在下一个可用的代理测试机上执行该测试脚本来节约时间，该代理测试机替代本地测试机执行所有的测试脚本。对于这种情况，该测试脚本必须模块化和独立的。
- **测试硬件配置（Testing hardware configurations）**—如果你正在测试不同的硬件配置，你可以在不同的代理测试机上执行测试脚本，而这些测试机设置了这些硬件配置。

Suites

一般地，多个测试脚本和多个测试机是包含在一个测试当中的。在执行时，测试脚本的回放是通过你设计的测试suites来协调的。这些测试suites添加一个工作负载到该服务器。你从本地测试机上执行这些suites。

一旦你使用TestManager创建了suites，这个suites描述成服务器的一个基线特征，你就可以反复地执行这些suites来和产品的连续构造相比较，然后使用TestManager的报告工具来分析这些结果。

Rational TestManager 与性能测试（Rational TestManager and Performance Testing）

利用TestManager进行性能测试，可以帮助你在将应用程序部署到真实环境之前发现并改正性能方面的问题。通过TestManager，你可以得到你需要的所有的工具，来确定，隔离，并分析性能瓶颈。

作为一个自动负载测试工具，TestManager模拟一个或多个实际用户执行各种计算任务。由虚拟测试者替换用户，TestManager消除用户的需要来手工地添加工作负载到该服务器中。

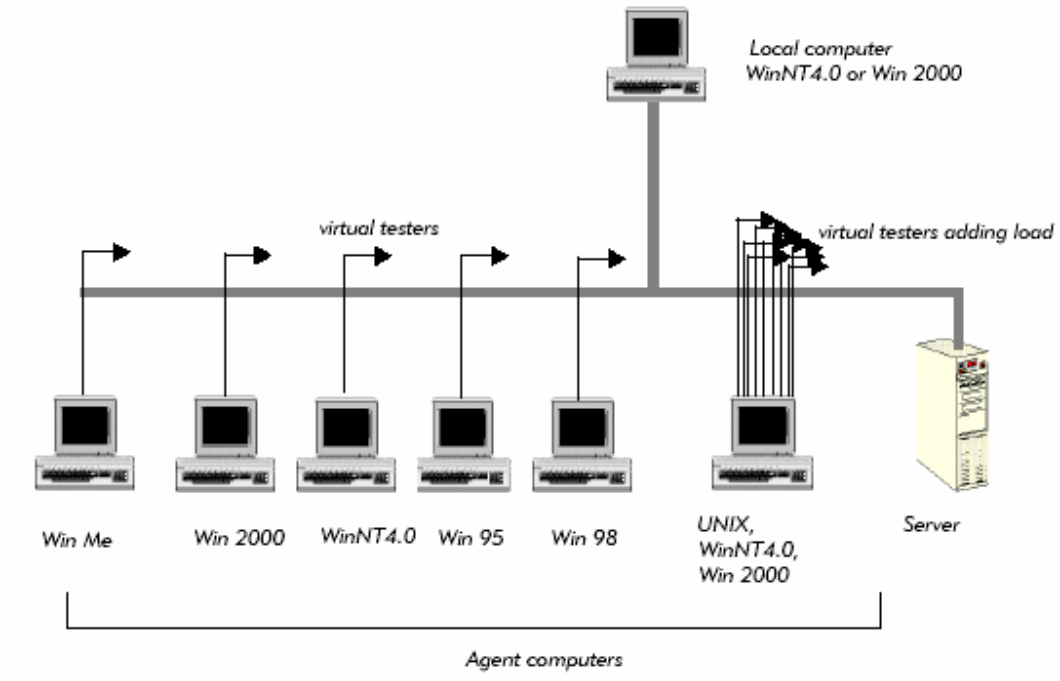
由于TestManager允许你在一台单独的测试机上回放多个虚拟测试者的活动，你可以在一些测试机—或一台测试机上，执行包含上百或上千个虚拟测试者的测试。

TestManager环境（The TestManager Environment）

TestManager能够使你在一个分布式的环境中执行suite。这个环境由单个的本地测试机（在该测试机上协调测试的执行和回放测试脚本），和零或更多台的代理测试机（在该测试机上回放测试脚本）构成。

服务器可以执行在多种操作系统下，并通过TCP/IP网络连接到本地和代理测试机上。

下图举例说明了一个典型的TestManager配置：



记录并回放测试脚本（Recording and Playing Back Test Scripts）

通过TestManager回放测试脚本来执行性能测试。一个测试脚本可以来自于许多的资源：

- 你可以利用Rational Robot来记录一个测试脚本。

  当你Rational Robot来记录一个脚本时，Robot在一个session中记录活动，然后自动地创建一个测试脚本，这个脚本代表了与服务器端的相互作用，以及所有的请求和响应。

  有关在Rational Robot中记录一个测试脚本的信息，参阅*UsingRational Robot*手册。
- 你可以使用任意的脚本语言来编写一个测试脚本。

  当你用此种方法提供一个测试脚本时，你必须写一个适配器以便TestManager能够识别出该测试脚本的类型。有关信息，参阅*Rational Test Extensibility Reference*手册。
- 你可以创建一个手工脚本，罗列一个虚拟测试者的任务。

性能测试的计划（Planning Performance Tests）

一个服务器的性能测试一般包含多个虚拟测试者的服务器负载。目的是要搞清楚在此工作负载下，服务器如何执行。

你可能希望回答一些性能方面的问题：

- 在正常条件下，服务器可支持多少虚拟测试者？
- 在正常条件下，遇到一些情况，服务器性能是否会突然地降低？
- 在你超出正常条件时，系统执行的如何？

  在一个“做最坏打算”的场景（scenario）中，系统性能是否大大地降低或完全地档机？
- 在多种硬件配置下系统执行的如何？

以下的章节将论述包含在一个测试计划中的这些关键步骤。

测试响应时间（Testing Response Times）

TestManager允许你度量多种性能指标。然而，分布式的功能测试按照直接的pass/fail响应度量其准确性，而性能测试还要度量响应的时间。例如：

- 一个活动的完全执行需要多长的时间？
- 在重工作负载条件下，服务器多快才能响应？

你也可以度量客户端的响应时间或服务器的响应时间，或者两者皆可。

为性能测试设置Pass 和Fail 的标准（Setting Pass and Fail Criteria for Performance Tests）

由于性能存在主观因素，确定要测试的性能特征和性能是否passes或 fails的标准就成为了必须要做的事。Passes或 fails的标准经常包含了一个响应时间的可接受的范围。

例如，你可以定义以下的作为一个可接受的响应时间：

对于100个虚拟测试者，所有事务处理的90%都有一个5秒或更少的响应时间的范围。响应时间最大不超过20秒。

对于500个虚拟测试者，所有事务处理的80%都有一个10秒或更少的响应时间的范围。响应时间最大不超过45秒。

确认测试性能的需求（Identifying Performance Testing Requirements）

当计划一个性能测试时，需要确定你测试要求的硬件和软件。例如：

- 服务器：数据库服务器，Web服务器，以及其他的服务器系统
- 客户机：Windows 2000, NT, 98, 95，或Me；网络计算机；或者Macintosh（苹果公司生产的一种型●号的计算机），以及UNIX工作站。
- 将要访问的数据库
- 将要运行的应用程序

此外，你需要以下的测试参数：

- 测试的数据库大小，以及其他的测试文档，以准确地代表实际的工作负载
- 分布的数据到达服务器以防止I/O的瓶颈。
- 如果你正在测试一个数据库，那么需要数据库关键参数的设置。

设计仿真的工作负载（Designing a Realistic Workload）

如果你正在测试性能，那么你的模型应准确地反映出站点的工作负载，这是必须达到的。因此，你必须确定在你的站点上发生的事务处理的类型。

例如，用户是否偶尔地查询和更新数据库，或者用户会经常地更新数据库？如果他们经常的更新数据库库，那么这些更新是复杂的，冗长的，或者是短暂的？

在设计工作负载时，考虑这些问题：

工作负载的时间间隔（The workload interval）--工作负载的时间周期的模型表现。

例如，工作负载的时间间隔可能是一个峰值时间，一个平均的天数，或者一个月末的记账周期。

测试变量（Test variables） --这是在性能测试期间将要改变的测试因素。

例如，你可能需要改变虚拟测试者的数量来了解在工作负载增加的情况下响音时间是如何退化的。

一次只改变一个变量是最好的做法。然后，如果性能伴随下一个测试而改变，那么你知道这个

变更是由这个变量引起的。

在你设置一个suite时，可以设置测试的变量。更多的信息，参阅67页*Implementing Tests as Suites*的内容。

虚拟测试者的分类（**Virtual tester classifications**）--你将虚拟测试者基于他们执行的活动类型分类到组内。

对于每组，你需要指定虚拟测试者的数量或总的虚拟测试者的百分比。例如，你可能把20%的虚拟测试者分组到报账（Accounting）中，30%的虚拟测试者到数据录入（Data Entry）中，以及50%的虚拟测试者到销售（Sales）中。

你在一个suite中设置用户组。然而，你应该保持这些用户组作为你计划的测试脚本，来关联该测试。这个测试脚本应当准确地反映出真实用户组的活动。有关设置用户组的信息，参阅227页*Inserting User Groups into a Suite*的内容。

用户工作的轮廓（User work profiles） --一个活动集，虚拟测试者执行的以及他们执行这些活动的频率。虚拟测试者的活动应当反映出尽可能接近用户实际操作的任务。

例如，如果Sales用户组访问数据库多出其他两个组的70%，那么要确保工作负载能够反映出这组活动的特点。

用户特征（**User characteristics**）--在执行一个事务处理之前，确定一个虚拟测试者暂停的时间长度，该事务处理被执行的速度，以及一个事物处理被不断地执行的次数。准确地建立一个真实用户的模型是重要的，因为这些值直接影响总的系统性能。

例如，一个用户考虑的时间为5秒，每分钟输入30个字符，那么对于一个考虑时间为1秒，每分钟输入60个字符的用户而言，第一个用户对于系统有一个更小的工作负载。

使用延迟和思考时间来对虚拟测试者建模。有关延迟的更多信息，参阅241页*Inserting a Delay*的内容。有关思考时间的内容，参阅*Using Rational Robot*手册。

在设计一个工作负载模型时，一定要这些因素，以确保有一个准确的测试环境。

仔细地考虑这些问题，将在一次长时间的执行中节约你的时间。更加清晰地定义你的测试目标，更加快捷的构建它们。

性能测试的实施（Implementing Performance Tests）

一旦你选择了pass和fail标准，硬件和软件需求，以及工作负载的模型，那么你就准备去创建测试脚本并设置此次测试吧。在该过程的这个阶段期间需要考虑的一些问题：

结束条件（The termination conditions）--如果一个虚拟测试者失败，应当停止该测试，还应当保持它的执行？

如果你实施了大量的虚拟测试者，并有一些失败了，一般情况下，该测试可以继续。然而，如果一个执行基本任务（如设置数据库）的虚拟用户失败，那么，该测试应当停止。

在suite中设置中止条件。有关设置中止条件的更多信息，参阅98页*Controlling How a Suite Terminates*的内容。

稳定的工作负载（The stable workload）--该测试应当等待，直到所有的虚拟测试者被连接，还是应当立刻启动此次测试的执行？

如果你试图度量虚拟测试者响应时间，那么你可能应当在实际的测试启动之前等待。

为了在性能报告中报告目标，你需要定义一个稳定的工作负载。

有关信息，参阅299页*Reporting on a Stable Load*的内容。

你将要测试的应用程序（The applications that you will test）--测试你全部的应用程序不是一件有“成本效益的”事情。消耗时间和资源测试一个对总体性能影响较小的应用程序，会造成对关键应用程序测试所应花费时间的损失。在计划和设计测试时，必须考虑到这个平衡。一般而言，你应当在你的系统上指定产生了80%的工作负载的那些20%的应用程序。例如，你可能不希望仅仅在一年的测试时间内包括一个更新数据库的应用程序的测试。

你将要执行测试的数据库（The database on which you will run the test）--决定你是否希望在成品数据库或一个测试数据库中执行测试。在当前使用的成品数据库中执行测试可以产生不正确的结果，作为正常用户工作负载（不包含工作负载模型）的影响。

性能测试实例（Examples of Performance Tests）

这一节将介绍一些典型的性能测试。每一个测试目标补充一个列表，列表中包含了在定义这样的测试时的关键元素。该表仅仅作为一个指南；这些不用作定义你可以包含在性能测试中所有可能的元素。

在正常条件下支持的虚拟测试者的数量（Number of Virtual Testers Supported Under Normal Conditions）

假设你希望确定虚拟测试者的数量，这些虚拟测试者是服务器可支持的，来确保系统能够符合你的可伸缩性的需求。在响应前，系统支持多少虚拟测试者是不可接受的？

例如，你估计一个数据库系统支持500个虚拟测试者。你可以计划并执行多任务的，具有300，

400，500，600，和700个虚拟测试者的当前性能的测试。下表包含了在你设计该测试时可能包含的关键元素：

Test Scripts	Suite	Reports
A test script to initialize the database. A test script to log virtual testers in. A test script for each virtual tester task: ▪ adding records ▪ deleting records ▪ querying the database ▪ running payroll reports	A fixed user group with one virtual tester. This virtual tester logs in, initializes the database, and sets an event indicating that the database is initialized. A scalable user group with many virtual testers. This group logs in and waits until the event is set. It then executes the scenario. A scenario that contains: ▪ a selector to randomly select a test script ▪ a test script for each virtual tester task	A test log to show whether all virtual testers in the suite successfully ran to completion. A Command Status report to show whether the server completed its requests successfully. Performance reports for each suite run: 300, 400, 500, 600, and 700 virtual testers. A Compare Performance report comparing the output of all five Performance reports.

测试脚本	**Suite**	**报告**
初始化数据库的测试脚本。 登陆虚拟测试者的测试脚本。 包含每个虚拟测试者任务的测试脚本： ●添加纪录 ●删除纪录 ●查询数据库 ●执行工资单报告	含有一个虚拟测试者的固定用户组。该虚拟测试者登陆，初始化数据库，并设置一个事件表明数据库已初始化。 含有多个按比例执行的虚拟测试者用户组。该组登陆并等待事件的设置。然后执行场景（scenario）。 一个场景包括： ●一个选择器随机地选择测试脚本。 ●含有每个虚拟测试者任务的测试脚本	一个测试日志，展现suite中的所有的虚拟测试者是否成功地的执行完毕。 一个Command Status（命令状态）报告，展现该服务器是否成功地完成它的请求。 Performance（性能）报告，每个suite执行：300，400，500，600，和700个虚拟测试者。 一个Compare Performance（比较性能）报告，比较所有的五个（Performance）性能报告的输出。

逐渐地增加虚拟测试者（Incrementally Increasing Virtual Testers）

在性能测试中的一个普通的需求要模型化，在不同的虚拟测试者执行它们工作的整个一个时间

跨度时将发生什么。例如，假设你希望测试服务器在早晨时的性能情况。你也希望知道服务器如何处理在一天之中增加的工作负载，以及特别的工作负载的峰值出现在何时。

利用TestManager，你可以模型化这些通过逐渐增加负载虚拟测试者的工作负载的类型。你可以通过该工作负载的模型开发，启动你希望的测试。例如，记录下应用程序的使用频率和容量。

然后，在设置你的suite时：

1 安排不同的用户组在不同的时间启动，覆盖suite的生命期。

2 对于每一个用户组，设置虚拟测试者的数量，执行测试脚本和迭代计数（可选），以适合于你的测试。

通过将你的虚拟测试者的启动时间和迭代计数分层化，你可以构造逐渐增加的负载。你也可以在suite执行中的特定时间内添加负载的峰值。

以下描述了一个代表重叠转移的测试实例：

- 你启动一个含有100个虚拟测试者的suite。该组的虚拟测试者需要对交易重复进行，因此你应该设置每个虚拟测试者来执行该交易的多次迭代；你应该设置足够的迭代来保持这一组虚拟测试者的执行，直到这个suites结束。你可以做个实验，来确定你需要多少个迭代。
- 通过suite，设置一个Delay（延迟）。Delay Type可能是从suite一开始起，或是一天的某个时刻的开始起。当延迟结束时，200个新的测试虚拟者开始。这是下一个重叠了首次转移的entry clerks转移。
- 在代表负载峰值合并的shift期间，300个虚拟测试者重复的执行交易。

下表总结了一个重叠shifts的测试实例：

Test Scripts	Suite	Reports
A test script to initialize the database. A test script to log virtual testers in. A test script for each virtual tester task: ▪ adding records ▪ deleting records ▪ querying the database ▪ running payroll reports	A fixed user group with one virtual tester. This virtual tester logs in, initializes the database, and sets an event indicating that the database is initialized. A fixed user group with 100 virtual testers. Each virtual tester logs in and waits until the event is set. Each virtual tester then executes many iterations of the scenario. A fixed user group with 200 virtual testers that delays for 2 hours. Each virtual tester then logs in, checks that the event is set, and executes many iterations of the scenario. One scenario that contains: ▪ a selector to randomly select a test script ▪ a test script for each virtual tester task	A test log to show whether all virtual testers in the suite successfully ran to completion. A Command Status report to show whether the server completed its requests successfully. Two Performance reports: ▪ One report on the time period from the start of the run until 2 hours have passed. ▪ One report on the time period from 2 hours until the end of the run. A Compare Performance report comparing the output of both Performance reports.

Test Scripts	Suite	Reports
一个初始化数据库的测试脚本。 一个登陆虚拟测试者的测试脚本。 针对每个虚拟测试者任务的测试脚本： ●添加记录 ●删除纪录 ●查询数据库 ●执行工资单报告	含有一个虚拟测试者的固定用户组。该虚拟测试者登陆，和初始化数据库，并设置事件指明该数据库已被初始化。 含有100个虚拟测试者的固定用户组。每个虚拟测试者登陆并等待，直到事件被设置。每个虚拟测试者随后执行该场景的多次迭代。 含有200个虚拟测试者的固定用户组，该组有2个小时的延迟设置。每个虚拟测试者登陆，和检查事件被设置，并执行该场景的多个迭代。 一个场景包括：	一个测试日志，展现suite中的所有的虚拟测试者是否成功地的执行完毕。 一个Command Status（命令状态）报告，展现该服务器是否成功地完成它的请求。 两个Performance reports（性能报告）： 一个是在从开始执行到2个小时（延迟）通过的时间段内的报告。 一个是在从两个小时（延迟）到执行结束的时间段内的报告。 一个Compare

	●一个选择器，可随机地选择测试脚本。 ●含有每个虚拟测试者任务的测试脚本。	Performance（比较性能）报告，比较两个 Performance reports（性能报告）的输出。

在压力条件下系统的性能如何（How a System Performs Under Stress Conditions）

压力测试可以被理解为在不间断的攻击下以引起服务器档机。在你怀疑服务器存在一些不牢固的区域时，使用压力测试，当一个操作被大批的执行或经过一个长时间执行时，可以完全地中断或者减弱服务器响应。

压力测试包含多个同时操作（比如在相同的时刻向服务器发送上百条的请求），而虚拟测试者则提供了执行这类压力测试的最实际和有效的手段。连续不断地执行测试脚本，帮助你了解应用程序在压力条件下执行的“长时”效果。

在一个简单的压力测试中，你可以创建一个测试，这个测试中的虚拟测试者不断和重复好几个小时地执行相同的操作。你的测试可能包含：

- 插入上千条的纪录到数据库中。
- 发送上千条的查询请求到数据库。

下表总结了一个压力测试的实例：

Test Scripts	Suite	Reports
A test script to initialize the database. A test script to log virtual testers in. A test script for each virtual tester task: ▪ adding records ▪ deleting records ▪ querying the database ▪ running payroll reports	A fixed user group with one virtual tester. This virtual tester logs in, initializes the database, and sets an event indicating that the database is initialized. A scalable user group with 1000 virtual testers. Each virtual tester logs in and waits at a sync point. When all the virtual testers are synchronized, each virtual tester executes many iterations of the scenario. One scenario that contains: ▪ a selector to randomly select a test script ▪ a test script for each virtual tester task	A test log to show whether all virtual testers in the suite successfully ran to completion. A Command Status report to show if the server behaved correctly, even under stress. Performance reports for each suite run: 900, 1000, and 1100 virtual testers. These Performance reports show when the system starts to degrade, and ensure that the degradation is graceful. A Compare Performance report comparing the output of each Performance report.

Test Scripts	Suite	Reports
一个初始化数据库的测试脚本。 一个登陆虚拟测试者的测试脚本。 针对每个虚拟测试者任务的测试脚本： ●添加记录 ●删除纪录 ●查询数据库 ●执行工资单报告	含有一个虚拟测试者的固定用户组。该虚拟测试者登陆，和初始化数据库，并设置事件指明该数据库已被初始化。 含有1000个虚拟测试者的按比例执行用户组。每个虚拟测试者登陆并在一个同步点处等待。当所有的虚拟测试者被同步时，每个虚拟测试者执行该场景的多个迭代。 一个场景包括： ●一个选择器，可随机地选择测试脚本。 ●含有每个虚拟测试者任务的测试脚本。	一个测试日志，展现suite中的所有的虚拟测试者是否成功地的执行完毕。 一个Command Status（命令状态）报告，展现在压力条件下服务器行为是否正确。 Performance（性能）报告，每个suite执行：900,1000,和1100个虚拟测试者。这些性能测试报告展现系统性能何时开始退化，以及确保这种退化是舒缓的。 一个Compare Performance（比较性能）报告，比较每个Performance reports（性能报告）的输出。

不同的系统配置如何影响性能（How Different System Configurations Affect Performance）

因为suite是被组织和执行的之一特点，TestManager给与自己良好的配置测试。你可能由于各种各样的原因需要进行配置测试。

例如：

- 你希望测试你的系统在具有更多（或者更少）的内存条件下如何运行。
- 你希望测试你的系统在具有不同的硬盘空间下条件如何运行。
- 你希望找到具有系统最优运行的那个网卡。

下表总结了一个含有100个虚拟测试者的配置测试：

Test Scripts	Suite	Reports
A test script to initialize the database. A test script to log in virtual testers. A test script for each virtual tester task: ▪ adding records ▪ deleting records ▪ querying the database ▪ running payroll reports	A fixed user group with one virtual tester. This virtual tester logs in, initializes the database, and sets an event indicating that the database is initialized. A fixed user group with 100 virtual testers. Each virtual tester logs in and waits until the event is set. Each virtual tester then executes many iterations of the scenario. One scenario that contains: ▪ a selector to randomly select a test script ▪ a test script for each virtual tester task	A test log to show whether all virtual testers in the suite successfully ran to completion. A Command Status report to show if the server returned expected responses, even under stress. Performance reports for each suite run on each configuration. A Compare Performance report comparing the output of each Performance report.

测试脚本	**Suite**	**报告**
初始化数据库的测试脚本。 登陆虚拟测试者的测试脚本。 包含每个虚拟测试者任务的测试脚本： ●添加纪录 ●删除纪录 ●查询数据库 ●执行工资单报告	含有一个虚拟测试者的固定用户组。该虚拟测试者登陆，初始化数据库，并设置一个事件表明数据库已初始化。 含有100个虚拟测试者的固定用户组。每个虚拟测试者登陆并等待事件的设置。然后每个虚拟测试者执行该场景（scenario）。 一个场景包括： ●一个选择器随机地选择测试脚本。 ●含有每个虚拟测试者任务的测试脚本。	一个测试日志，展现suite中的所有的虚拟测试者是否成功地的执行完毕。 一个Command Status（命令状态）报告，展现该服务器在压力下是否返回了期望的响应。 执行在每个配置上的每个suite的Performance（性能）报告。 一个Compare Performance（比较性能）报告，比较每个（Performance）性能报告的输出。

性能测试结果分析（Analyzing Performance Results）

TestManager生成大量的测试数据，而且在第一次，大量的完全数据可能是覆盖的。然而，如果

你仔细地计划了你的测试，那么你应该确定哪些数据对你来说是重要的。

首先，你应该检查你的数据在统计上是有效的。为此，执行有关你的测试数据的一个Performance报告和Response vs. Time报告。

注意事项：在suite成功执行的后期，TestManager会自动地执行Performance和Response vs. Time报告。

Performance报告包含了两列内容：平均数和标准偏差。如果平均数少于3次的标准偏差，那么你的数据对于有意义的结果而言可能就过于分散了。

Response vs. Time图展现了相对于执行经过时间的响应时间。该数据应该是在达到一个稳恒态的性能下取得，而不是在更好或更糟的性能条件下步进地获得。如果该响应时间是在更好的性能下步进地获得的，那么你可能就要将登陆时间也包含在你的测试结果中，而不是去衡量一个稳定的工作量。或者在你数据库中访问这些数据的总量可能少于现实的数据，结果是所有的访问在缓存中被平衡。

在你确认了你的实例是有效的之后，开始分析你的测试结果。当你分析结果时，使用一个多层次接近（approach）。例如，如果你驾车从一个城市到另外一个，你希望使用一张美国地图来计划一个总的行车路线，并且需要一张更详细的城市地图来获得你的目的地。

同样地，在你分析你的测试结果时，你应当首先在一个宏观层上开始，然后在移动到一个更为细节的层次上。

下一节总结了你可以使用的来分析测试结果的这些不同的细节层次。有关性能测试报告的更多信息，参阅291页上*Reporting Performance Testing Results*的内容。

多重执行的结果比较（Comparing Results of Multiple Runs）

分析的第一层包含对单个suite执行结果的图形摘要信息的评估，然后比较通过多重执行的结果。例如，检查针对单个虚拟测试者或在单个的suite执行期间进行业务处理的响应时间分布。然后比较平均响应时间，这些响应时间通过多重执行，且具有不同数量的虚拟测试者。

第一层的分析让你确知你的性能目标是否已基本满足。它帮助你确认在数据中的趋向，并强调在什么地方会出现性能问题—例如，性能可能在250个虚拟测试者时有相当程度的退化。

要得到该类型的分析结果，执行Performance和Compare Performance报告。

具体请求和响应的比较（Comparing Specific Requests and Responses）

分析的第二层包含要检查的统计概要，以及针对具体虚拟测试者的请求和系统响应的实际数据。概要统计包含了响应时间的标准偏差和百分比分布，它表明通过单个的虚拟测试者系统的响应如何变化。

例如，如果你测试一个SQL数据库，你可以跟踪具体的SQL请求和相应的响应来分析出现的性能问题和潜在的引起性能退化的原因。

针对第二层的分析，你可以：

1. 通过执行Response vs. Time报告和获取的两个timestamp，来确定一个稳定的量度间隔。第一个timestamp发生在虚拟测试者从开始阶段的任务退出时，它是最后一个虚拟测试者的timestamp，该虚拟测试者开始进行的“实际”工作：添加纪录，删除纪录，及其他等。第二个timestamp是首个虚拟测试者的timestamp，该虚拟测试者退出系统。你现在就可以确定一个稳定的量度间隔。
2. 使用仅仅是由这两个timestamps确定的间隔来创建一个Performance报告。
3. 描绘Performance报告图，来查证该分布已经变平了。
4. 执行Performance ，Compare Performance，和Command Usage报告来检查针对该量度间隔的概要统计。

性能问题的原因确定（Determining the Cause of Performance Problems）

第三层的分析帮助你了解性能问题的原因和意义。

统计结果的分析（Analyzing Results Statistically）

详细的分析获得低层数据并使用统计的测试来帮助你获得有用的结论。虽然该分析提供客观的和定量的标准，但比起第一层和第二层的分析会消耗更多的时间，并且需要具备一定的统计学基础。

在这一层上分析你的测试数据时，你可以使用统计学（*statistical significance*）的概念来帮助识别，在不同响应时间中是否是真实的，或者识别一些关联与测试数据收集的随机事件。在一个基本层次上，随机性是关联与任何一个事件的。统计的测试确定是否存在一个系统的差异，而这些差异不能通过随机事件来解释。如果该差异不是通过随机性产生的，那么该差异在统计上是有意义的。

要执行第三层的分析，执行Performance和Response vs. Time报告。

以下是在第三层分析期间需要考虑的一些量度：

最小值—最低响应时间。

最大值—最高响应时间。

平均值—平均响应时间。该平均值的确定，是通过计算所有响应时间值的和，再除以响应时间数量之后得到。

中间值—数据的中点。二分之一的响应时间值小于该点，并且它们的二分之一大于该点。

标准偏差—围绕平均值的测试数据是如何被分组的。

百分比—在某一点之上或者之下的响应时间的百分比。一个经常的百分比是90%。

外层—测试数据中的一个高于或低于其他值的数据。

例如，系统A和系统B都具有一个12毫秒的平均响应时间。这不是必需的平均值，系统的响应是相同的。进一步的结果评估显示，系统A的响应时间有11，12，13，和12，且系统B的响应时间有1，20，25，和2。虽然他们的平均响应时间是一样的，但他们的最小值，最大值，以及标准偏差是完全不同的。

检测计算机资源并协调你的系统（Monitoring Computer Resources and Tuning Your System）

性能问题可能是由你的服务器只具备有限的硬件资源引起的而非软件设计本身。例如，你的硬盘工作的服务时间慢的不能接受，这是因为一个集中的硬盘传送被发送到一个单一的硬盘，而不是分布到几个硬盘机上的缘故。这个问题典型的扑救方法是，在少量的活动下，通过重新安置一些频繁存取的文件（比如交换文件或临时文件）到一个硬盘。

性能问题也可以由超载局域网（LAN）的部分（segments）或路由器引起，结果是加剧了网络拥塞。甚至是最简单的从客户端到服务器的来回延迟和回退带了的那几秒钟。这个问题典型的扑救方法是，通过分担一个超载的局域网部分到两个或者三个部分，且他们之间由路由器互联。有时，你需要添加第二块网卡到服务器中，以便他们可以不通过路由器直接进入到两个局域网部分。

这些硬件方面的局限会减缓响应时间量度，这不是能通过改变软件设计可解决的。

TestManager可以使你匹配CPU，内存，以及硬件利用规格，这些都是在具有虚拟测试者响应时间数据的情况下。你可以在一个suite回放期间检测你的计算机资源利用率，然后对这个利用率数据作图覆盖相符合的虚拟测试者的响应时间，来确定不平衡的硬件资源是否造成了响应时间的减缓。

有关执行Response vs. Time报告的更多信息，参阅301页*Mapping Computer Resource Usage onto Response Time*的内容。

性能测试 suite 的设计（Designing PerformanceTesting Suites） 10

本章描述如何设计性能测试的 suites。它包含了以下主题：

□关于 suites

□由 Robot session 创建 suite

□插入用户组到 suite 中

□插入测试脚本到 suite 中

□在测试脚本，测试用例或者 suite 中设置前置条件

□插入其他项到 suite 中

□使用事件和依赖协调执行

□Suites 的执行

关于 suites（About Suites）

Suite用层次来展现你希望执行的工作量。比如，用户组，每个用户组中的用户数量，以及在这样的用户组中执行的测试脚本，和每个脚本的执行次数。

通过suite，你可以：

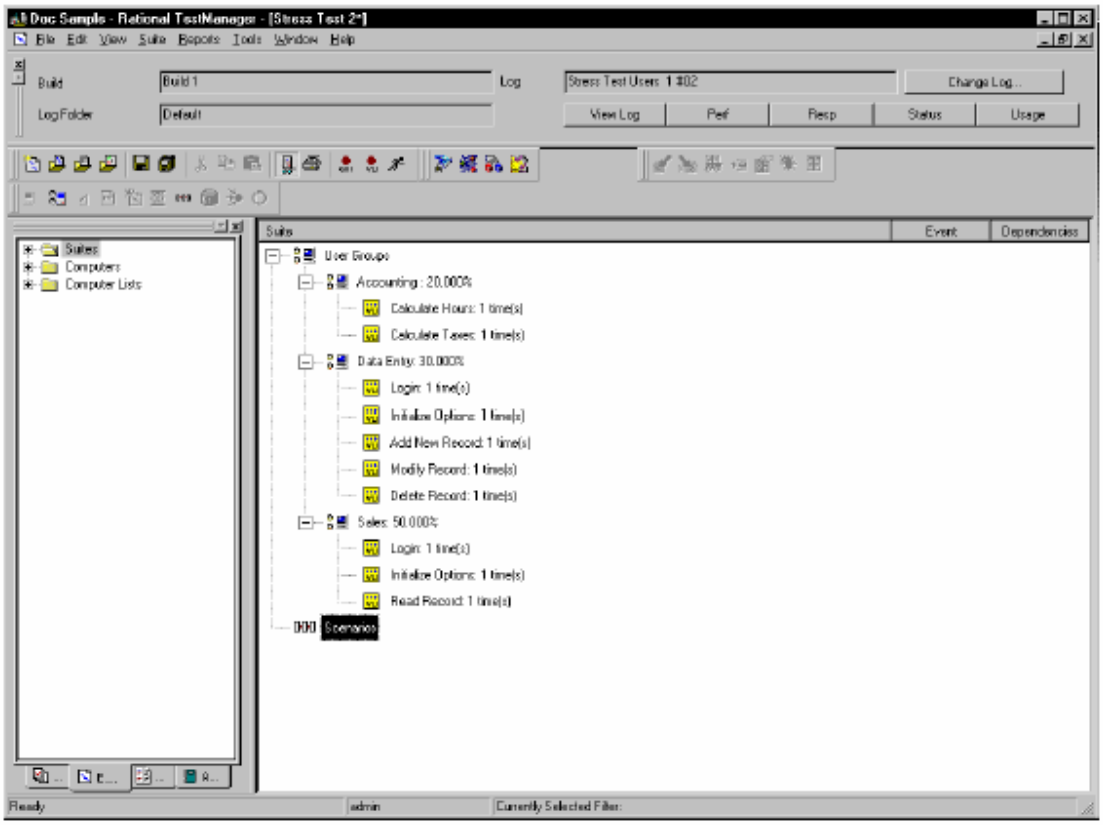
●执行测试脚本和测试用例。

●将测试脚本分组，模拟不同类型虚拟用户的活动。

●设置测试脚本执行顺序。

●使虚拟用户同步执行。

下面的这个简单的suite中展现了三个用户组：Accounting,，Data Entry，和Sales。



在这个suite中：

●Accounting用户组执行两个测试脚本：一个计算工资单小时数，一个计算工资单的税额。

●Data Entry用户组执行五个测试脚本：登陆，初始化数据库选项，以及三个变更数据库记录。

●Sales用户组执行三个测试脚本：登陆，初始化数据库选型，以及读取数据库记录。

注意事项：在这一章的suites中包含VU测试脚本，用来进行性能测试。Suite中，也包含GUI脚本，VB脚本，或其他的用户自定义类型测试脚本。

由 Robot Session 创建 suite（Creating a Suite from a Robot Session）

如果你在Robot中有一个记录的session，那么你可以通过TestManager回放在这个session中的测试脚本。

当你由session创建suite，并执行这个session时，你会把执行这个session期间所记录的所有的客户端/服务器请求，有顺序地记录下来。

由session创建的suite顺序地记录测试脚本并维护它们。如果你希望保留这个顺序，就由session来创建该suite。如果你希望在suite中更加灵活的控制脚本，那么你就可逐个添加测试脚本。

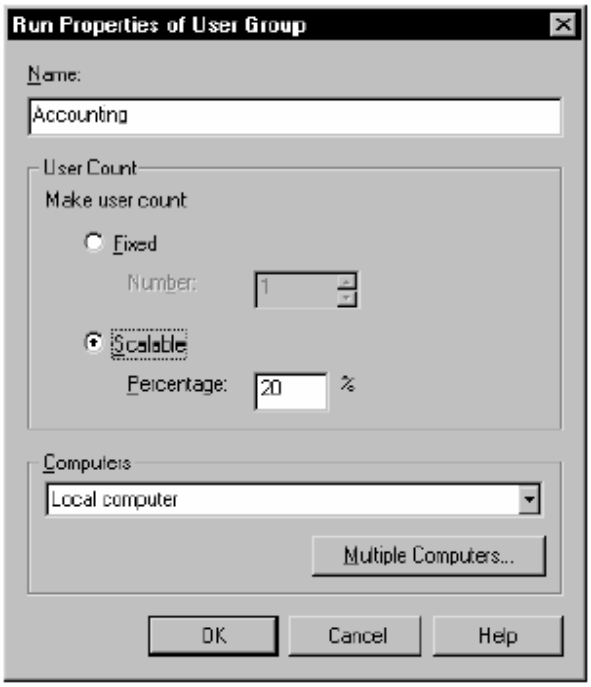
你也可以从头创建suite或基于一个现有的suite来创建suite。更多有关创建suites的信息，参阅73页*Creating a Suite*的内容。

插入用户组到 Suite 中（Inserting User Groups into a Suite）

User group（用户组）是性能测试suites中的基本构件。一个用户组是执行相同活动的虚拟测试者的一个集合。例如，上面的那个suite中包含了3个用户组：Accounting，Data Entry和Sales。

插入一个用户组到suite：

●在一个打开的suite中，点击**Suite > Insert > User Group**。



当添加一个用户组到suite中时，必须要说明该用户组包含的虚拟测试者的种类，是固定测试者还是按比例执行的虚拟测试者：

- 固定的（**Fixed**）——准确地说是一批静态的虚拟测试者。输入所希望执行的最大数量的虚拟测试者。例如，如果输入50个虚拟测试者，在每次执行suite时，Sales组中执行的虚拟测试者的数量可达到50个之多。

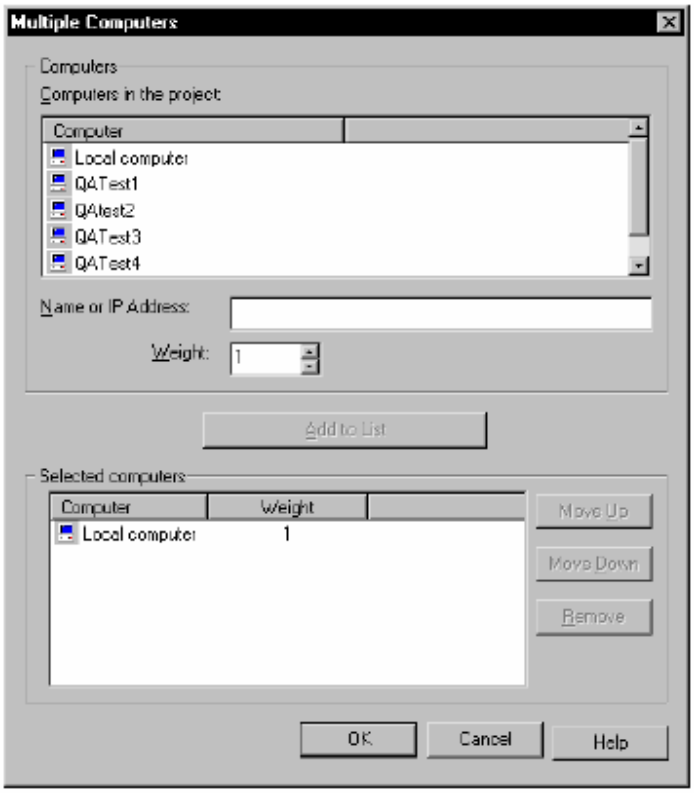
  通常情况下，分配固定数量的虚拟测试者到一个不添加工作负载的用户组中。如，一个虚拟测试者可能执行Warmup测试脚本来打开数据库，而另外一个虚拟测试者可能执行Shutdown

测试脚本来保存并关闭该数据库。

- 按比例执行（**Scalable**）——准确地说是一批动态的虚拟测试者。输入该用户组所代表的工作负载百分比。如，Accounting组可能代表了20%的虚拟测试者，Data Entry组可能代表了30%的虚拟测试者，而Sales组可能代表了50%的虚拟测试者。每次执行suite，明确将要执行的虚拟测试者的总量；TestManager按照你要求的百分比在这些按比例执行的用户组中自动分配虚拟测试者。

在多台测试机中分配用户组内的虚拟测试者：

- 点击**Suite > Insert > User Group**，然后点击**Multiple Computers**。



当定义一个用户组时，必须说明该用户组执行的测试机。默认的测试机为TestManager的本地测试机（Local computer），该用户组也可以在任意定义好的代理测试机（Agent computer）上执行。

一般地，在以下的情况下用代理测试机执行（Agent computer）用户组：

- 性能测试需要特殊的客户端库（libraries），或功能测试需要指定代理测试机上的软件配置。该用户组必须在已安装库或者软件的测试机上执行。
- 为特殊测试机设计的功能测试。

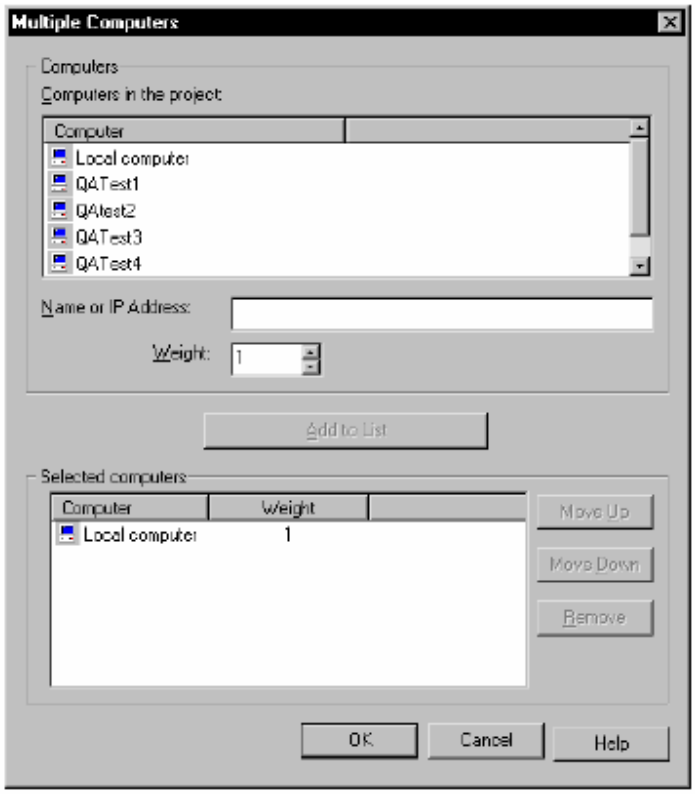
注意事项：复制任意用户自创建的扩展C库，Java类文件，或者COM组件，对设置代理测试机的测试是必要的。

在多台测试机中分配虚拟测试者。通常在如下情况下，在多台测试机上执行一个用户组：

- 功能测试必须尽可能快地执行。通过在不同的测试机上同时执行虚拟测试者，可以节约时间。
- 有大量的虚拟测试者，并且本地机（Local computer）不具备足够的CPU或内存资源以支持这些工作负载。通过在每台分布的测试机上执行较少的虚拟测试者，可以节约资源。

在多台测试机中分配用户组内的虚拟测试者：

- 点击**Suite > Insert > User Group**，然后点击**Multiple Computers**。



插入测试脚本到 Suite 中（Inserting Test Scripts into a Suite）

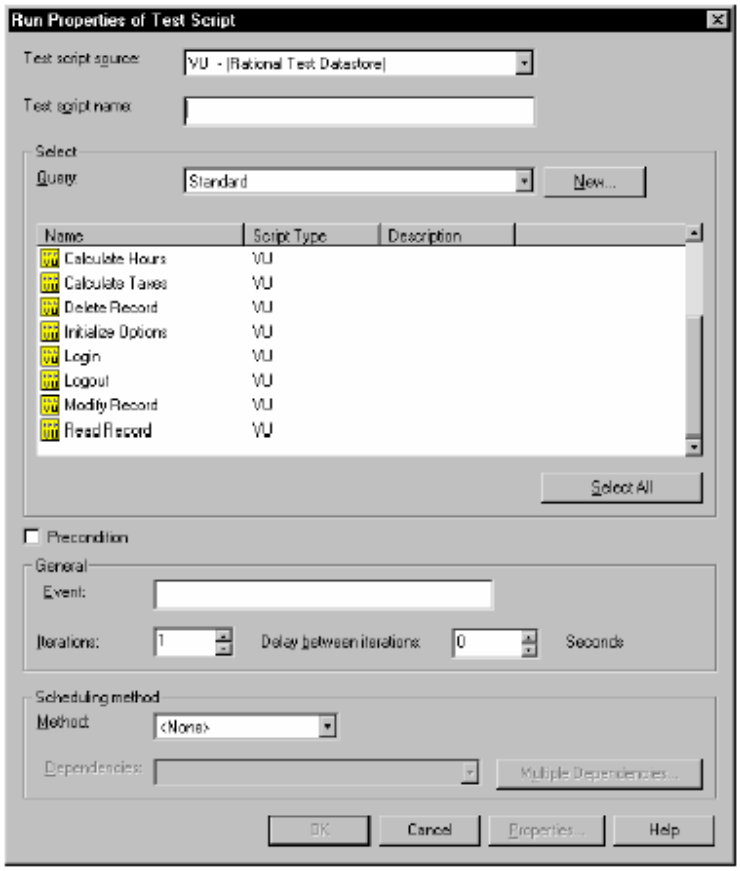
suite中插入用户组并添加用户组执行的测试脚本。226页的suite展现了与测试脚本关联的每个用户组。Accounting组执行两个测试脚本，DataEntry组执行五个测试脚本，Sales组执行三个测试

脚本。

注意事项：不能在一个用户组内同时混入GUI和VU测试脚本。但可以混入其他类型的测试脚本。

suite中插入测试脚本：

●从一个打开的suite中，选择执行测试脚本的用户组，然后点击**Suite > Insert > Script**。



在测试脚本，测试用例，或 Suite 中设置前置条件（Setting a Precondition on a Test Script, Test Case, or Suite）

插入测试脚本、测试用例、或suite到一个suite中，可以定义“成功完成”作为后续的suite顺列执行的前置条件。为了后续suite项在相同条件下的执行，该项必须通过。

例如，假设一个suite包含了两个suites，每一个都包含了一个预置的测试脚本和若干个测试用例。如果在这个预置的测试脚本上设置一个前置条件，并且该测试脚本失败了，那么TestManager跳过这个suite中的剩余测试用例。该suite在第二个suite开始时重新开始执行。

设置前置条件：

●右键点击测试脚本，或测试用例，选择**Run Properties**。

前置条件只适用于说明测试脚本，测试用例，或suite的实例。

如果多次的插入一个测试脚本，而且希望在所有的测试脚本实例上设置一个前置条件，那么就必须为每一个测试脚本设置前置条件。

插入其他项到 Suite 中（Inserting Other Items into a Suite）

一个suite只需要用户组和测试脚本来执行。但是在现实工作模型中的suite，实际的用户执行的工作要比简单的模型复杂的多。一个真正的suite可能也包含测试用例，控制测试suites，场景（scenarios），延迟（delays），同步点（synchronization points），以及交易者（transactors）来表示各种各样的虚拟测试者动作。

除了这些可以在TestManager中添加的项之外，还可以通过Rational Robot来添加某个特征到测试脚本中去。这些项——定时器、组块和注释—在*Using Rational Robot*手册中有详细的讨论。

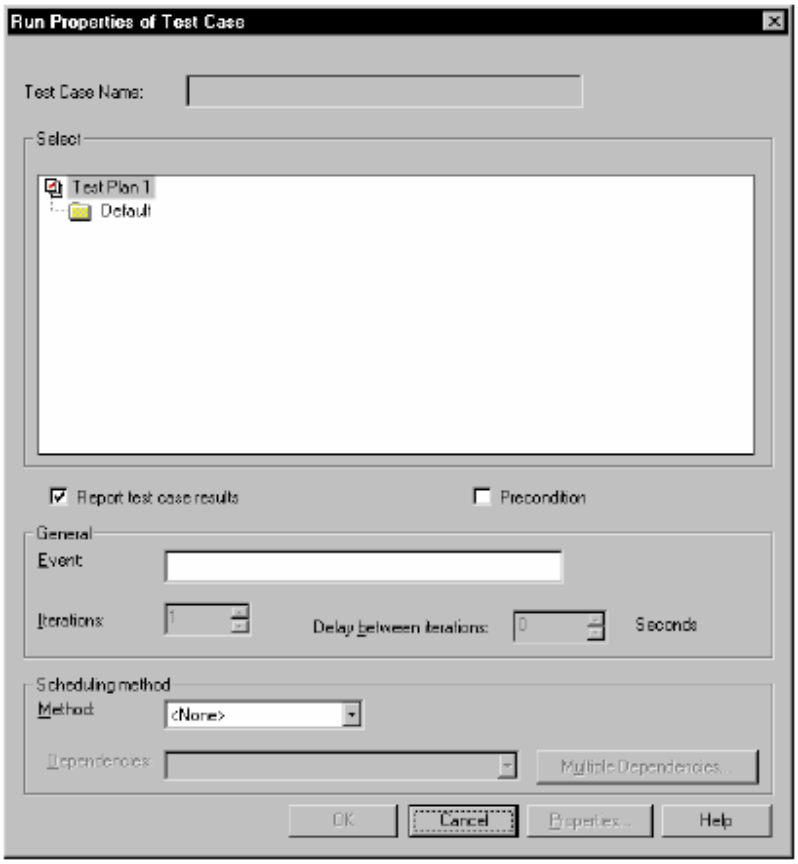
插入测试用例到Suite中（Inserting a Test Case into a Suite）

插入测试用例到suites中，便于一次执行多个测试用例，并一起保存在执行的测试用例集合。

插入配置的测试用例来查证测试用例在不同环境中的执行。当在suites中插入配置的测试用例，TestManager会自动地分配测试用例到适合地配置测试机中。

插入测试用例到suite中的方法：

●从打开的suite中，点击**Suite > Insert > Test Case**。



在一个测试用例上设置前置条件。设置一个前置条件时，该测试用例必须成功完成执行，以便其他具有此相同父用例的suites项执行。

有关前置条件的设置，参阅231页*Setting a Precondition on a Test Script, Test Case, or Suite*的内容。

设置测试用例的前置条件的方法：

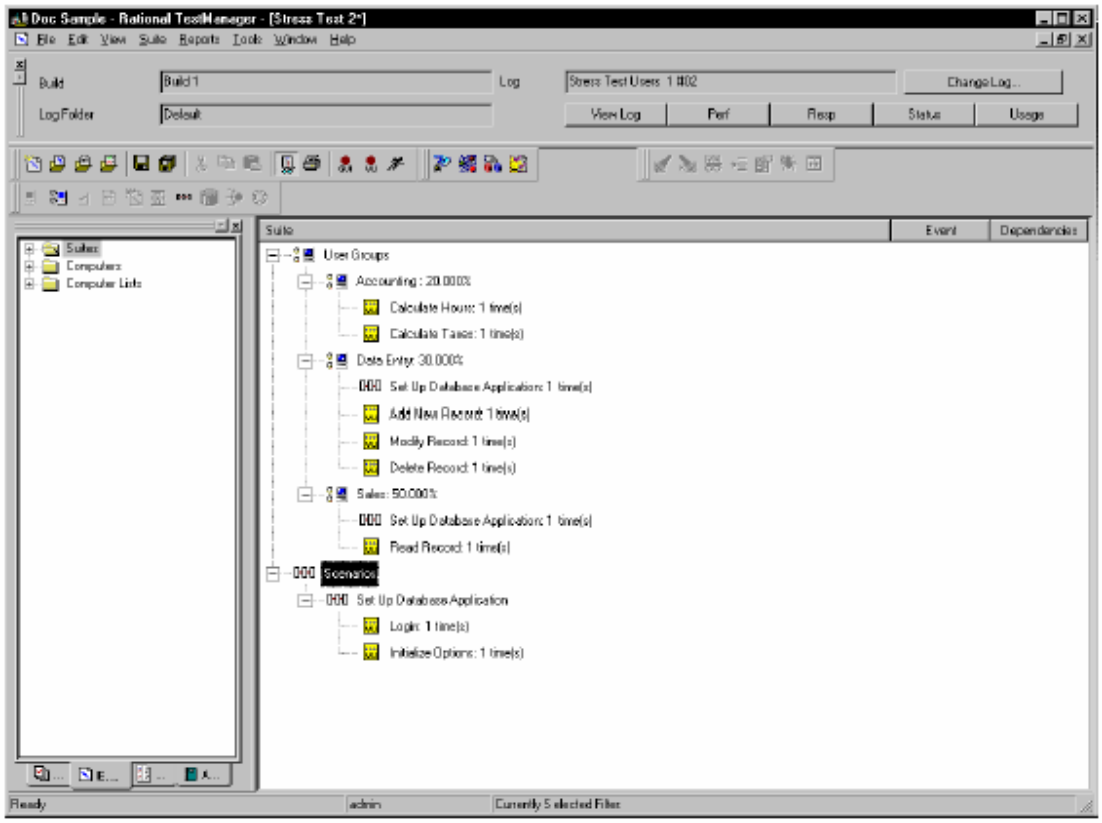
●右键点击测试用例，然后选择**Run Properties**。

插入一个场景（Inserting a Scenario）

“场景”使测试脚本被分组，使他们能被更多的用户组共享。如果是一个使用了多个测试脚本的复杂suite，将这些测试脚本分组到一个场景下，这对于构造你的suite有很大的好处，并且能更容易地读取和维护。

在suite的**Scenarios**部分通过创建一个场景来定义它，并向其中插入一些项。要使一个用户组去执行一个场景，需要插入该场景的名称到这个用户组中。否则，该场景不能被执行。

226页中的suite，Data Entry 和Sales用户组执行的测试脚本是Login和Initialize Options。将这两个测试脚本保存到一个场景中，可以简化这个suite。下面的这个suite展现了Login和Initialize Options这两个测试脚本被分组在Set Up Database Application场景下：

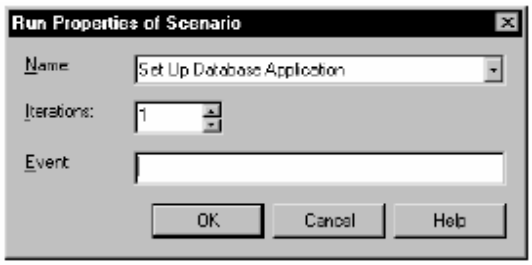


创建一个新的场景：

●从suite的Scenarios部分，点击**Suite > Insert > Scenario**。

插入场景到suite中：

●点击你希望放置场景的部分，然后点击**Suite > Insert > Scenario**。



在创建场景之后，该场景未添加到一个用户组之前，放置一个场景是一个不错的方法。一个场景只需要测试脚本就可执行。

然而，如同一个用户组，一个真正的场景也可以包含选择器，延迟，同步点，和交易者。甚至场景中也可以包含其他的场景。

插入一个Suite到Suite中（Inserting a Suite into a Suite）

虽然在一般情况下，性能测试中使用的suite包含用户组（而非测试机组），但是也可以插入一个包含了测试机组的suite到另一个suite中。插入suite到另一个suite中的优势是变更，这个变更可以持续的构造子suite。因此，插入一组测试脚本到一个基于测试机的suite中，在不同的suites中使用该组，但只能更新这个suite一次。

更多信息，参阅171页*Inserting a Suite into a Suite*的内容。

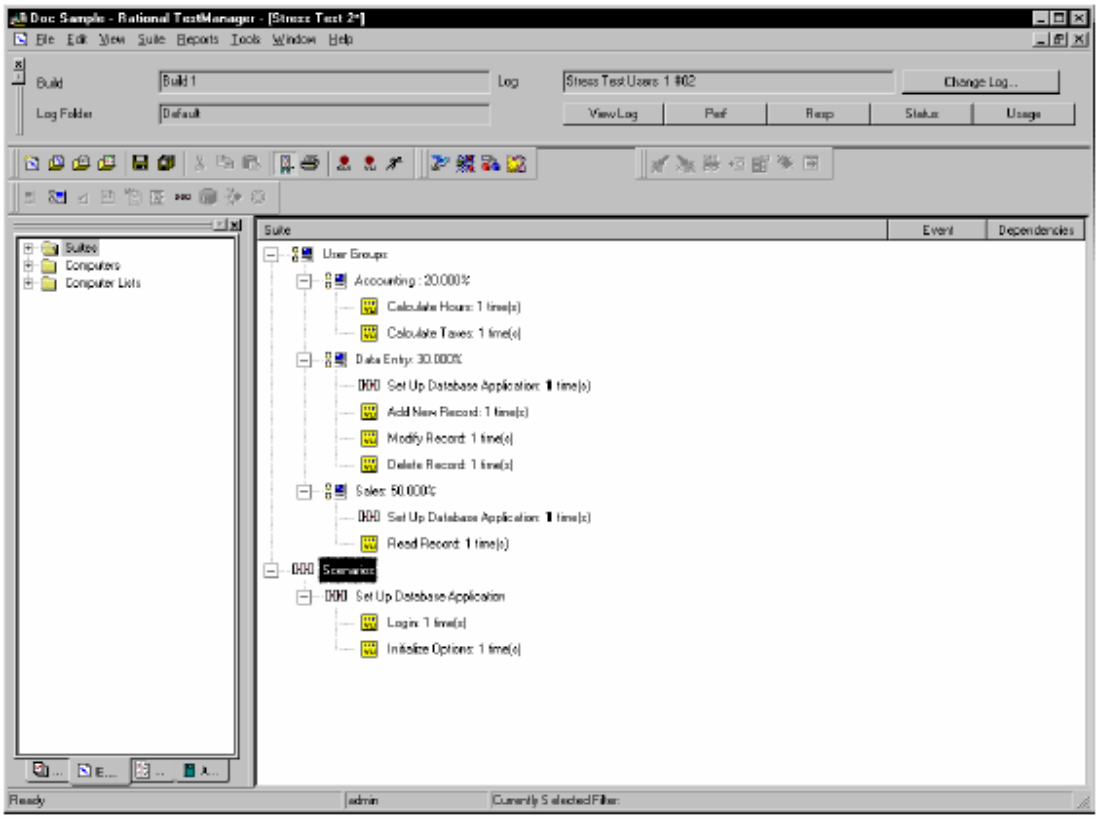
插入一个选择器（Inserting a Selector）

TestManager允许通过设置“选择器”（*selector*）来设置suite项，是suite以不同的顺序执行。在suite中，相比运行一个简单的连续顺序项，“选择器”对suite提供了更为高级和复杂的控制。一个*selector*告诉TestManager每个虚拟测试者执行的项，和执行的顺序。例如，如果希望从一组测试脚本中随机的重复选择一个测试脚本。选择器可以帮助你做这些事。

插入选择器到suite中：

●选择需要包含选择器的用户组或场景，然后点击**Suite > Insert > Selector**。

考虑下面的选择器，那些不包含任意的选择器：



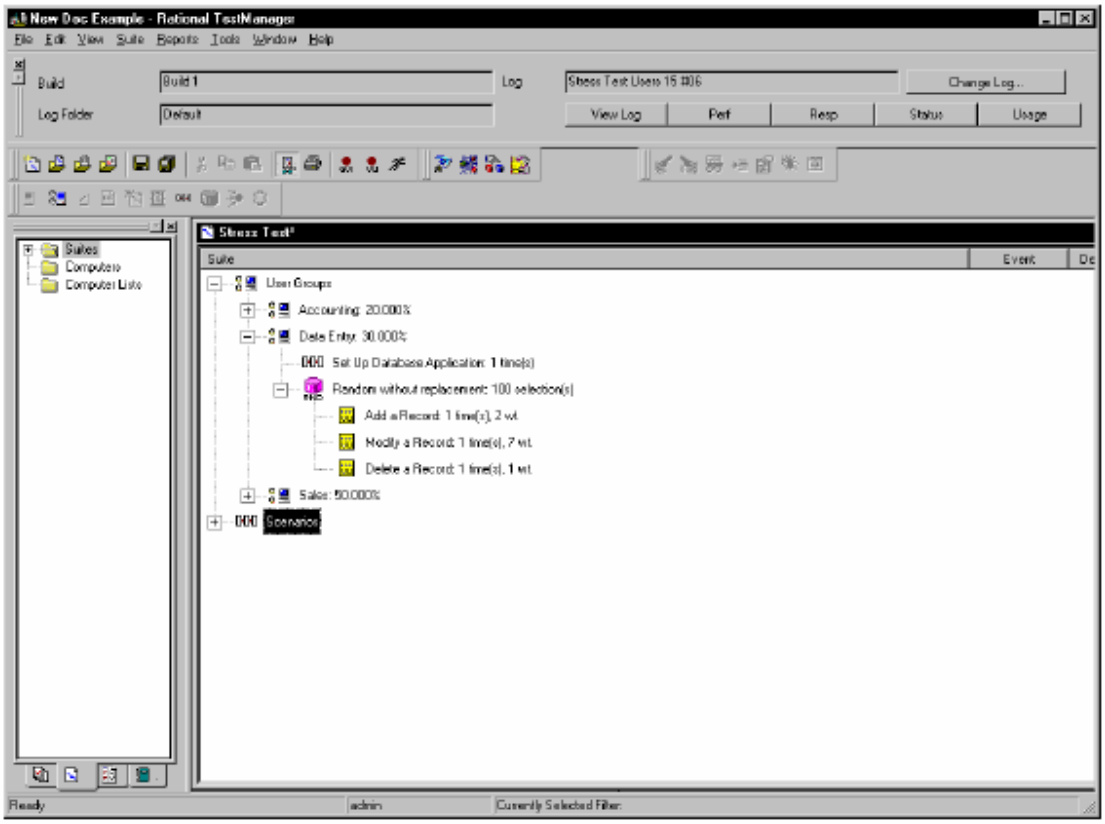
当你执行具有50个虚拟测试者的suite时，TestManager分配10个虚拟测试者到Accounting，15个虚拟测试者到Data Entry，和25个虚拟测试者到Sales（基于明确说明的有比例的用户组）。所有的50个虚拟测试者在相同的时刻开始执行。

- 这10个Accounting虚拟测试者按顺序执行每个出现在该suite中的测试脚本：先是Calculate Hours，然后是Calculate Taxes。
- 这15个Data Entry虚拟测试者按顺序执行每个出现在该suite中的测试脚本：执行Set Up Database Application场景，然后执行Add New Record，Modify Record，和Delete Record测试脚本。
- 这25个Sales虚拟测试者先执行Set Up Database Application场景，然后执行Read Record测试脚本。

然而，假设你的Data Entry虚拟测试者实际上是随机地添加纪录，删除纪录，和修改纪录。而且，他们不是以相同的频率来执行这些任务。因为他们删除每个纪录，修改了7个纪录，并添加了2个纪录。

要使你的用户组反映出这些行为，插入一个“随机选择器”（*Random selector*）到Data Entry用户组中。下面的suite展现了Data Entry用户组设置的选择器，在没有代替者的情况下随机

地选择测试脚本。



当你执行具有50个虚拟测试者的suite时，围绕用户组的说明，每个Data Entry虚拟测试者：

- 执行Set Up Database Application场景。
  - 每次迭代选择一个测试脚本：Add New Record，Add New Record，或Delete Record。从100次迭代开始起，每个Data Entry虚拟测试者20次纪录添加， 70次纪录修改，和10次纪录删除的操作。添加，修改和删除操作具有任意的顺序。

选择器的类型（Types of Selectors）

TestManager提供以下的选择器类型：

- 连续的（**Sequential**）——按顺序执行出现在suite中的每一个测试脚本和场景。该类型是默认的。
  - 并行的（**Parallel**）——使它的测试脚本或场景分布到一个可用的虚拟测试者中（每测试机一个虚拟测试者）。一般地，在功能测试中使用这个选择器。

    该项按顺序分开，依据虚拟测试者来执行测试脚本。一旦有一项被执行，那么它就不能再次被执行。

并行选择器在不关注其迭代的情况下来分配每个测试脚本。

- 有重置的随机（**Random with replacement**）——该选择器随机的执行这些项，每次有一个项被选择，那么它再次被选择的机率与剩余项相同。

  例如，一个水桶里有10个红色的球和10个绿色的球。选择红色球和绿色球的机会是一样的，都是50%。第一个被选择的球是红色的，然后将这个红色的球和其他红色的球一起重新放置在水桶中。每次你挑出一个球，你依然有50%的机会获得一个红色的球。

  每次选择之后，该球被重新放置，水桶中总是有10个红色的球和10个绿色的球。甚至每次你都可能（但未必）挑出一个红色的球。同样的，具有替换的随机选择器并不保证执行他们中的每一项，特别地，如果你设置了一个比其他脚本执行频率更高的测试脚本。换句话说，如果你的水桶里有19个红色的球和一个绿色的球，那么这个绿色的球可能一次都不会被选择。

- 无重置的随机（**Random without replacement**）——该选择器随机的执行这些项，但每次有一个项被选择，那么它再次被选择的几率会发生变化。例如，考虑相同的水桶中有10个红色的球和10个绿色的球。同样的，第一个被选择的球是红色的。然而，这个球不会被重新放回水桶中。因此，下一次你就有较大的机会来挑出一个绿色的球。每次你选择一个球，你的几率都会变化。

  因此，如果第一次选择的10个球是红色的，那么下一次选择的这10个球为绿色的几率是100%。同样的，只要选择器的迭代数量大于或等于该选择器中的项数，无重置的随机选择器将在其中执行每一个项。

- 动态负载均衡（**Dynamic load balancing**）——利用动态负载均衡，这些项不是被随机选择的。再次考虑这个有红色球和绿色球的水桶。你分配给每个球以相同的“重量”。如果第一个被选择的球是红色的，那么第二个选择的球总会是绿色的。这是因为每个被选择的球或测试脚本，系统“动态平衡”工作负载以接近你设置的50-50的重量。你也可以设置其他的总量。这里的关键点是，下一个测试脚本的执行不是被随机选择的，它的选择是根据你设置的重量来平衡工作负载。

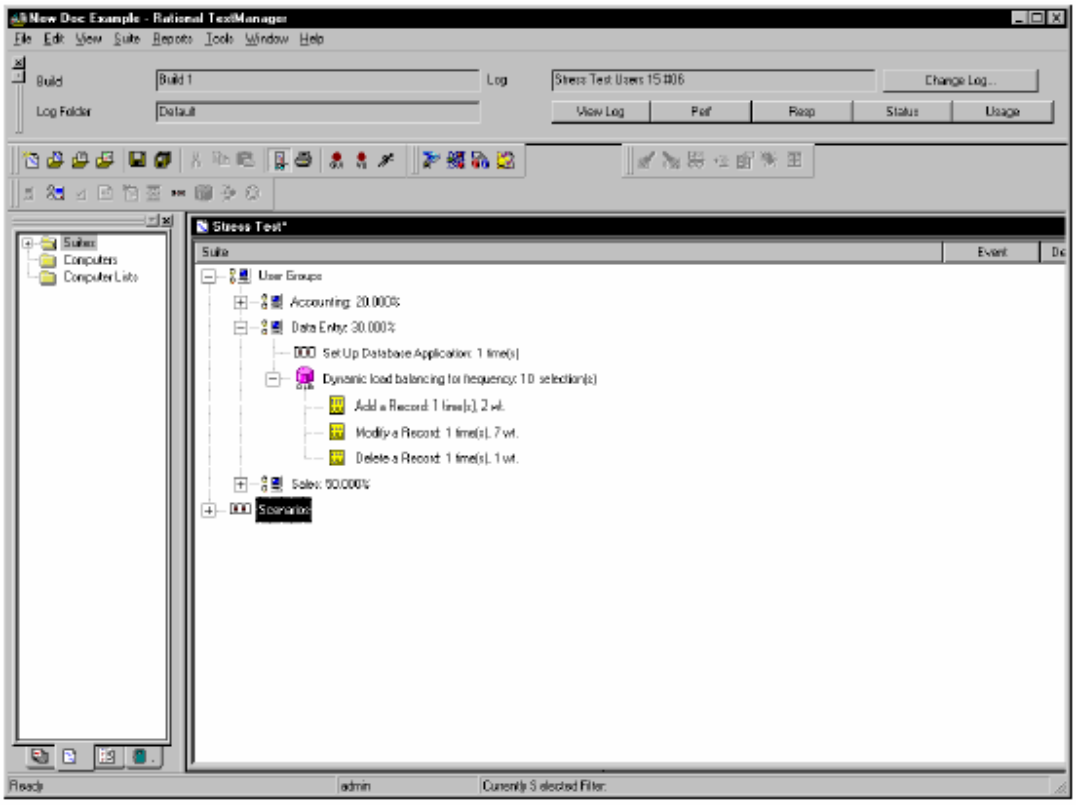
同时也可以平衡工作负载的时间或频率。例如，假设动态均衡脚本A和脚本B，并且使用了相同的重量。不管怎样，脚本A执行的两倍时长和脚本B是一样的。

如果动态负载均衡的是时间，那么负载是通过每个测试脚本的执行时间来均衡的。因为脚本A获得两倍的执行时长，它实际上和脚本B一样被选择的只有一半。

如果动态负载均衡的是频率，那么两个测试脚本执行的次数是相同的。如果脚本A执行500

次，那么脚本B也会执行500次。事实上，脚本A获得长时间的执行不是均衡中的因素。

动态负载均衡是通过用户组中的所有虚拟测试者来完成的。例如，下图展示了含有15个虚拟测试者的Data Entry组。三个测试脚本，Add New Record，Modify Record，和Delete Record，是包含在动态负载均衡选择器中的。



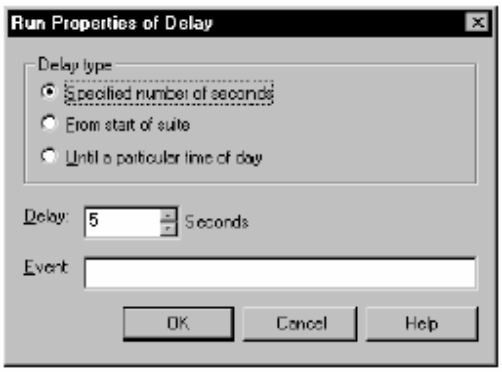
在执行该suite时，第一个Data Entry虚拟测试者选择Modify Record脚本，因为最大的重量。但是由于工作负载是通过所有的Data Entry虚拟测试者来均衡的，所以在第一个虚拟测试者退出之后，TestManager重新计算重量来反映出这个细节，这个已经被选择的测试脚本具有最大的为7的重量。在较晚的虚拟测试者准备选择一个测试脚本的时候，重量已经发生变更，以使他们有更大的选择Add New Record的机会。

插入一个延迟（Inserting a Delay）

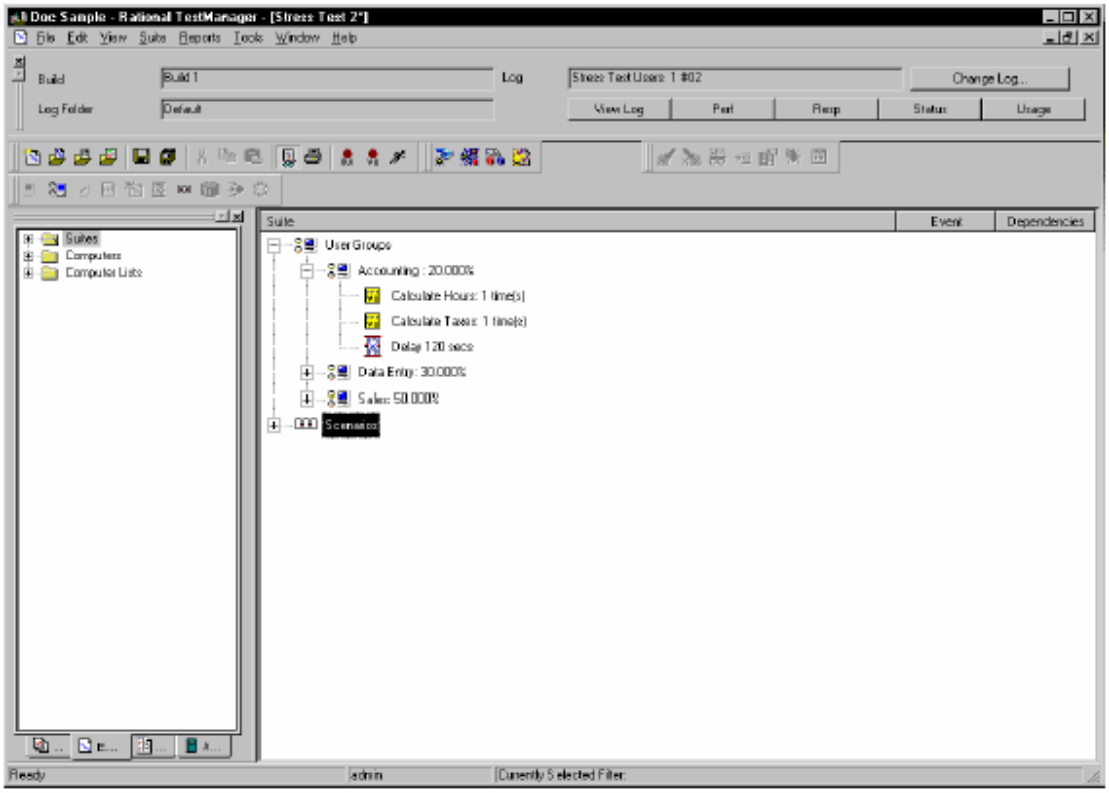
“延迟”是告诉TestManager在suite中它执行下一个项之前，需要暂停多长的时间。

插入一个延迟到suite：

●点击需要添加延迟的用户组，场景，或者选择器，然后点击**Suite > Insert > Delay**。



在性能测试中，使用延迟来效仿典型用户行为。例如，如果Accounting用户组计算时间和税额，然后暂停两分钟，在Calculate Taxes测试脚本之后添加一个延迟，就像下面的suite所展示的：



在suite中或一个测试脚本中插入一个延迟。插入延迟的好处在于，该延迟在suite中是可见的，并且在不编辑该测试脚本的情况下，该延迟是容易变更的。

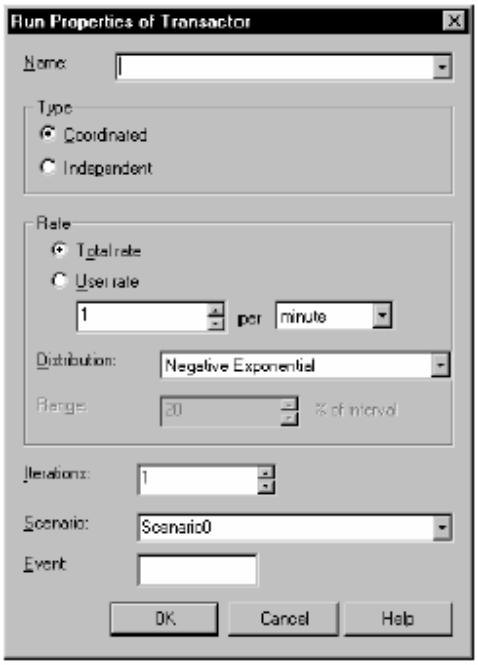
插入交易者（Inserting a Transactor）

“交易者”告诉TestManager在给定的时间段内每个虚拟测试者执行的任务数量。例如，测试Order Entry组，需要每小时完成10 forms，或者在测试一个Web服务器，希望能够支持每分钟100次的点击量。使用交易者来效仿这种基于时间的行为。

在suite中插入交易者：

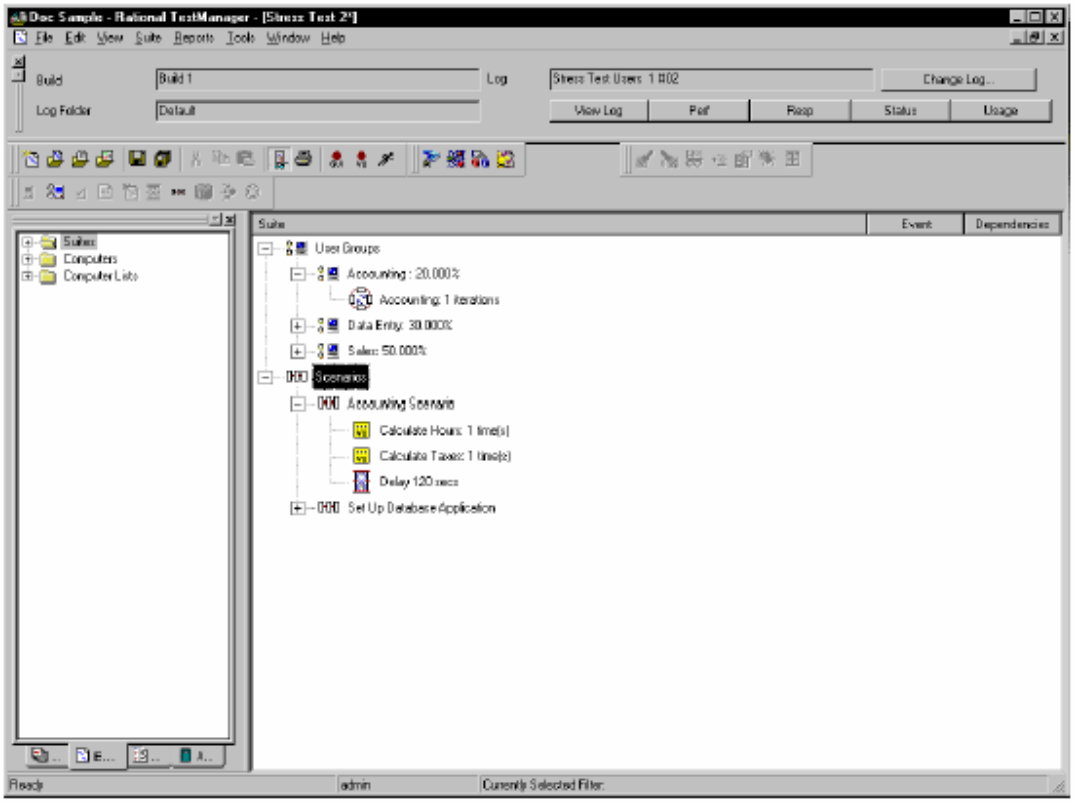
选择交易者的用户组或选择器，点击**Suite>**

插入>交易者



在前一节添加了一个延迟到Accounting用户组。这个延迟使虚拟测试者在计算时间和税额之后暂停两分钟，如242页的图示。

然而，假设Accounting组在计算时间和税额时，它的交易率为每小时10笔交易。那么可以编辑suite来反映出这个交易率，具体是利用交易者来替换选择器和延迟。下面的suite展现了添加了一个交易者之后的Accounting用户组。



这个suite和242页的那个几乎完全相同，只是它包含了：

- 一个交易者，它告诉TestManager你希望维持的速率，以及你希望维持这个速率的时长。
- 一个场景，它包含了交易者将要执行的一些项。

交易者类型可以是下面这两种类型中的一种：

- “协调的”交易者，它是一个内建的同步点，用来指定你希望达到的总速率。虚拟测试者一起工作来产生工作负载。例如，如果执行一个具有10个虚拟测试者的suite，再执行具有20个虚拟测试者的相同的suite，总交易率是不会发生变化的。

  当模拟一个适用于服务器的总交易率时，需要使用一个协调的交易者，而不是一个虚拟测试者执行一项任务的具体时间率。例如，要模拟一个Web服务器可以处理的每分钟点击量，可以使用协调的交易者。

- “独立的”交易者使每个虚拟测试者独立地进行操作。它不会将虚拟测试者协调在内建的同步点之下。例如，如果执行一个具有10个虚拟测试者的suite，然后再执行具有20个虚拟测试者的相同的suite，那么总交易率将增加一倍—因为虚拟测试者的数量增加了一倍。

  如果不同的用户组在不同的时间执行交易，或者需要模拟单个的行为而非一组行为的话，可以使用“独立的”交易者。例如，要模拟Accounting用户组每小时只执行10次计算，而不是在

相同的时间里进行全部的计算，可以使用一个独立的交易者。

如果你定义了交易者类型，那么你必须指定该交易者的速率：

- 总速率（**Total rate**）——对于一个协调的交易者，一般情况下选择总速率。这是因为不论是100个虚拟测试者还是50个虚拟测试者的参与，它都不会影响TestManager提交“交易事务处理”这个速率。
- 用户速率（**User rate**）——对于独立的交易者，必须选择用户速率。

  如果期望改变速率并希望不必每次都要编辑suite，那么可以选择用户速率。例如，假设你插入了一个协调的交易者，同时希望比较每分钟分别为100次点击，200次点击和300次点击时的工作负载—每个suite执行时增加这些工作负载。如果你选择用户速率，那么就没有必要去修改交易者属性中的速率。相反的，当你执行分别为100个，200个，和300个虚拟测试者的suite时，该速率是成比例换算的。

接下来，指定交易者的分布：

- 连续分布意味着每个交易完全地发生在你说明的速率上。例如，如果交易率是每分钟4次，那么交易者开始与15秒，30秒，45秒和60秒—完全地每分钟4次，均匀地分开，有一个15秒钟的间隔。虽然这种分布在概念上来说是简单的，但它不能准确地模拟用户的随机行为。

  连续分布对于模拟一个自动过程是有效的。例如，你可能希望模拟一个环境，在其中，虚拟测试者每半个小时上传数据到数据库。
- “均匀分布”是指在时间中，交易在指定的速率中达到平均，即使每次交易之间的时间是连续的。通过均匀分布，每个交易开始之间的时间在选择的范围内是被随机选择的。考虑这个范围，交易执行所通过的“窗口”。

  例如，交易率为每分钟4次（每15秒钟间隔一次）。如果选择一个20%的范围，那么交易在15秒间隔的每一侧上有一个3秒的窗口，因为15秒的20%是3秒。

  因此，第一次交易开始与12—18秒（15减或加3）。第二次交易在第一次开始之后开始，15秒减或者加3秒。如果第一次开始与12秒，那么第二次交易则开始与24到30秒。当然，如果第一次开始与18秒，那么第二次交易则开始与30到36秒。

  由于每次交易在你指定的范围内是随机开始的，所以这个速率比你选择的短时间内的速率快或者慢一些，这是正常的。例如，交易在一分钟内每12秒开始一次（回忆那个窗口是12—18秒），对于这个初始的间隔，速率是每分钟5次—不是所选择的每分钟4次。但是，在整个时间段中，交易率达到每分钟4次的平均。

  利用均匀分布，交易在指定的范围内具有相同的执行机率。交易开始与这个窗口中的任意

地方。在例子中，第一次开始与12秒的交易和开始与18秒的交易的几率是相等的。

- “负指数分布”就是改变交易开始时的机率。这种分布最接近地模拟用户行为的特点，通过使活动逐渐地减少（by a tapering off of activity）而伴随的突然增加的大量活动。在同一个例子中，每分钟4次交易，一次立即开始的交易机率是高的，但在整个时间内会减少。TestManager维持这个期望的平均速率。

假设要在2点时召开一个会议。大多数的人都是在2点钟到达的，有一些人是在2点过5分的时候到达的，较少的人是2点过10分到达的。也许最后的人是在2点半的时候来的。这个到达时间就近似一个负指数分布。大多数的人准时参加会议，然后到达率将会下降。用数学的理论来说，一个具有平均（时间）间隔的负指数分布，它的（时间）间隔是随机选取的，而这个（时间）间隔等于1/速率。

交易者可以被插入到一个用户组或独立地在一个顺序或随机的选择器中。如果在一个随机选择器中插入一个独立的交易者，那么必须指明该选择器的重量。有关选择器的信息，参阅238页*Types of Selectors*的内容。

交易者可以设置一个事件。有关事件的信息，参阅254页*Using Events and Dependencies to Coordinate Execution*的内容。

插入同步点（Inserting a Synchronization Point）

“同步点”通过在一个特殊点（即同步点）处暂停每个虚拟测试者的执行，使你协调虚拟测试者的活动，直到以下的一个事件发生：

- 所有关联到同步点的虚拟测试者到达这个同步点。

  当一个虚拟测试者遇到一个同步点时，该虚拟测试者停止并等待其他的虚拟测试者到达。当被指定到达该同步点的虚拟测试者全部到达该点时，TestManager释放这些虚拟测试者并使他们继续执行该suite。

- 在所有的虚拟测试者到达该同步点之前，到达一个超时时间。

  当一个虚拟测试者遇到一个同步点时，该虚拟测试者停止并等待其他的虚拟测试者到达。其他虚拟测试者到达该同步点并等待。然而，在所有的虚拟测试者到达该同步点之前，已到了超时时间，TestManager释放这些虚拟测试者并使其继续执行该suite。还没有到达该同步点的虚拟测试者，在未达到超时时间之前，不能再该同步点处停止。他们也会继续执行该suite。

- 你在监控suite时，手动释放虚拟测试者。

  当一个虚拟测试者遇到一个同步点时，该虚拟测试者停止并等待其他的虚拟测试者到达。其他虚拟测试者到达时该同步点等待。然而，在这个时候，来决定从该同步点处释放这些虚

拟测试者，并继续执行suite。所有的虚拟测试者已经到达或还没有到达该同步点。那些没有到达同步点的虚拟测试者，在手动地释放他们之前，不能在该同步点处停止。他们也会及继续执行suite。

插入同步点：

- 插入测试脚本——用以下的方法使用Rational Robot插入一个同步点到测试脚本中：
  - 在录制期间，使用工具栏的按钮或Insert菜单。
  - 在测试脚本编辑期间，通过手动输入同步点命令到测试脚本中。

    插入同步点到测试脚本中来准确地控制测试脚本暂停的地方。例如，就在你发送一个请求到服务器之前，插入一个同步点命令。

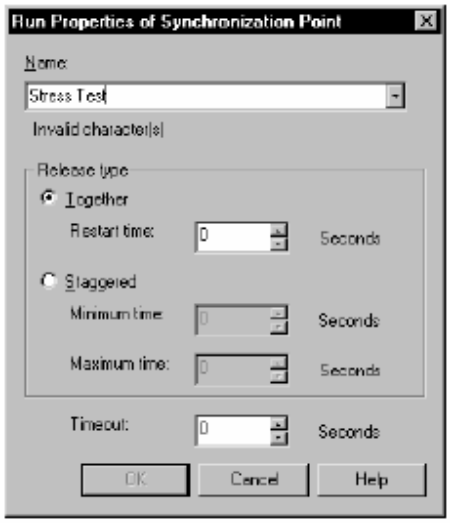
    同步点依赖于相同逻辑情况下使用这种方法，而这种相同的逻辑是在编辑期间，你添加到测试脚本中去的。
- 插入suite—通过TestManager插入一个同步点到suite中。

  插入同步点到TestManager中的suite，来暂停测试脚本之前或之间的执行，而不是一个测试脚本内部的执行暂停。插入同步点到suite中，有下面这些优点：
  - 你可以在不去编辑测试脚本的情况下，容易地移动同步点的定位。
  - 在suite内部的同步点是可见的，而不是隐藏在一个测试脚本中。

  当你插入同步点到TestManager中的suite时，比之分配一个同步点到一个测试脚本的工作，你可以做的更多。例如：
  - 你可以说明你是否希望虚拟测试者在相同或者不同的时刻处被释放。
  - 如果虚拟测试者在不同的时刻（交错）被释放，那么你就可以指定虚拟测试者被释放的最小和最大时间。
  - 可定义超时时间

插入同步点到suite中：

- 点击**Suite > Insert > Synchronization Point**。



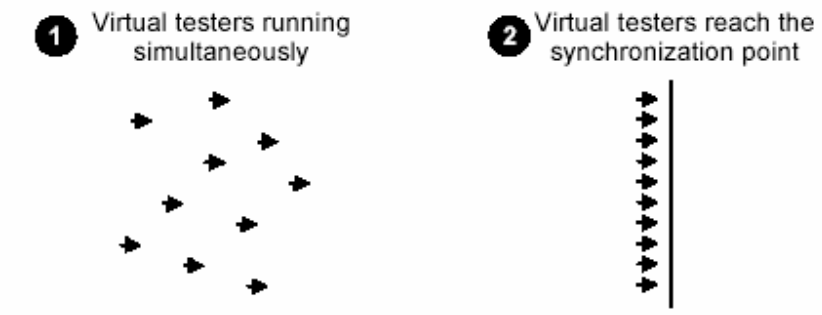
例如，当执行一个压力测试时（在极端条件下执行你的应用程序来查看他们或服务器是否会档机）， suite可能包含这样的虚拟测试者，他们不断地执行某项操作并连续地重复几个小时。要最有效地执行压力测试，可以同步这些虚拟测试者，以使他们在相同的时刻内执行操作来进行系统的压力测试。也可以通过插入一个同步点的方式来到达这种目的，以协调这些虚拟测试者来同时执行某项功能。

同步点如何工作（How Synchronization Points Work）

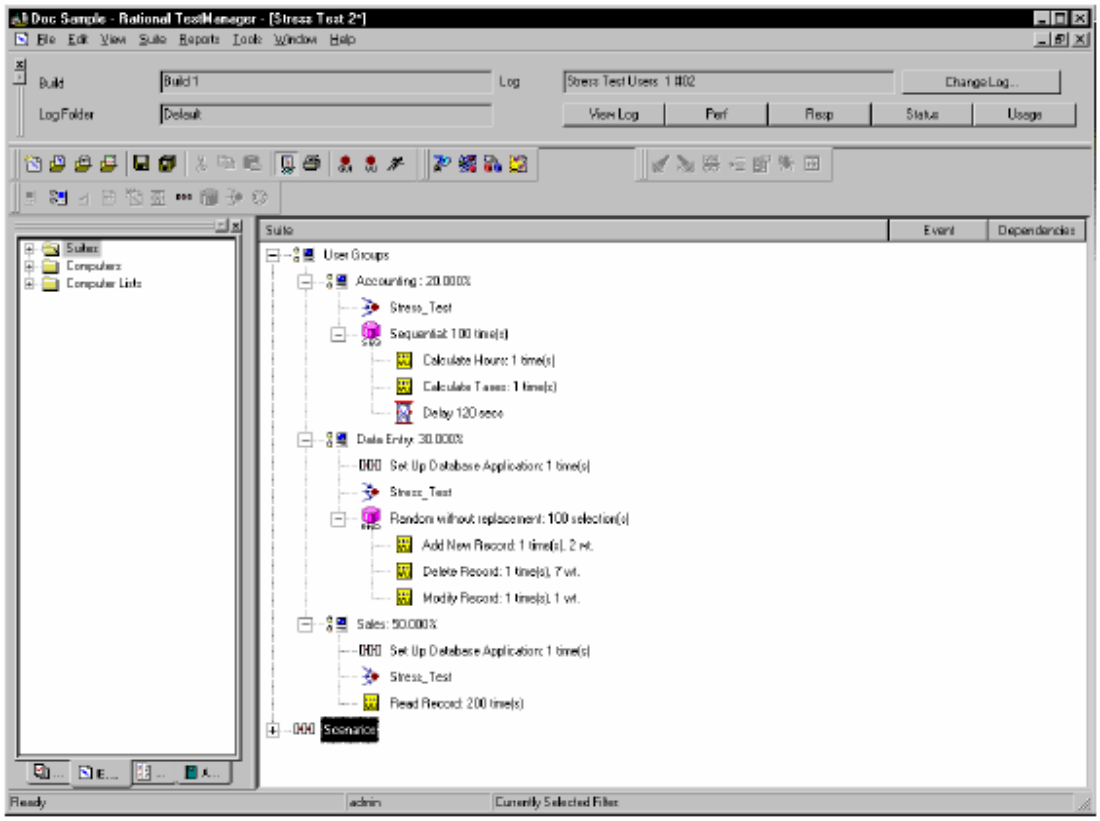
在测试开始时，所有的虚拟测试者开始执行其下的测试脚本。直到他们到达同步点，否则会继续执行下去。在测试脚本中指定时，同步点是一个程序式命令（在VU测试脚本中是sync_point，在SQABasic测试脚本中是SQASyncPointWait，在VB测试脚本中是TSSSync.SyncPoint，在Java测试脚本中是TSSSync.syncPoint）。在suite中指定时，同步点的放置类似于其他的suite元素（延迟，交易者，以及其他等）。

下图说明了一个同步点：

虚拟测试者在同步点处暂停，直到TestManager释放他们。



下面的suite展现了被称为压力测试的同步点：



Accounting用户组中的虚拟测试者等待同步点。

Data Entry和Sales用户组执行Set Up Database Application场景，然后等待同步点。当所有的虚拟测试者到达该测试点时，他们被释放。

如果执行一个含有10,000虚拟测试者的测试，当所有的虚拟测试者到达这个“压力测试”同步点时，他们被释放。在这个例子中：

- Accounting组有2000个虚拟测试者，每一个测试者计算时间和税额，暂停2分钟，然后重新计算时间和税额。每个虚拟测试者重复进行100次这样的操作。
- Data Entry组有3000个虚拟测试者，每一个测试者添加，删除，或者修改一个记录。每个虚拟测试者重复进行100次这样的操作。
- Sales组有5000个虚拟测试者，每一个测试者读取一个记录。每个虚拟测试者重复进行200次这样的操作。

在设置同步点时，你必须说明虚拟测试者如何从同步点处释放：

- 同时（*Together*）——一次性释放所有的虚拟测试者。

  指定一个重新开始的时间来延迟虚拟测试者。例如，如果重新开始的时间设置为4秒，在虚拟测试者到达该同步点之后（或已经超时），他们等待4秒，然后全部被释放。

默认的重新开始时间为0，这意味着当最后一个虚拟测试者到达该同步点时，所有的虚拟测试者立刻被释放。

- 交错（*Staggered*）——一个接一个的释放虚拟测试者。

  每个虚拟测试者等待被释放的时间总量是随机选取的，并且在指定的最大和最小的时间范围内成均匀分布。例如，如果最小时间为1秒，最大时间为10秒，在虚拟测试者到达该同步点（或发生超时）之后，每个虚拟测试者在被释放之前，等待1到4秒。所有的虚拟测试者随机地分布在1和4秒之间。

对于同步点的超时的时间被指定为，TestManager等待虚拟测试者到达该同步点的总时间。如果所有关联与一个同步点的虚拟测试者在超时时间结束时，没有到达该同步点，那么TestManager释放任意的虚拟测试者以等待。在第一个虚拟测试者到达该同步点时，超时时间开始。

虽然在一个超时之后到达同步点的虚拟测试者不被保留，但该虚拟测试者会在同步点处被延迟。例如，如果超时时间已到达，且重新开始的时间为1秒，最大时间为4秒，那么虚拟测试者被延迟在1和4秒之间。

默认超时为0，这意味着此处没有超时时间。设置超时是有用的，因为一个虚拟测试者可能遇到问题，并且可能从来都没有到达过同步点。在你设置时间时，你没有停止其他的虚拟测试者，这是因为问题只是这一个虚拟测试者的问题。

Suite或测试脚本可以有多个同步点，每一个都具有唯一的名称。给定的同步点名称可以在多个测试脚本和（或）suites中被引用。

为什么要使用同步点？（Why Use Synchronization Points?）

通过同步虚拟测试者，可以使其在相同的时刻执行相同的活动，可以使这些活动发生在测试中的一些特殊的点上—例如，应用程序向服务器发送一个查询时。

测试脚本中的同步点常常和计时器一起使用，来确定在定时活动上不同虚拟测试者负载的影响。例如，要模拟数据检索上的虚拟测试者的负载：

1 录制一个提交请求和展示结果的测试脚本（在此例中名为VU1）时，执行以下的活动：

a 插入名为`TestQuery`的同步点到测试脚本中

b 开始一个block。

该block定时执行交易。它也将该block和计时器名称关联于执行此交易的模拟命令的名称。

c 提交查询，并等待结果展现。

d 停止该block。

2 录制提供负载的测试脚本时，就在你刚刚开始记录提供负载的任务之前，插入另一个

TestQuery同步点。例如，就在你点击提交定购单的按钮之前，添加一个同步点。将此测试脚本命名为VU2。

3 添加VU1和VU2到suite中。

4 多次执行该suite，每次添加不同数量的虚拟测试者到VU2脚本中。然而，在每次测试中，只需要一个虚拟测试者来执行VU1测试脚本就可以了。

理论上，当VU2中同步的虚拟测试者数量增加时，通过VU1计时器纪录的时间也应该增加。

在本例中，TestQuery同步点确保：

● VU2中的所有虚拟测试者在相同时刻提交他们的表单—为此提供最大的并行虚拟测试者的负载。

● VU1中的虚拟测试者在VU2的虚拟测试者加载服务器的时间内，提交它的查询请求—为此提供一个最大负载的临界时间。

测试脚本中的同步点释放时间和超时时间（Release Times and Timeouts for Synchronization Points in Test Scripts）

当你插入同步点到suite时，你不能为插入到测试脚本中的该同步点定义最大和最小的释放时间或超时时间。按照缺省：

● 虚拟测试者保留基于脚本的同步点被同时释放。

● 这里没有时间限制虚拟测试者在同步点处被保留多久。

然而，如果suite中的同步点具有一个释放时间的范围和为其定义的超时时间，那么这个释放时间和超时时间适用于所有具有相同名称的同步点—即使同步点是测试脚本中的。

同步点的范围包含所有的测试脚本，外加所有的用户组。这些测试脚本和用户组都引用一个特殊同步点的名称。

例如，假设在suite中已有下面的这些用户组：

含有75个虚拟测试者的Data Entry用户组。这个用户组执行一个测试脚本，包含一个名为Before Query的同步点。

含有10个虚拟测试者的Engineering用户组。这个用户组执行与Data Entry用户组不同的测试脚本。但在这个测试脚本中也包含一个名为Before Query的同步点。

含有25个虚拟测试者的Customer Service的用户组。这个用户组执行一个不包含同步点的测试脚本。然而，该用户组确实存在一个为其定义的同步点。该同步点也被命名为Before Query。

在suite执行时，当所有的110个虚拟测试者到达他们各自的同步点时，TestManager释放保留在Before Query同步点处的虚拟测试者。

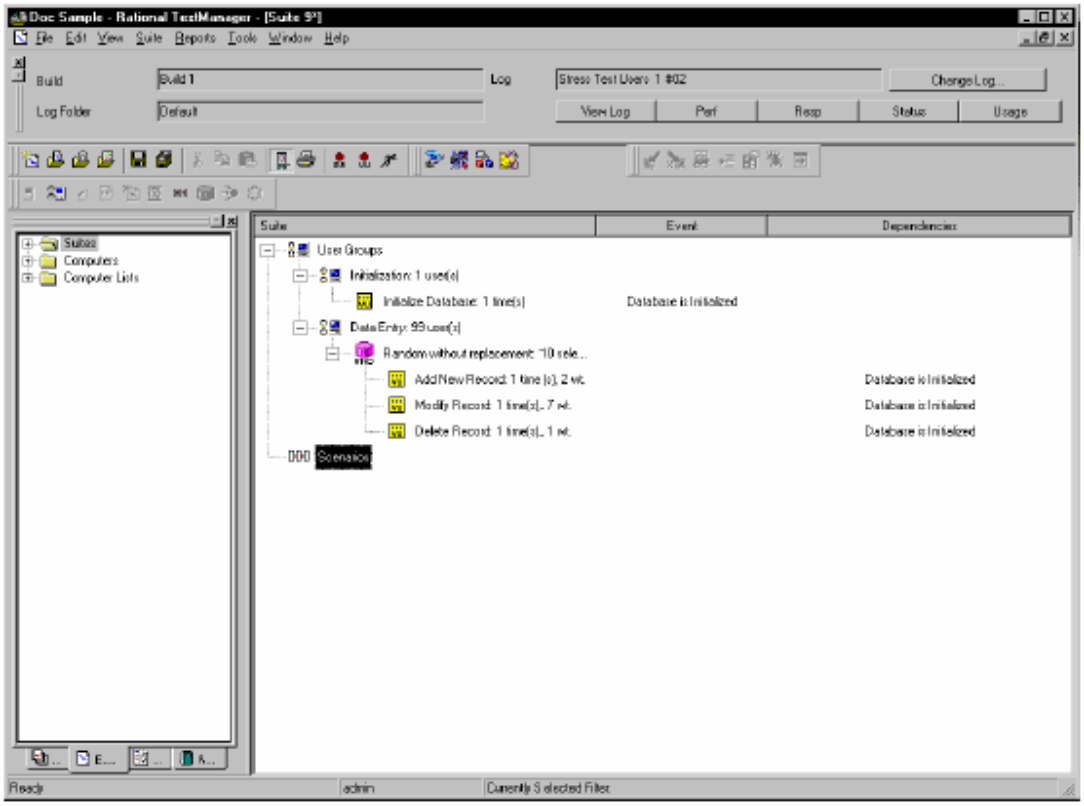
使用事件和依赖协调执行（Using Events and Dependencies to Coordinate Execution）

“事件”是一个机制，用来协调项在suite中执行的方式。例如，你正在执行一个具有100个虚拟测试者的suite，来访问数据库。你希望第一个虚拟测试者去初始化数据库，而其他的99个虚拟测试者等待初始化的完成。要这样运作，你可以设置初始化事件上的一个“依赖”，阻塞这99个虚拟测试者，直到事件（第一个虚拟测试者初始化数据库）发生。

你可以在suite中有多个事件。当suite中只有一项能够设置事件时，有很多的项可以依赖于这个事件。

注意事项：事件和依赖只需要活动的发生—没必要他们成功地完成。如果suite部分需要不仅仅需要活动的发生，还要求成功的完成，那么可以在测试用例，测试脚本，或者suite上使用前置条件。有关前置条件的信息，参阅231页*Setting a Precondition on a Test Script, Test Case, or Suite*的内容。

下面的这个suite展示了99个虚拟测试者的等待，直到第一个虚拟测试者初始化数据库的事件发生。



该suite中的第二行列出了事件，并且第三行列出了依赖。在这个suite中，Initialize Database测试脚本一完成，它就将事件设置为Database Is Initialized。Add New Record， Modify Record，和Delete Record测试脚本依赖于该事件，并且只能在事件被设置之后执行。

注意事项：在前一个例子中，Data Entry用户组中的虚拟测试者随机地执行测试脚本。在这个用例中，你必须添加依赖到选择器中的每一个测试脚本中，因为你不知道那个脚本会第一个执行。但是，如果Data Entry用户组是顺序执行这些测试脚本的，那么只要向第一个测试脚本添加一个依赖就可以了。

- 添加一个设置或依赖于事件的测试脚本：

点击**Suite > Insert > Test Script**。

注意事项：上一个例子展示了如何添加设置了事件的测试脚本和另一个依赖于事件的测试脚本。然而，场景，交易者，和延迟也都可以设置事件，且他们的执行也可以依赖于事件。

Suites的执行（Executing Suites）

在你已经创建和保存了suite之后，你真正执行它之前，你可以：

检查suite中的错误。

检查代理测试机（Agent computers）的状态。

控制suite的执行时间信息。

控制如何suite中止。

执行suite。

最后，在suite执行时，你可以监控suite的执行过程。

有关所有这些主题的信息，参阅87页*Executing Tests*的内容。

使用数据池（Working with Datapools）11

这一章描述如何创建和管理数据池。它包含了以下的主题：

- 什么是数据池
- 一个数据池的计划和创建
- 数据类型
- 数据池的管理
- 用户自定义数据类型的管理
- 数据池唯一行的生成和检索

- 在Rational Test之外创建数据池
- 在Rational Test之外创建列值

在你使用数据池之前，你应该使你熟悉本章介绍的概念和过程。

注意事项：这一章描述在一个TestManager suite中，VU和GUI测试脚本在回放时对数据池的访问和使用。有关数据池的附加信息，可以在许多不同的Rational文档中看到：

- 数据池过程，参阅Rational TestManager帮助。
- 在VB或者Java测试脚本中使用数据池，参阅合适的Rational Test Script Services API文档。
- 在常规测试脚本类型中使用数据池，参阅*Rational TestManager Extensibility Reference*手册。
- 在测试脚本的期间创建数据池，参阅*Using Rational Robot*手册和Robot帮助。
- 有关数据池和GUI测试脚本的信息，参阅*SQABasic Reference*手册。

什么是数据池（What Is a Datapool?）

“数据池”就是一个测试的数据集合。它为回放的测试脚本提供测试数据变量。

数据池使你可以自动地提供测试数据变量给虚拟测试者，这是在高容量条件下，即潜在的包含了执行几千个事务处理的几百个虚拟测试者。

一般情况下，你使用数据池可以：

- 每一个执行测试脚本的虚拟测试者可发送真实的测试数据（可以包含唯一的数据）到服务器。
- 多次数执行相同事务处理的单个虚拟测试者可发送针对每个事务处理的真实（通常是不同的）测试数据到服务器。

如果在回放期间不使用数据池，那么每个虚拟测试者会发送相同的字面值到服务器—该值是在测试脚本中定义的。

例如，假设你一个VU测试脚本以发送订单号53328到服务器。如果有100个虚拟测试者执行该测试脚本，那么该订单号53328会被发送到该服务器达100次。如果你使用数据池，那么每个虚拟测试者可发送不同的订单号到该服务器。

数据池工具（Datapool Tools）

可以使用Robot或TestManager来创建和管理数据池。下表展示了你使用这两种产品可以执行的活动：

Activity	Robot	TestManager
Automatically generate datapool commands in a test script.	•	
Create a datapool and automatically generate datapool values.	•	•
Edit the DATAPOOL_CONFIG section of a VU test script.	•	
Edit datapool column definitions and datapool values.	•	•
Create and edit datapool data types.		•

Activity	Robot	TestManager
Perform datapool management activities such as copying and renaming datapools.		•
Import and export datapools.		•
Import data types.		•

活动	**Robot**	**TestManager**
在测试脚本中自动地生成数据池命令。	•	
创建数据池并自动地生成数据池值。	•	•
编辑一个VU测试脚本的DATA_CONFIG部分。	•	
编辑数据池的列定义和数据池的值。	•	•
创建和编辑数据池的数据类型。		•
执行数据池的管理活动，比如复制和重命名数据池。		•
输入和输出数据池。		•

输入数据类型。		•

这一章论述数据池并阐明在TestManager中执行关联到数据池的活动。有关在Rational Robot中使用数据池的相关信息，参阅*Using Rational Robot*手册。

数据池文件的管理（Managing Datapool Files）

数据池由两种文件组成：文本文件和说明文件。

- 数据池的值可以保存在文本文件中，文件后缀名为.csv。
- 数据池的列名保存在说明文件中（文件后缀为.spc）。Robot或TestManager负责创建和维护该文件。但是，你不应该直接编辑该文件。

.csv和.spc文件被保存在你项目的TMS_Datapool的目录中。

当然，如果你输入一个数据池，Robot或TestManager自动地创建和管理.csv和.spc文件，但是，这些文件是基于你通过用户界面提供的指令（指导）来执行这些活动的。

如果你输入一个数据池，那么应该由你来创建.csv文件并填充数据。然而，Rational Test软件依然会负责创建和管理这个输入数据池的.spc文件。

有关输入数据池的信息，参阅278页*Importing a Datapool*和284页*Creating a Datapool Outside Rational Test*的内容。

注意事项：TestManager自动地复制一个.csv文件到每一个需要它的代理测试机。如果代理测试机的.csv文件变成out-of-date（过期数据），TestManager会自动地更新它。

数据池游标（Datapool Cursor）

数据池的“游标”，或行指针，可以在访问该数据池的所有虚拟测试者之间被共享，或者对于每个虚拟测试者可唯一。

在所有的虚拟测试者之间对数据池游标的共享，可以给与一个协调的测试。

因为数据池的每一行是唯一的，每个虚拟测试者可以共享相同的游标，并且依然可以发送唯一的到数据库。

此外，一个共享的游标可以不间断的跨过suite的执行。例如，假设在suite执行时访问的数据池最后的一行是第100行：

- 如果该游标是不间断的跨过suite的执行，那么数据池行的访问，在一个新的suite执行时，首次访问是从第101行恢复开始。
- 如果该游标不是连续的，那么在一个新的suite执行时，对数据池行的访问，首次从第1行重新开始。

注意事项：执行SQABasic测试脚本的虚拟测试者，可以在一个TestManager suite中回放时，共享一个游标，但不是在Robot中的回放发生时。

有关游标范围的定义，参阅*Using Rational Robot*手册中的**“游标”**主题的描述。

行访问顺序（Row Access Order）

行访问顺序是在测试执行期间被访问的行的顺序。

利用GUI测试脚本，你可以控制数据池游标的行访问顺序，使用SQABasic `SQADatapoolOpen` 命令的“顺序”变量。

利用VU测试脚本，你可以控制行访问顺序，通过在脚本对话框的Robot Configure Datapool 中**Access Order**的设置。（参阅*Using Rational Robot*手册。）

对于其他类型的测试脚本。参考合适的API文档。

数据池的界限（Datapool Limits）

一个数据池，如果是Robot或TestManger自动地为其生成数据，那么达到150列。如果，你是从一个数据库或其他的数据源输入一个数据池，那么可以达到32,768列。同样的，一个数据池可以达到2,147,483,647行。

数据池解决何种类型的问题?（What Kinds of Problems Does a Datapool Solve?）

如果在测试执行期间，你只回放一个测试脚本，那么这个测试脚本很可能就不要去访问数据池。但通常情况下，在测试执行期间，特别是在性能测试时，你需要多次执行相同的测试脚本。例如：

- 在性能测试时，你可能希望之行一个测试脚本的多个实例，以使该测试脚本同时地执行多次。（牢记，一个虚拟测试者是一个测试脚本的一个执行实例。）
- 在功能测试和性能测试期间，你经常会执行一个测试脚本的多个迭代，以使该测试脚本不间断地执行多次（模拟一个虚拟测试者重复地执行相同的任务）。

如果这个被使用在每个测试脚本实例和每个测试脚本迭代的值是相同的文字值—该值是在录制或手动输入到测试脚本中的值—你可能在suite执行期间遇到问题。

下面的这些问题的例子，可由数据池解决：

- 问题：在录制期间，你为一个新雇员创建一个人员文件，并使用了雇员的唯一标示：社会保险号。每次在测试脚本回放时，这里会试图创建相同的人员文件并提供相同的社会保险号。该应用程序拒绝了复制的请求。

  解决方案：使用一个数据池，每次在测试脚本回放时，发送包含唯一社会保险号的不同雇员

数据到该服务器。

- 问题：在录制期间你删除了一个记录。回放期间，测试脚本的每个实例和迭代试图删除相同的记录，并出现“Record Not Found”的错误结果。

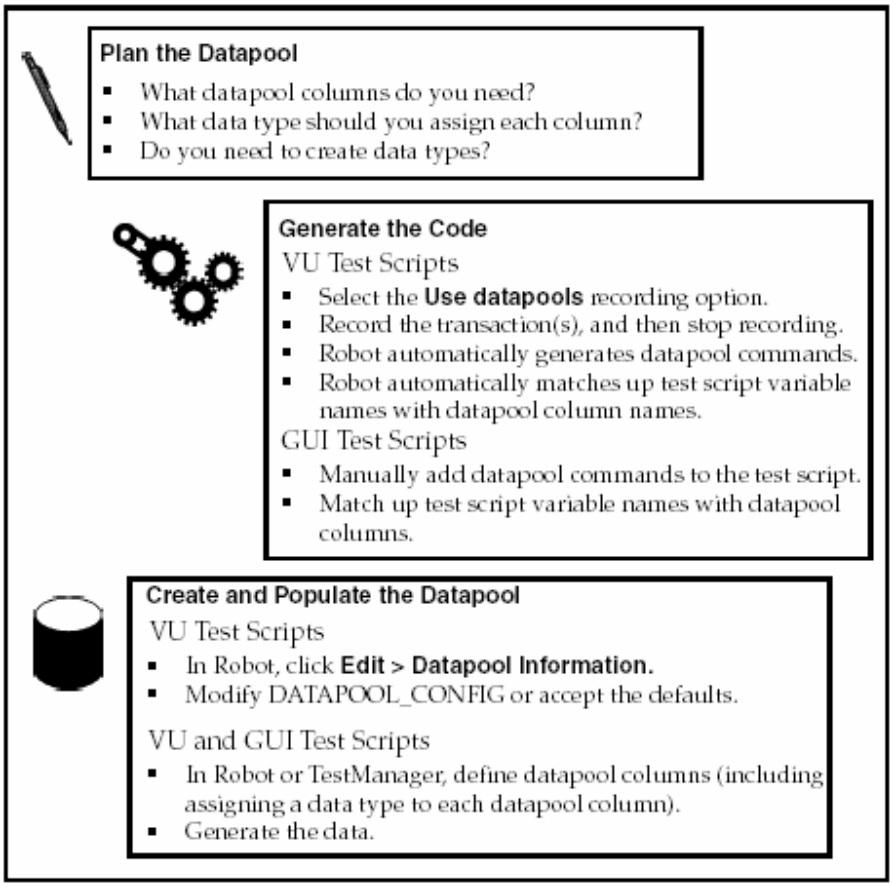
  解决方案：使用一个数据池，每次在测试脚本回放时，删除请求中参考不同的记录。

- 问题：针对一个性能测试，你录制一个测试脚本时，客户应用程序读取数据库的记录。在回放期间，那个相同的记录被读了几百次。因为客户应用程序是设计良好的，他在缓存中获得该记录，并在随后的数据取得中出现了快速检索的假象。这时，性能测试产生的响应时间是不准确的。

  解决方案：每次在测试脚本回放时，使用数据池来请求不同的记录。

数据池的计划和创建（Planning and Creating a Datapool）

该阶段的摘要涉及到在测试中使用数据池时，对其的准备工作，如下图说明。图中展示的该阶段顺序是对测试脚本所使用的数据池进行计划和创建的典型顺序。



以下的步骤提供了有关这些活动的细节：

1 计划数据池.

确定你需要的该数据池的列。换句话说，就是你希望从数据池检索并发送到服务器的那些值（名称，地址，日期，以及其他）的种类。

一般地，你需要为每一个测试脚本设置数据池列，可使你在录制脚本期间来变化的分配数据池的值。

例如，假设你的客户端程序有一个名为**Order Number**的字段。

在录制期间，你为这个字段输入一个值。VU测试脚本中，该值被自动地分配给一个测试脚本的变量。在回放期间，这个变量由一个数据池列分配成唯一的订单编号。

该阶段需要一些客户端应用程序的知识，以及处理的数据类型。

帮助你确定你需要的数据池的列，录制一个初步的脚本。Rational Robot自动地捕获所有的值，而这些值是在录制期间由客户端应用程序提供的，并且将它们列出在测试脚本的结束部分的DATAPOOL_CONFIG部分中。更多的信息，参阅266页*Finding Out Which Data Types You Need*

的内容。

2 生成数据池代码.

在运行时访问数据池，一个测试脚本必须包含数据池命令，比如打开数据池，和检索一行数据的命令。在VU测试脚本中，一个DATAPOOL_CONFIG部分也必须被指出。这一节包含了有关如何创建和访问数据池的信息。

数据池代码可利用以下这些方法产生：

- VU测试脚本，Robot在你完成一个session的录制时，会自动地生成数据池的代码。Robot可觉察到在录制期间分配到session中的所有值变量，并且在测试脚本中将这些变量的每一个与数据池的列相匹配。

  在录制期间使Robot自动地生成数据池命令，需要你在录制该测试脚本之前，确保已选择了Session Record Options对话盒**Generator**标签中**Use datapools**。

  注意事项：你依然需要为数据池提供数据，并使其有效。
- SQABasic测试脚本，你需要手动地输入输入数据池命令，当然将测试脚本变量与数据池列相匹配也是手动地。有关数据池编写数据池命令的信息，参阅*Using Rational Robot*手册。
- 有关在其他类型的测试脚本中使用数据池的信息，参阅合适的Rational Test Script Services API文档。

3 生成和填充数据池.

数据池命令存在于测试脚本中之后，你就可以创建和填充该数据池了。

针对一个你在Robot中编辑的VU测试脚本，开始创建和填充一个数据池，点击**Edit > Datapool Information**。

为一个测试脚本创建和填充一个数据池，包含以下的步骤：

- 编辑测试脚本的DATAPOOL_CONFIG部分。例如，你可能希望改变默认的数据池访问标志，或者针对一个特殊的测试脚本变量排除一个数据池列。或者，在一个VU测试脚本中，Robot创建这部分的时候，你可以接受所有的这些Robot指定的默认设置。

  有关编辑一个测试脚本的DATAPOOL_CONFIG部分的信息，参阅*Using Rational Robot*手册。
- 在阶段的计划期间，定义你确定需要的数据池列。例如，对于一个订单编号列，你可以说明一个订单编号最大的字符数量，并且，该订单编号列是否必须包含唯一的值。

  有关数据池列的信息，参阅*Using Rational Robot*手册。

  你也可以为每一个数据列指定一个数据类型。数据类型向一个数据池列提供它的值。有关数据类型的信息，参阅264页*Data Types*的内容。

- 生成数据。一旦你配置数据池，并定义了它的列，你就可以通过点击**Generate Data**来填充这个数据池了。

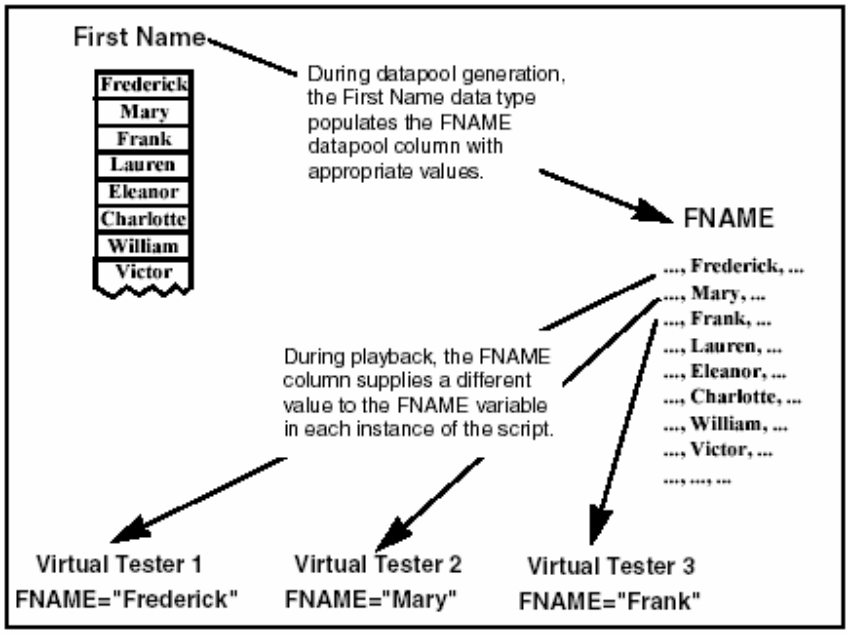
注意事项：你也可以手动地创建和填充一个数据池文件，并将其输入到datastore。更多的信息，参阅284页*Creating a Datapool Outside Rational Test*的内容。

数据类型（Data Types）

数据池的“数据类型”是数据池列的一个数据源。

例如，Names – First数据类型（发送时在Rational Test中作为一个标准数据类型）包含普通的英文姓名的一个列表。假设你分配这种数据类型到数据池的FNAME列。当你生成该数据池时，TestManager以Names – First数据类型给FNAME列填充所有的值。

下图展现了数据类型之间的关系，数据池列，以及在回放期间分配给测试脚本变量的值：



标准数据类型和用户定义的数据类型（Standard and User-Defined Data Types）

这里有两种数据池的数据类型，如下：

- **标准数据类型** Rational Test带有的。这些数据类型包含通常的用法，真实的一类数据类型，比如名和姓，公司名称，城市，以及号码。

For a list of the standard data types, see *Standard Datapool Data Types* on page 341

对于标准数据类型列表，参阅341页*Standard Datapool Data Types*的内容。.

- 用户定义的数据类型 由你创建。如果标准数据类型没有包含你希望提供给测试脚本变量的一类值，那么你必须创建一个数据类型。

  用户定义的数据类型用于以下这些情况：

  - 当一个字段接受一定数量的合法值时。例如，假设一个数据池列为测试脚本名为“Color”变量提供数据。该变量为服务器提供定购的产品颜色。如果该产品只有红，绿，蓝，黄，和棕这几种颜色，那么这些值被分配到“color”列中。标准数据类型不包含这些确切的值。

    确认该变量由数据池分配了一个有效值：

    i 在你创建数据池之前，创建名为Colors的数据类型，它包含5个相应值（`Red`，`Green`，`Blue`，`Yellow`，`Brown`）

    ii 当你创建数据池时，分配这个Colors数据类型到数据池列COLOR。COLOR列将为测试脚本的“color”（颜色）变量提供值。

  - 当你需要生成多字节字符的数据时，比如那些使用的外文字符集。更多的信息，参阅268页*Generating Multi-Byte Characters*的内容。

创建数据池之前，搞清作为数据源使用的标准数据类型和需要你创建的用户定义的数据类型。虽然在你创建数据池本身的同时来创建数据类型是可能的，但如果你首先创建的是用户定义的数据类型，那么创建数据池的过程将会更加平稳。

搞清你需要的数据类型（Finding Out Which Data Types You Need）

确定是否要分配一个标准数据类型或一个用户定义的数据类型到数据池的每一列，你需要了解在回放期间提供给测试脚本变量的值类型—例如，该变量是否包含姓名，日期，订单号码，等等。

找出为一个变量提供的值类型：

- 在VU测试脚本中，查看测试脚本的`DATAPOOL_CONFIG`部分。（在录制期间，Robot自动地添加该信息到一个VU测试脚本中，前提是你在Session Record Options对话框的**Generator**标签中选择了**Use datapools**。）

  `DATAPOOL_CONFIG`部分在录制期间包含一行数据值，这些值被分配到一个测试脚本变量。在下面的例子中，值`329781`被分配到变量*CUSTID*：

  ```
  INCLUDE, "CUSTID", "string", "329781"
  ```

有关测试脚本`DATAPOOL_CONFIG`部分，参阅*Using Rational Robot*手册。

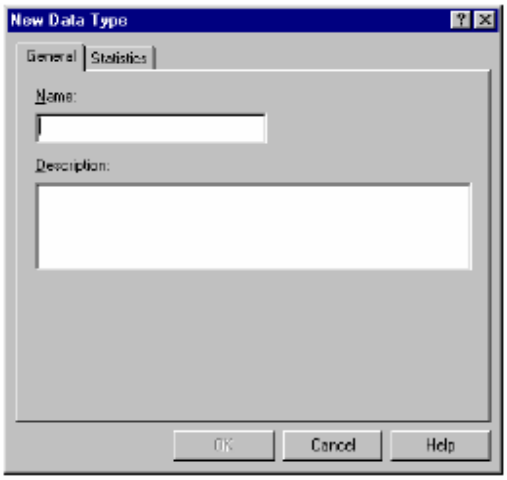
创建用户定义的数据类型（Creating User-Defined Data Types）

如果标准数据类型不能为一个测试脚本变量提供正确的值类型，那么你可以创建一个用户定义

的数据类型。

创建一个用户定义的数据类型：

- 点击**Tools > Manage > Data Types**，并点击**New**。



当你创建一个用户定义的数据类型时，它被罗列在Datapool Specification对话框（你定义数据池列的地方）的**Type**栏中。**Type**也包括所有的标准类型的名称。用户定义的数据类型在该列表中有一个星号（*）标志。

注意事项：你可以指定数据由一个标准数据类型到一个用户定义的数据类型。有关信息，参阅280页*Editing User-Defined Data Type Definitions*的内容。

由用户定义的数据类型生成唯一值（Generating Unique Values from User-Defined Data Types）

在回放期间，你可能希望一个用户定义数据类型为测试脚本变量提供唯一的值。要这样做，该用户定义的数据类型必须包含唯一值。

此外，当你在Datapool Specification对话框中定义该数据池时，使以下针对数据列的设置关联到用户定义的数据类型：

- 设置**Sequence**为“Sequential” （连续的）。
- 设置**Repeat**为1。
- 确认**No. of records to generate**值在你的用户定义数据类型中不超过唯一值的数目。

有关在对话框中设置值的信息，参阅271页*Defining Datapool Columns*的内容。

生成多字节字符（Generating Multi-Byte Characters）

如果你希望在你的数据池中包含有多字节字符（例如，要支持日语或其他的包含了多字节字符的语言），你可以用下面的这些方法的任一个来实现：

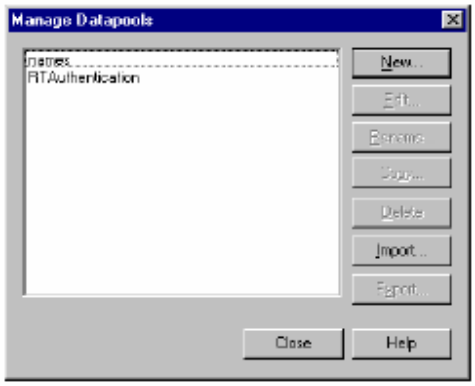
- 通过一个用户定义的数据类型。有关信息，参阅266页*Creating User-Defined Data Types*的内容。

  编辑器为用户定义的数据提供全面的支持Input Method Editor（输入法编辑器）的操作。IME使你可以由标准的键盘输入多字节字符，比如Kanji和Katakana字符，以及多字节的ASCII。IME包含了Microsoft Windows的日语版本。

- 通过从文件数据类型中读取。有关信息，参阅288页*Creating a Column of Values Outside Rational Test*的内容。

数据池的管理（Managing Datapools）

由Manage Datapools来管理数据池。在这个对话框中执行的活动影响在当前datastore中的数据池的存储。有关数据池存储地方的信息，参阅278页*Datapool Location*的内容。



创建一个数据池（Creating a Datapool）

在使用TestManager创建数据池时，你必须说明以下的内容：

- 该数据池的名称和描述.

  选择一个可达40个字符的名称。当一个描述为可选的时，输入1可以帮助你指明该数据池的目的。数据池的描述最多可达255个字符。

- 列名.

  数据池的列也被称为字段。在VU测试脚本中，数据池的列名必须与测试脚本变量的名称相匹配。

  名称是用例敏感的（case-sensitive）。

- 每一个数据池列的数据类型和属性.

  有关定义数据池列属性的信息，参阅271页*Defining Datapool Columns*的内容。

- 产生记录的数目。

创建并自动地填充一个数据池：

- 点击**Tools > Manage > Datapools**，并点击**New**。

如果存在错误（If There Are Errors）

如果该数据池的值没有被成功地生成，那么TestManager会询问你是否希望查看一个错误报告，而不是一个生成数据的摘要。查看这个报告，可以帮助你确定数据池字段中有哪些需要改正。

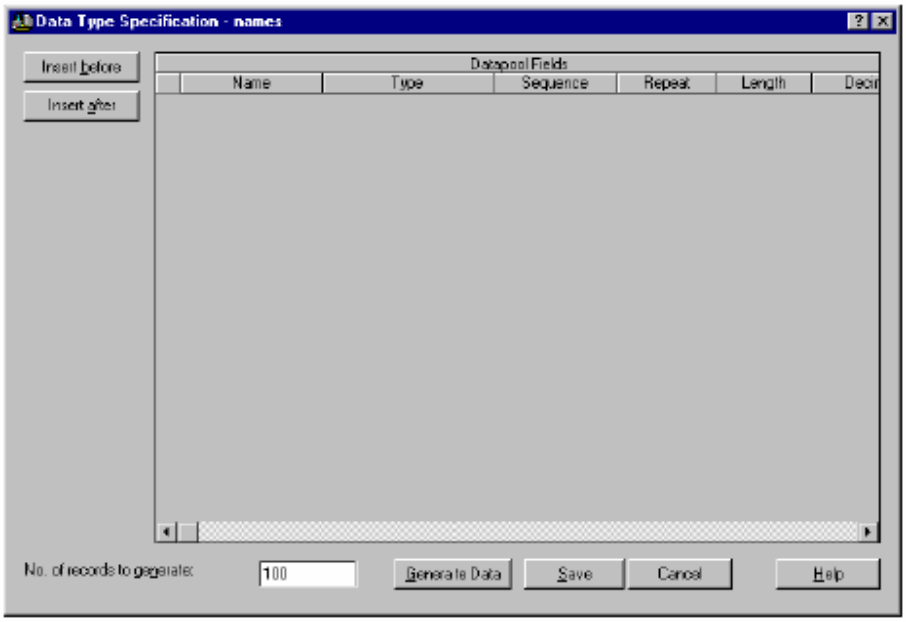
要查看一个错误报告，在TestManager询问你是否需要查看一个错误报告或者摘要数据的时候，点击**Yes**。

查看数据池的值（Viewing Datapool Values）

如果一个数据池包含了复杂的值（例如，嵌套字符串，或字段分隔符等包含在数据池值中字符），你应该查看数据池的值，以确保该数据池的内容是你所期望的。

要查看产生的值：

- 在Manage Datapools对话框中，选择你刚才创建的这个数据池，点击**Edit**，然后点击**Edit Datapool Data**。



使数据池可用于一个测试脚本（Making the Datapool Available to a Test Script）

对于一个可以访问数据池的测试脚本而言，该测试脚本必须包含数据池命令，比如打开数据池和检索数据值的命令。VU测试脚本也必须含有DATAPOOL_CONFIG。

你可以添加数据池命令和DATAPOOL_CONFIG到一个测试脚本，当然，这可以是在你用TestManager创建这个数据池之前或者之后。有关在录制期间自动地添加数据池命令和DATAPOOL_CONFIG到一个测试脚本中的信息，参阅*Using Rational Robot*手册。

定义数据池的列（Defining Datapool Columns）

使用下表，可帮助你在Datapool Specification对话框中定义数据池列：

Grid column	Description
Name	The name of a datapool column (and its corresponding test script variable). If you change the name of a datapool column, be sure that the new name matches all instances of its corresponding test script variable. If you create a datapool outside of the Rational Test environment and then import it, TestManager automatically assigns default names to the datapool columns. Use **Name** to match the imported datapool column names with their corresponding test script variables. Names are case-sensitive. You can use an IME to type multi-byte characters in datapool field names.
Type	The standard or user-defined data type that supplies values to the datapool column in **Name**. User-defined data types are marked with an asterisk (*). Specify the data type to assign to the datapool column, as follows: ▪ To select a standard data type or an existing user-defined data type, click the currently displayed data type name, and then select the new data type from the drop-down list:  ▪ See Appendix B for a description of the standard data types. ▪ If you type rather than select the name of a user-defined data type, enter an asterisk before the user-defined data type name. For example, to specify the user-defined data type MyData, type: `*MyData` ▪ To create a new user-defined data type, enter the data type name (without the asterisk) in the field, and then press RETURN. After you click **Yes** to confirm that you want to create a user-defined data type, the Data Type Properties - Edit dialog box appears. ▪ For information about creating a data type, see *Creating User-Defined Data Types* on page 266.

Grid column	Description
Sequence	The order in which the values in the data type specified in **Type** are written to the datapool column. Select one of these options from the drop-down list: ▪ **Random** – Writes numeric and alphanumeric values to the datapool column in any order. ▪ **Sequential** – Writes numeric values sequentially (for example, 0, 1, 2...). With decimal numbers, the sequence is based on the lowest possible decimal increment (for example, with a **Decimals** value of 2, the sequential values are 0.00, 0.01, 0.02, ...). ▪ **Sequential** is only supported for numeric values (including date and time values) and values generated from user-defined data types. When you select Sequential with numeric data types, and you specify a **Minimum** and **Maximum** range, **Interval** must be greater than 0. ▪ **Unique** – With data type Integers - Signed, ensures that numbers written to the datapool column are unique. Also, set **Repeat** to 1, and define a **Minimum** and **Maximum** range. Do not confuse the **Random** and **Sequential** settings in this grid with **Random** and **Sequential** access order in the Configure Datapool in Script dialog box.The **Random** and **Sequential** settings in this grid determine the order in which values are written to an individual datapool column at datapool creation time. **Random** and **Sequential** access order determine the order in which virtual testers access datapool rows at suite runtime.
Repeat	The number of times a given value can appear in a datapool column. **Repeat** cannot be set to 0. To make values unique with Integers - Signed data types and user-defined data types, set **Repeat** to 1. For unique Integers - Signed values, also set **Sequence** to either Sequential or Unique. When defining unique values, make sure that the number of rows you are generating is not higher than the range of possible unique values.
Length	The maximum number of characters that a value in the datapool column can have. If the datapool column contains numeric values, **Length** specifies the maximum number of characters a number can have, *including* a decimal point and minus sign, if any. For example, for decimal numbers as high as 999.99, set **Length** to 6. For decimal numbers as low as -999.99, set **Length** to 7. **Length** cannot be 0.
Decimals	Specifies the maximum number of decimal places that floating point values can have. Maximum setting is 6 decimal places.

Grid column	Description
Interval	Writes a sequence of numeric values to the datapool column. The sequence increments by the interval you set. For example, if **Interval** is 10, the datapool column contains 0, 10, 20, and so on. If **Interval** is 10 and **Decimal** is 2, the datapool column contains 0.00, 0.10, 0.20, and so on. Minimum interval is 1. Maximum interval is 999999. With numeric data types (including dates and times), when **Sequence** is set to Sequential and you specify a **Minimum** and **Maximum** range, **Interval** must be greater than 0. Use **Interval** only with numeric values (including dates and times).
Minimum	Specifies the lowest in a range of numeric values. For example, if the datapool column supplies order number values, and the lowest possible order number is 10000, set **Type** to Integer - Signed, **Minimum** to 10000, and **Maximum** to the highest possible order number. Use **Minimum** only with numeric values (including dates and times).
Maximum	Specifies the highest in a range of numeric values. For example, if the datapool column supplies values to a variable named *ounces*, set **Type** to Integer - Signed, **Minimum** to 0, and **Maximum** to 16. Use **Maximum** only with numeric values (including dates and times).
Seed	The number that Rational Test uses to compute random values. The same seed number always results in the same random sequence. To change the random sequence, change the seed number.
Data File	The path to the user-defined data type file. The path is automatically inserted for you. This field is not modifiable. Data type files are stored in the Datatype directory of your project. You never have to modify these files directly.

表格列	**描述**
Name	数据池的列名称（它对应于测试脚本变量）。 如果你改变一个数据池的列，那么需要确保新的名称与它对应的测试脚本变量的所有实例相匹配。 如果你是在Rational Test环境之外创建了一个数据池，并将其输入，那么TestManager会自动地为这个数据池的列指定默认名称。使用“Name”将输入的数据池的列名与它们对应的测试脚本变量相匹配。名称是用例敏感的（case--sensitive）。 你可以使用IME在数据池字段名中输入多字节字符。

Type	标准和用户定义的数据类型为Name栏中的数据池列提供值。用户定义的数据类型用星号（*）标示。 说明数据池列指定的数据类型，如下： ● 选择一个标准数据类型或者一个现存的用户定义的数据类型，点击当前显示的数据类型名称，然后从下拉列表中选择新的数据类型：  ● 参阅附录B有关标准数据类型的描述。 ● 如果你是输入而非选择了一个用户定义的数据类型的名称，那么在该用户定义的数据类型输入名称之前，输入一个星号。例如，说明一个用户定义的数据类型MyData，输入： *MyData ● 要创建一个新的用户定义的数据类型，在字段中输入该数据类型的名称（不带星号），然后按RETURN。在你点击Yes以确认你希望创建的用户定义的数据类型之后，Data Type Properties—Edit对话框出现。 ● 创建一个数据类型的有关信息，参阅266页Creating User-Defined Data Types的内容。
Sequence	在Type中被说明的数据类型的值按顺序被写入数据池列。可从下拉列表中选择这些选项： ● Random（随机）--以任意顺序将数字与字母的值写入数据池列。 ● Sequential（顺序）--顺序地写入数值（例如，0，1，2……）。对于小数，该顺序是按照可能的最低位增加（例如，有一个小数为2位小数，则顺序值是0.00，0.01，0.02，……）。 ● Sequential只支持数值（包括日期和时间值）以及由用户定义的数据类型产生的值。 当你选择Sequential时，你要指定一个最小和最大的范围，并且Interval（间隔）必须大于0。 ● Unique（唯一）--数据类型为有符号整数，确保被写入数据池列的数是唯一的。并设置Repeat为1，定义一个最小和最大的范围。 不要将现在所说的表格中Random和Sequential设置与Script 对话框中ConfigureDatapool的Random和Sequential访问顺序相混淆。Random和Sequential设置决定了数据池创建时，值被写入到独立的数据池列的顺序。Random和Sequential访问顺序决定了suite执行时，虚拟测试者访问数据池行的顺序。
Repeat	一个给定值出现在一个数据池列中的次数。 Repeat不能被设置为0。 要使整数值唯一—有符号数据类型和用户定义的数据类型，设置Repeat为1。对于唯一整数—有符号，并设置Sequence为Sequential或Unique。 当定义唯一值时，确认创建的行数不高于可能的唯一值的范围。
Length	在数据池列中能达到的字符值的最大位数。如果数据池列包含数值，Length说明了能够达到的字符的最大位数，如果有的话，其中还包含了小数点和负号。

	例如，小数999.99，它的长度设置为6。 而对于小数-999.99，它的长度设置为7。 Length不能为0。
Decimals	说明小数位浮点值的最大位数。最大设置为6位小数。
Interval	写一列数值到数据池列。该序列按照你设置的间隔增加。例如，如果Interval为10，则数据池列包含0，10，20，等。如果间隔为10 ，小数为2，则该数据池的列包含0.00，0.10，0.20，等。 最小间隔为1。最大间隔为999999。 数字类型（包括日期和时间），当**Sequence**被设置为Sequential，并且你指定了最小和最大的范围，Interval必须大于0。 只有数值（包括日期和时间）使用Interval（间隔）。
Minimum	指定数值的最低范围。例如，如果数据池列提供订单编号值，而且最小可能的订单编号为10000，设置Type为有符号整数，则Minimum为10000，且Maximum为最大可能的订单编号。 只有数值（包括日期和时间）使用Minimum（最小值）。
Maximum	说明数值的最大范围。例如，如果数据池列提供一个名为ounces（盎司）的变量值，设置Type为有符号整数，则最小值为0，最大值为16。 只有数值（包括日期和时间）使用Minimum（最大值）。
Seed	Rational Test用户使用的数字以计算随机值。 相同的seed数字一般会导致相同的随机顺序。 要改变随机顺序，就要改变seed数字。
Data File	用户定义的数据类型的文件路径。该路径自动插入。该字段不能被修改。 数据类型文件被保存在项目的Datatype目录中。 你不能直接地修改这些文件。

这里有一些项也存在不能被修改的情况，这依赖于你选择的数据类型。例如，如果你选择Names – First的数据类型，那么你就不能修改**Decimals**，**Interval**， **Minimum**， 或**Maximum**等项。如果你为有符号整数的数据类型生成了唯一的值，那么**Length**，**Minimum**，**Maximum**，和**No. of records to generate**（生成的记录编号）必须是相容的（一致的）。例如，如果你希望唯一的数字从0到999，那么**Length**设置为1，**Maximum**设置为5000，或者**No. of records to generate**（生成的记录编号）数目大于1000，都会引起错误。

注意事项：你可以使用IME来输入多字节字符，但它只能被输入到Name列中。

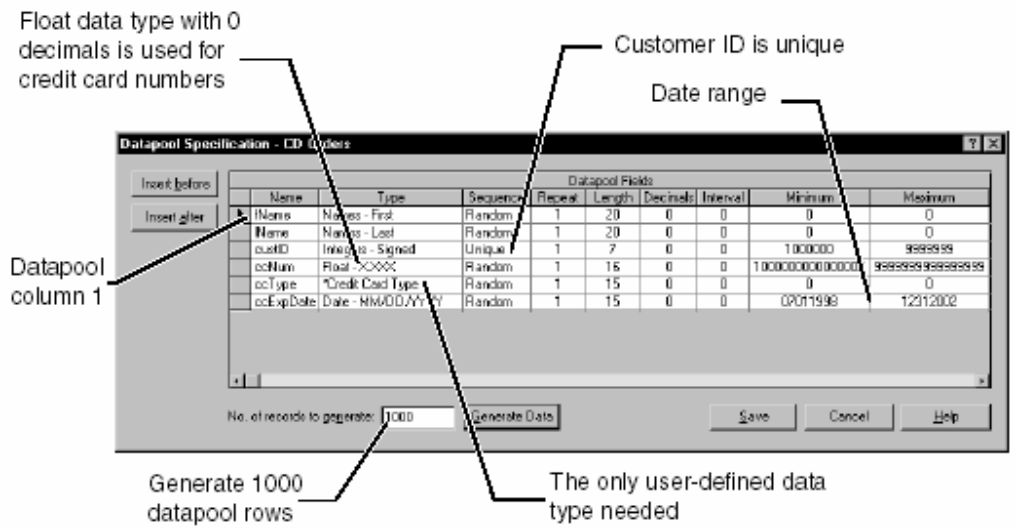
当你编辑其他列时，IME会自动停止（不可用）。

例1：定义数据池列（Example of Datapool Column Definition）

假设你希望录制顾客购买的这样一个事物处理，并输入到数据库中。在录制期间，你向客户端应用程序提供以下的有关顾客的基本信息：

- 顾客姓名
- 顾客的ID
- 信用卡号码
- 信用卡类型
- 信用卡终止日期

在你录制了这个测试脚本之后，创建数据池。在Datapool Specification对话框中定义该数据池的列，如下图所示：



以下是一些你选择的该数据池的列：

- **fName**列. 标准数据类型Names – First，为这个数据池列提供男性和女性的名称。
- **lName**列. 标准数据类型Names – Last，为这个数据池列提供姓氏。
- **custID**列. 标准的有符号整数型，为该数据池列提供ID号码。在此例中，由于所有的顾客ID号码都是7位数字，所以**Minimum**和**Maximum**的范围设置为1000000到9999999。而且，所有ID号码必须唯一，所以**Sequence**被设置为Unique（唯一）。

  注意事项：对于有符号的整数型，**Sequence**只能设置为Unique（唯一）。
- **ccNum**列. 有符号的整数型，生成为多达9位的数字。

  注意事项：由于信用卡号码包含大于9位的数字，所以标准数据类型Float（浮点）X.XXX常被使用向该数据池列提供信用卡号码。
- **Decimals**设置为0，仅产生整数。**Sequence**设置为Random，以生成随机的信用卡号。要产

生唯一的号码，**Repeat**设置为1。

- **ccType**列. 这是仅有的需要由用户定义的数据类型提供值的列。用户定义数据类型Credit Card Type只包含4个值—American Express，Discover，MasterCard和Visa。
- **ccExpDate**列. 标准的日期数据类型—MM/DD/YYYY，向数据池的该列提供信用卡的终止日期。有效的终止日期范围设置为：7月1日1998到12月31日2002。**Sequence**设置为Random，以产生随机的日期。

例2：生成数据池的值（Example of Datapool Value Generation）

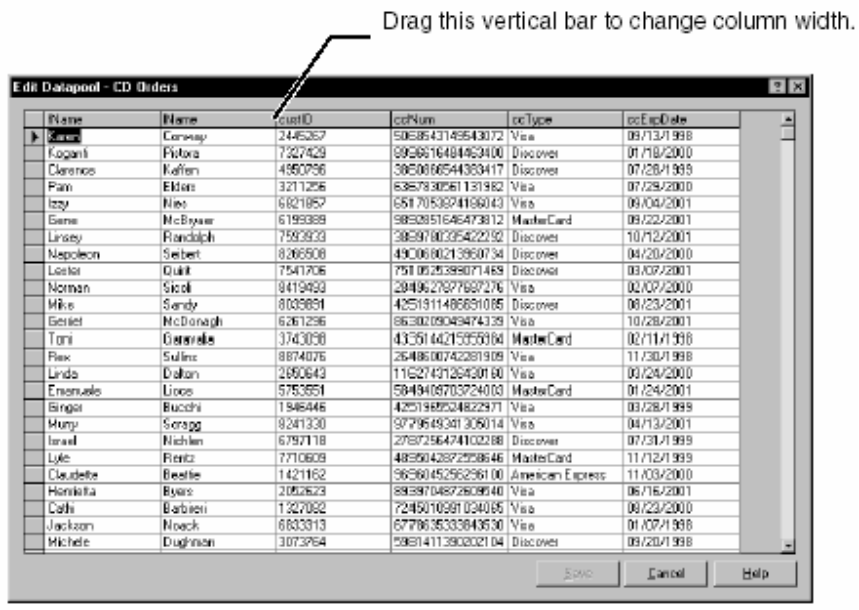
在Datapool Specification对话框中定义了数据池列之后，点击**Generate Data**以产生该数据池的值。

要查看你生成的值：

1 点击**Close**。

2 点击**Edit Datapool Data**。

你将看到：



编辑数据池列的定义（Editing Datapool Column Definitions）

Datapool Specification对话框允许你定义并编辑数据池文件中的列。数据池列的定义以行的形式罗列在该对话框中。

数据池列也被称为“字段”。

要编辑一个现存的数据池列的定义：

- 点击**Tools > Manage > Datapools**，选择该数据池，进行编辑，然后点击**Edit**。

当你完成数据池列的编辑时，选择是否要为该数据池生成数据。

查看产生的值：

- 在Datapool Properties - Edit对话框中，点击**Edit Datapool Data**。

如果该数据池的值不能成功地产生，那么TestManager会询问你是否希望查看一个错误报告。查看该错误报告，可以帮助你确定需要更正的该数据池的字段。要查看一个错误报告，在TestManager询问你是否需要查看一个错误报告或摘要数据时，点击**Yes**。

删除数据池列（Deleting a Datapool Column）

数据池列的定义以行的形式罗列在该对话框中。要从该列表中删除一个数据池列，选择要删除的行，并按删除键。

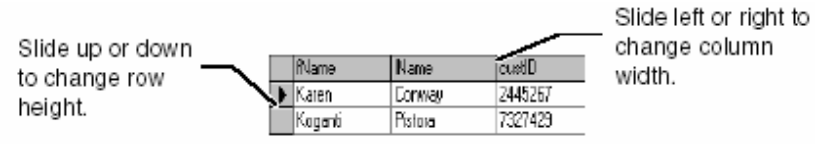
编辑数据池的值（Editing Datapool Values）

要查看或编辑一个现存的数据池的值：

- 点击**Tools > Manage > Datapools**，选择该数据池，进行编辑，并点击**Edit**。

在数据池的值时，要注意：

- 当你点击一个值进行编辑时，一个箭头图标会出现在你正在编辑的那一行的左侧。
- 当你开始编辑该值时，一个笔状图标会出现在那一行的左侧，指明编辑状态。
- 要取消变更，在你将插入点移出该字段之前，按CTRL + Z。
- 要查看编辑菜单，选择文本，进行编辑，然后点击鼠标右键。
- 增加列宽，移动单个的列名竖线。
- 增加行高，移动单个的行线。



有关TestManager产生数据池值的相关示例信息，参阅275页*Example of Datapool Value Generation*的内容。

重命名或复制一个数据池（Renaming or Copying a Datapool）

当你重命名或复制一个数据池时，你必须为该数据池指定一个新的名称，且最多为40个字符。

重命名或复制一个数据池：

- 点击**Tools > Manage > Datapools**。

删除数据池（Deleting a Datapool）

删除一个数据池，移除该数据池的.csv和.spc文件，以及所有的从datastore而来的对该数据池的参照。

删除一个数据池：

- 点击**Tools > Manage > Datapools**。

输入一个数据池（Importing a Datapool）

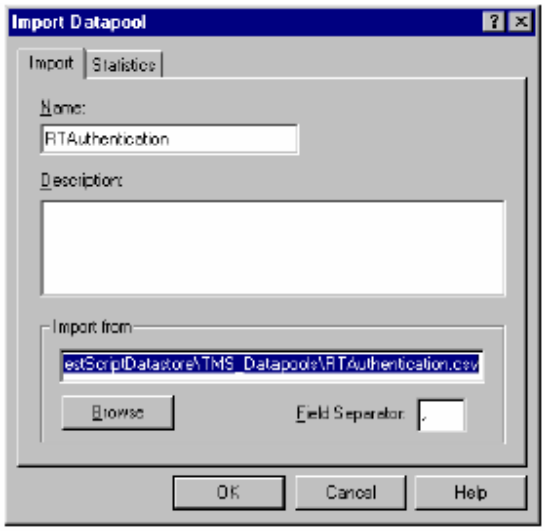
你自己可以创建并填充一个数据池，使用像是Microsoft Excel这样的工具。例如，你可能希望从你的数据库中输出数据到一个.csv文件中，然后使用这个文件作为你的数据池。

如果你自己创建了一个数据池，那么你需要输入它到相同的datastore中，作为测试脚本将会访问的数据池。你可以使用TestManager输入一个数据池的.csv文件。

当你输入一个数据池时，你经常会去改变该数据池的列名，以使其与对应的测试脚本变量相匹配。更多的信息，参阅287页*Matching Datapool Columns with Test Script Variables*的内容。

输入一个数据池的.csv文件：

- 点击**Tools > Manage > Datapools**，然后点击**Import**。



数据池的定位（Datapool Location）

当你输入一个数据池时，TestManager复制该数据池的.csv文件，使Datapool目录关联与当前的项目和datastore。

例如，如果当前项目为MyProject，且当前的datastore目录为MyDatastore，那么给数据池被保存在下面的目录中：

C:\MyProject\MyDatastore\DefaultTestScriptDatastore\TMS_Datapool

该目录也包括了这个数据池的说明（.spc）文件。当你创建并输入一个.csv文件时，TestManager

会自动地为你创建.spc文件。

你不能直接地去编辑这个.spc文件。

注意事项：在你输入一个数据池之后，保存源文件时，你要指定数据池在该目录中。Rational Test软件没有对这种文件更进一步的需求。

从另一个项目输入一个数据池（Importing a Datapool from Another Project）

你可以使用TestManager的Import特点，来复制一个你为另一个项目创建的数据池。当你输入一个数据池到一个新项目时，源数据池对于原项目依然是可用的。

输入一个数据池到一个新的项目中：

- 点击**Tools > Manage > Datapools**，然后点击**Import**。

注意事项：如果该数据池中包括了用户定义的数据类型，那么要在输入该数据池之前输入该数据类型。有关信息，参阅281页*Importing a User-Defined Data Type*的内容。

输出一个数据池（Exporting a Datapool）

你可以使用TestManager 的Export特点，来复制一个数据池到任意的目录下，当然是符合你的计算机的目录结构的。当你输出一个数据池时，源数据池会留在他的项目和datastore中。

不要试图输出一个数据池到另一个Rational Test项目中。相反的，使用import特点可以输入该数据池到一个新的项目中。更多的信息，参阅278页*Importing a Datapool*的内容。

在你的计算机上输出一个数据池：

- 点击**Tools** > **Manage > Datapools**。

对用户定义数据类型的管理（Managing User-Defined Data Types）

使用TestManager管理用户定义的数据类型。你可以编辑数据类型的值和数据类型的定义。也可以重命名，复制，和删除数据类型。

有关创建用户定义的数据类型的信息，参阅264页*Data Types*的内容。

编辑用户定义数据类型的值（Editing User-Defined Data Type Values）

如果你希望添加，移除，或者修改数据类型的值，或者只想修改可选的描述，编辑该数据类型。你只能编辑用户定义的数据类型，而不是标准的数据类型。

编辑一个用户定义的数据类型：

- 点击**Tools > Manage > Data Types**。

编辑用户定义数据类型的定义（Editing User-Defined Data Type Definitions）

如同所有的数据类型一样，一个用户定义的数据类型是一种重要的单列数据池。这个单列包含了你输入到该用户定义的数据类型中的值。

你可以在Datapool Specification对话框中编辑数据类型列的默认定义，就如同你编辑数据池列的默认属性一样。

如果你编辑一个用户定义的数据类型的定义，然后为该类型产生值，那么你需要覆盖该类型的所有已存在的值。

编辑一个用户定义的数据类型：

- 点击**Tools > Manage > Data Types**。

你也可以给一个用户定义的数据类型添加值，通过向其提供来自一个标准数据类型的值。通过TestManager自动产生的值，可以减少手工的输入量。

例如，假设你希望创建包含一个有效产品ID号的用户定义的数据类型。这个有效的ID号码的范围是1000001到1000100。

然而，在这里的第四位和第五位之间存在一个“-”（破折）号。

而不是输入所有的100个号码，需要加上破折号，你可以用TestManager生成这些号码并指定他们为一个用户定义的数据类型。然后，你可以编辑该数据类型的值和每一个ID号。

在你选择了自动生成值的同时，你必须说明TestManager在生成数据时使用的引导值。这些值包括：

- **Type** = Integers – Signed（有符号整数）
- **Sequence** = Sequential（连续的）
- **Repeat** = 1
- **Length** = 7
- **Interval** = 1
- **Minimum** = 1000001
- **Maximum** = 1000100
- **No. of records to generate**

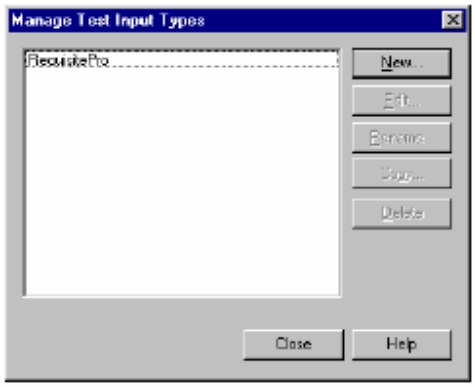
注意事项：你也可以指定标准数据类型Read From File到一个用户定义的数据类型。有关使用Read From File数据类型的信息，参阅288页*Creating a Column of Values Outside Rational Test*的内容。

输入一个用户定义的数据类型（Importing a User-Defined Data Type）

你可以从一个项目输入一个用户定义的数据类型到另一个中去。当你输入一个用户定义的数据类型到一个新的项目中去时，源数据类型依然适用于原项目。

输入一个用户定义的数据类型到一个新的项目中：

- 点击**Tools > Manage > Test Input Types**。



重命名或复制一个用户定义的数据类型（Renaming or Copying a User-Defined Data Type）

当你重命名或者复制一个用户定义的数据类型时，你必须为该数据类型指定一个新的名称，最多可达40个字符。

重命名或复制一个用户定义的数据类型：

- 点击**Tools > Manage > Data Types**。

删除一个用户定义的数据类型（Deleting a User-Defined Data Type）

在TestManager中删除一个用户定义的数据类型：

- 点击**Tools > Manage > Data Types**。

生成并检索唯一的数据池行（Generating and Retrieving Unique Datapool Rows）

很多的数据库测试工作，最好每行有唯一的测试数据。例如，如果一个测试包含的虚拟测试者是添加顾客的定购信息到数据库，那么每一个定购应该是唯一的—换句话说，在新记录中至少有一个字段是必须是一个“键”字段，它具有唯一的数值。

当你在Datapool Specification对话框中定义数据池列时，说明一个给定的数据池列是否应该包含唯一的数值。如果你指定一个或者多于一个的数据池列包含唯一的数值，那么这个由Rational

Test软件产生的数据池保证会包含唯一的行。

然而，即便是一个数据池包含了全部的唯一行，它在一个测试脚本执行时，也有可能为其提供重复的行。

生成和检索唯一的数据池行，你需要在你定义该数据池时执行一些简单的工作。

在数据池被单个的或者多个测试脚本（包含VU和GUI测试脚本）访问时，使用下面的引导规则。

保证唯一行的检索（What You Can Do to Guarantee Unique Row Retrieval）

确保一个数据池为测试脚本在执行时提供唯一的（数据）行，可参照这些引导规则：

What to do	How to do it
Specify at least one column of unique data.	In the Datapool Specification dialog box, specify that at least one datapool column should contain unique data. Unique data can be supplied through the Integers - Signed data type, through the Read From File data type, and through user-defined data types. With the Integers - Signed data type, take all of these actions: ▪ Set **Sequence** to Unique or Sequential. ▪ Set **Repeat** to 1. ▪ If **Sequence**=Unique, set an appropriate range in **Minimum** and **Maximum**. ▪ Make sure that the values of **Length** and **No. of records to generate** are appropriate for the set of numbers to generate. With the Read From File data type, see *Generating Unique Values* on page 289 for information. With user-defined data types, see *Generating Unique Values from User-Defined Data Types* on page 267 for information.
Generate enough datapool rows.	Generate at least as many unique datapool rows as the number of times the datapool will be accessed during a test. For example, if 50 virtual testers will access a datapool during a test, and each virtual tester is set for 3 iterations each, the datapool must contain at least 150 rows. You specify the number of rows to generate in the **No. of records to generate** field of the Datapool Specification dialog box.
Disable cursor wrapping.	If the datapool cursor wraps after the last row in the datapool has been accessed, previously fetched rows are fetched again. Disable cursor wrapping in any of these ways: ▪ When editing the `DATAPOOL_CONFIG` section of a VU test script in the Configure Datapool in Script dialog box, set **Wrap at end of file?** to **No**. ▪ When editing a VU test script in Robot, add `DP_NOWRAP` to the list of flags in the *`flags`* argument of the `DATAPOOL_CONFIG` statement or the `datapool_open` function. ▪ When editing a GUI test script in Robot, set the *`wrap`* argument of the `SQADatapoolOpen` command to **False**.

What to do	How to do it
Use sequential or shuffle access order.	With sequential or shuffle access, each datapool row is referenced in the row access order just once. When the last row is retrieved, the datapool cursor either wraps or datapool access ends. With random access, rows can be referenced in the access order multiple times. Therefore, a given row can be retrieved multiple times. You can set row access order in any of these ways: ▪ When editing the `DATAPOOL_CONFIG` section of a VU test script in the Configure Datapool in Script dialog box, set **Access Order** to **Sequential** or **Shuffle**. ▪ When editing a VU test script in Robot, add `DP_SEQUENTIAL` or `DP_SHUFFLE` to the list of flags in the *`flags`* argument of the `DATAPOOL_CONFIG` statement or the `datapool_open` function. ▪ When editing a GUI test script in Robot, set the *`sequence`* argument of the `SQADatapoolOpen` command to `SQA_DP_SEQUENTIAL` or `SQA_DP_SHUFFLE`.
Do not rewind the cursor during a test.	If you rewind the datapool cursor during a test (through the VU `datapool_rewind` function or the SQABasic `SQADatapoolRewind` command), previously accessed rows will be fetched again.

What to do （做什么）	**How to do it （怎么做）**
指定最少一列的唯一数据.	在Datapool Specification对话框中，指定最少一个数据池列应包含唯一的数据。唯一数据可以通过有符号整数类型，Read From File数据类型，和用户定义的数据类型提供。 有符号整数类型中，需要进行这些活动： ● 设置**Sequence**为Unique或Sequential。 ● 设置Repeat为1。 ● 如果**Sequence**=Unique，设置一个合适的最小和最大的范围。 ● 确认值的**Length**（长度），确保**NO. of records to generate**（产生的记录号码）适合产生的数字集。 Read From File数据类型，参阅289页*Generating Unique Values*的内容。 用户定义的数据类型，参阅267页*Generating Unique Value from User-Defines Data Types*的内容。

生成足够的数据池行.	生成至少和测试期间访问该数据池的次数一样多的唯一的数据池行。 例如，如果一个测试中，有50个虚拟测试者将访问一个数据池，并且每个虚拟测试者被设置为3次迭代，那么该数据池必须包含至少150行。 在Datapool Specification对话框的**NO. of records to generate**（产生的记录号码）区段中指定生成的行的数目。
禁用游标环绕.	如果数据池的游标在该数据池的最后一行被访问后发生环绕，那么先前检索的行会被再次检索。 用以下的方法可禁用游标环绕： ● 当你在Script对话框Configure Datapool中编辑一个VU测试脚本的**DATAPOOL_CONFIG**部分时，将**Wrap at end of file?**设置为**NO**。 ● 在Robot中编辑一个VU测试脚本时，给DATAPOOL_CONFIG语句或datapool_open函数的*flags*变量的列表中添加DP_NOWRAP。 ● 在Robot中编辑一个GUI测试脚本时，设置SQADatapoolOpen命令的wrap语句为False。
使用senquential或shuffle访问顺序.	利用senquential或shuffle访问顺序，每一个数据池行刚好每次参照该行的访问顺序。当最后一行被检索时，该数据池的游标会发生环绕或者数据池的访问结束。 利用random（随机）访问，行可以在访问顺序中被多次参照。因此，一个给定的行可以被多次检索。 你可以用以下的任一方法设置行访问顺序： ● 当你在Script对话框Configure Datapool中编辑一个VU测试脚本的**DATAPOOL_CONFIG**部分时，设置Access Order为Senquential或Shuffle。 ● 在Robot中编辑一个VU测试脚本时，给DATAPOOL_CONFIG语句或datapool_open函数的*flags*变量的列表中添加DP_SEQUENTIAL或DP_SHUFFLE。 ● 在Robot中编辑一个GUI测试脚本时，设置SQADatapoolOpen命令的sequence语句为SQA_DP_SEQUENTIAL或SQA_DP_SHUFFLE。
不要在测试期间重绕游标.	如果你在测试期间重绕数据池的游标（通过VU的datapool_rewind函数或SQABasic的SQADatapoolRewind命令），那么先前访问的行将会被再次检索。

注意事项：Rational Test可以保证一个数据池在通过Robot或者TestManager产生数据的时候包含有唯一的行。

在 Rational Test 外创建数据池（Creating a Datapool Outside Rational Test）

创建一个数据池文件并填充数据，你可以使用任意的文本编辑器，比如Windows Notepad，或者任意的应用程序，比如Microsoft Excel或Microsoft Access，可以保存为.csv格式。

例如，你可以创建一个数据池文件，并输入数据，可以一行一行的输入，一个值一个值的输入。或者，你可以从你的数据库中将数据输出到一个.csv文件中，当然这个文件是由像Excel这样的工具创建的。

在你创建并填充了一个数据池之后，你可以使用TestManager将该数据池输入到datastore。有关输入一个数据池的信息，参阅278页*Importing a Datapool*的内容。

数据池的结构（Datapool Structure）

一个数据池保存为一个文本文件，扩展名为.csv。该文件有下列的特征：

- 每一行包含一个记录。
- 每一个记录包含由一个字段分隔符分隔的字段值。任意字符可以被使用来作为字段分隔符。以下是一些常见的字段分隔符：
  - 逗号（，）。在U.S和U.K中是典型的默认符号。
  - 分号（；）。在大多数的其他国家中是典型的默认符号。
  - 冒号（：）。
  - 竖线（|）。
  - 斜线（/）。
- 数据池文件中的每一列都包含数据池字段值的一个列表。
- 字段值可以包含间隔。
- 如果该值在双引号中，那么这个单值可包含一个分隔符。例如，“Jones, Robert”在记录中是一个单值，而不是两个。

  引号只能使用在该值被保存在数据池文件中时。引号不是这个值的一部分。
- 一个单值可包含嵌入的字符串。例如，“Jones, Robert“Bob””在记录中是一个单值，而不是两个。
- 每一个记录以换行结束。
- 数据池的列名被保存在.spc文件中。（Robot和TestManager编辑.spc文件。不能直接编辑.spc

文件。）

- 保存在datastore中的数据池名称是与根（root）数据池名相同的（无.csv扩展）。数据池名称的最大长度为40个字符。

例子（Example Datapool）

下面是一个含有3行数据的数据池文件的例子。在这个例子中，字段值通过逗号分隔开：

```
John,Sullivan,238 Tuckerman St,Andover,MA,01810
Peter,Hahn,512 Lewiston Rd,Malden,MA,02148
Sally,Sutherland,8 Upper Woodland Highway,Revere,MA,02151
```

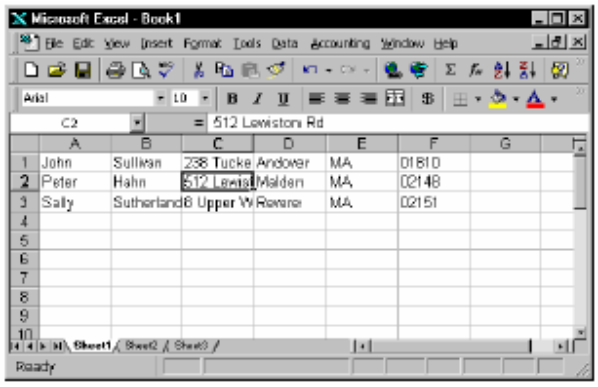
使用Microsoft Excel 创建数据池（Using Microsoft Excel to Create Datapool Data）

当你使用Microsoft Excel来填充一个数据池时，不要使用285页的Windows分隔符来分隔值。Excel在你将该数据池保存为.csv格式的时候，自动地插入分隔符。

使用Microsoft Excel创建并填充一个数据池：

- 点击**File > New**创建一个新的Excel工作簿。

下面的例子就是在Microsoft Excel中被看作的填充数据：



- 每一列表示一个字段。
- 每一行作为一个独立的数据池纪录包含了数据池字段的值。

在Excel 中保存数据池（Saving the Datapool in Excel）

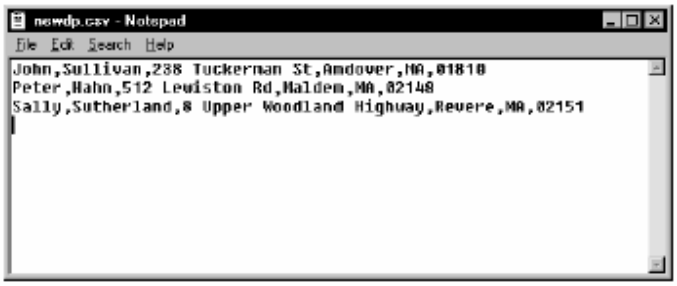
当你完成对数据池行值的添加时，可将该数据池保存为.csv格式。

使用Microsoft Excel保存一个数据池文件：

- 点击**File > Save As**。

注意事项：不要在datastore中指定该数据池的目录。当你迟些时候使用TestManager 的Import特性输入该数据池时，TestManager自动地复制该数据池到当前项目和datastore中的Datapool目录中。

如果使用Windows Notepad打开你刚刚创建并保存了的数据池，那么可以看到：



使数据池的列与测试脚本变量相匹配（Matching Datapool Columns with Test Script Variables）

当你创建一个.csv文件并将其作为一个数据池输入时，TestManager自动地为数据池的每一列指定列名（也就是数据池的字段名）。

数据池的列名必须与提供数据（包括一个case对）的测试脚本变量相匹配。但最为可能是，当你创建并输入一个数据池时，TestManager指定的列名不会与测试脚本变量关联的名称相匹配。因此，你需要在输入该数据池时编辑由TestManager自动指定的列名。你可以这样做，在Datapool Specification对话框中修改一列的**Name**值。

在数据池编辑时如何打开Datapool Specification对话框，请参阅*Editing Datapool Column Definitions* 276页的内容。

输入列的最大数目（Maximum Number of Imported Columns）

你可以输入一个含有32,768列的数据池。如果你在Datapool Specification对话框中打开一个输入的数据池，那么你可以查看并编辑所有的数据池的列定义，可达到它的列数目的上限。

一个数据池如果是由Datapool Specification对话框产生数据的话，那么只有一个150例的限制。

在 Rational Test 之外创建列值（Creating a Column of Values Outside Rational Test）

利用Rational Test创建的数据池包含了通过一个ASCII文本文件提供的列值。你可以使用这种特性，例如，如果你希望该数据池含有来自一个数据库的列值。

从一个外部文件填充一个数据池，需要两个基本步骤：

1 创建含有值的文件。

2 通过标准数据类型Read From File将该文件的值指定到一个数据池列。

步骤1.创建文件（Step 1. Create the File）

如果你希望使用一个文件作为数据池列的数据源，那么该文件必须是一个标准的ASCII文本文件。该文件必须包含一单个的列值，且每一个值通过回车结束。

你可以用自己喜欢的任意方法来创建这个文件—例如，你可以下面的这些方法：

- 用Microsoft Notepad输入列表的值。
- 从一个数据库输出一列值到一个文本文件中。

步骤2.分配该文件的值到数据池的列（Step 2. Assign the File's Values to the Datapool Column）

一旦该文件的值存在，你可以指定该值，如同你指定任意的值的集合到一个数据池列（通过一个数据类型）一样。在这个事例中，通过Read From File数据类型来分配值。

要这样做的话，可以从Datapool Specification对话框的**Type**列中，选择该数据类型Read From File，从而可以由外部的文本文件来为该数据池的列提供数值。

你可以使用Read From File数据类型向多个列（在相同的数据池中）分配数值。

生成唯一数值（Generating Unique Values）

你可以使用Read From File数据类型向一个你在Rational Test之外创建的数据池列产生唯一数值。要通过Read From File数据类型来创建唯一数值，具有数据类型访问的该文件中必须包含唯一数值。

此外，当你在Datapool Specification对话框中定义数据池时，为关联于ReadFrom File数据类型的数据池列进行以下的设置：

- 设置**Sequence**为Sequential。
- 设置**Repeat**为1。
- 确保**No. of records to generate**的数值不会超出你通过Read From File data数据类型关联的唯一数值的数量。

有关在Datapool Specification对话框中设置数据的信息，参阅271页*Defining Datapool Columns*的内容。

性能测试结果报表（Reporting Performance Testing Results） 12

这一章讨论性能测试的报表，以及评估该报表提供数据的一些建议方法。它包含了以下的主题：

- 关于报表
- 执行一个报表
- 定制报表
- 输出报表
- 改变报表默认
- 报表的类型

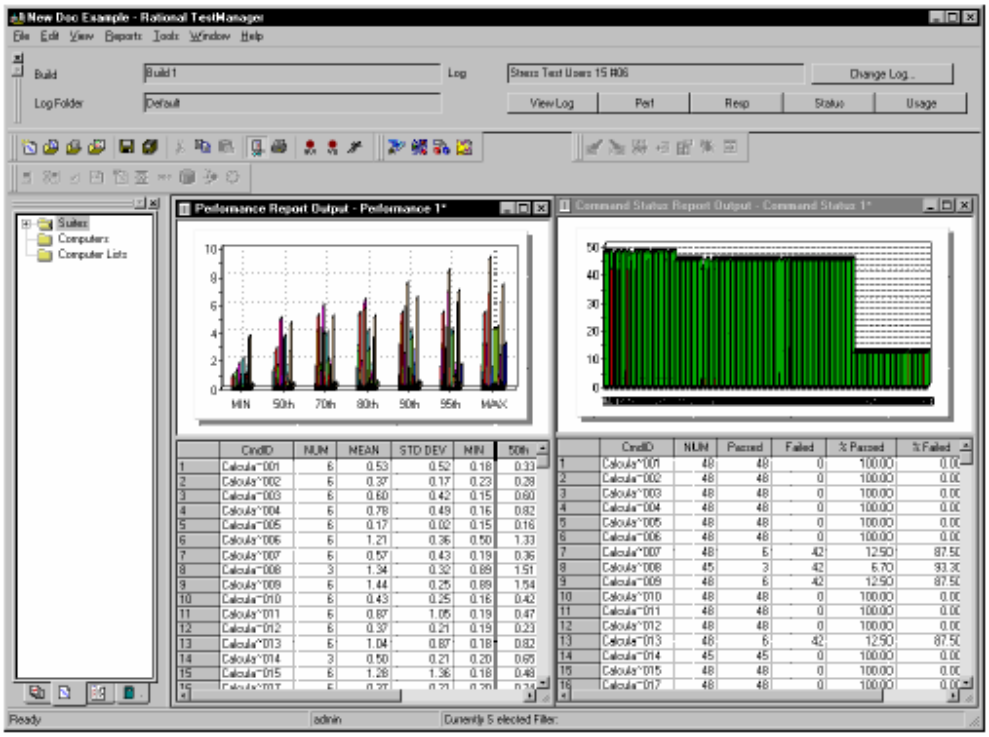
关于报表（About Reports）

TestManager提供若干种类型的报表，帮助你分析一个成功或失败的给定suite的执行，以及服务器在指定条件下的性能。例如，你可以确定一个虚拟测试者执行一个命令的时长是多少，以及不同的suite执行时变化的响应时间的长度。

你可以基于报表类型定义新的报表。定制报表可以帮助你进一步改进测试，通过你的测试计划和测试用例，显示你做判断时，需要的确切数据。

TestManager不能区分报表定义和处理的报表数据。大多数的活动可以在一个报表定义上执行，当然，你也可以在处理的报表数据上执行。

默认方式，如果一个测试成功地完成，并且该测试生成合适的数据，TestManager会自动地执行下图展现了一个简单的Command Status和Performance报表：



在你检查了报表数据之后，你可以保存或者删除该报表。如果你保存该报表，TestManager会给它一个默认的名称，该名称基于这个报表的类型以及该类型报表的数量—例如，Performance 1。TestManager将报表保存在该项目中的日志下。要再次查看该报表，你可以打开这个保存的报表。如果你删除了报表，通过执行相同日志下相同类型的报表，你可以重建它。

Report	Function	Information
Performance	Display the response times, and calculate the mean, standard deviation, and percentiles for each command in the suite run. The report groups responses by command ID and shows only responses that passed. In contrast, Response vs. Time reports show each command ID individually and show passed and failed responses.	*Performance Reports* on page 308
Compare Performance	Compare the response times measured by Performance reports. After you have generated several Performance reports, use the Compare Performance report to compare specific data.	*Compare Performance Reports* on page 312
Response vs. Time	Display individual response times and whether a response has passed or failed. This report is useful for looking at data points for individual responses as well as trends in the data. The report shows each command ID individually and the status of the response. In contrast, the Performance reports group responses by command ID and they show only passed responses. You can right-click on the report, select a computer that was in the run, and graph the resource monitoring statics for that computer. These are the same statistics that you display when you monitor resources during a suite run.	*Response vs. Time Reports* on page 317
Command Status	Obtain a quick summary of which and how many commands passed or failed. The report displays the status of all emulation commands and SQABasic timer commands.	*Command Status Reports* on page 320
Command Usage	View cumulative response time and summary statistics, as well as throughput information for emulation commands for all scripts and for the suite run as a whole.	*Command Usage Reports* on page 322

Report（报表）	**Function（功能）**	**Information（相关信息）**
Performance（性能）	显示响应时间，并计算平均值，标准偏差，以及在suite执行中每一个命令的百分比。 该报表通过命令ID分组响应，且只展现那些通过的响应。	308页Performance Reports
Compare Performance（比较性能）	比较通过性能报表衡量的响应时间。在你生成若干性能报表之后，使用Compare Performance报表比较指定的数据。	312页Compare Performance Reports

Response vs. Time （响应时间）	展现单个的响应时间，以及一个响应是通过还是失败。这种报表被用来查看单个响应的数据点，以及该数据中的趋势。 该报表单独地展现了每个命令的ID和响应的状态。比较起来，Performance（性能）报表是通过命令ID将响应分组，且只展现那些通过的响应。 你可以在报表中右键点击，选择执行的测试机，并描绘针对这台测试机的资源监视静态图。在一个suite执行期间，当你监视资源时，这里展现相同的统计。	317页的Response vs.Time Reports
Command Status （命令状态）	包含一个快速的摘要，有多少的命令通过或者失败。该报表展现了所有的仿真命令和SQABasic定时器命令的状态。	320页的Command Status Reports
Command Usage （命令利用率）	查看累计响应时间和统计结果摘要，以及针对所有测试脚本和一个完整suite执行的所有仿真命令的吞吐量信息。	322页的Command Usage Reports

注意事项：升级自Rational Suite PerformanceStudio或Rational LoadTest的用户可能希望访问早先生成的Analog和Trace报表的可用信息。现在，那些报表中的信息通过Test Log窗口是可用的。有关查看和使用日志，参阅127页*Evaluating Tests*的内容。

执行一个报表（Running a Report）

TestManager在一个suite执行结束时自动地执行默认的Performance 和Command Status报表（如果不是该suite中途失败或者日志数据没有产生）。这些报表提供相当数量的有关你的测试执行的信息，你可能希望执行其他的报表和/或改变任意给定报表中的信息。这一节讨论如何执行不同的报表。

从Report栏或TestManager菜单执行报表。

注意事项：你也可能希望查看日志文件—“未加工”的结果文件—在你执行报表之前。有关在Test Log窗口中查看日志的信息，参阅127页*Evaluating Tests*的内容。

从Report 栏执行一个报表（Running a Report from the Report Bar）

执行一个报表最快的方法是在Report栏上点击一个按钮。在Report栏上，TestManager列出由最后的suite创建的日志。如果你不指定为另一个日志，那么TestManage是用这个日志中的信息来执行一个报表。

从Report栏执行一个报表：

- 点击**View > Report Bar**，然后点击任意的一个报表按钮。

注意事项：利用你自己的报表来填充，你可以定制Report栏。更多的信息，参阅307页*Changing the Reports that Run from the Report Bar*的内容。

从菜单栏执行一个报表（Running a Report from the Menu Bar）

虽然TestManager使你从Report栏可快速地执行报表，但你可以利用此方法，将一个日志下的每一类型，只执行一个报表。然而，你可能希望从一系列的日志来执行许多的报表。例如，如果你定义了一些新的Performance报表，你可能希望针对相同的日志执行每个报表。

你可以从菜单栏执行这些报表。

从菜单栏执行一个报表：

- 点击**Reports > Run**，并选择要执行的报表类型。

定制报表（Customizing Reports）

TestManager可以使你针对特殊的测试需求来定制报表。

你可以通过以下的方法来定制一个报表：

- 过滤数据。

  例如，你可以过滤报表以使它只包含一个虚拟测试者组，只包含测试脚本，以及某几个命令ID。

- 改变一个报表的高级选项。

  例如，你可以修改一个Response vs. Time报表，以使那些过长的响应不包含在这个报表中。

- 改变一个图形的类型和外观。

  例如，你可以将一个图形显示为一个折线图或者一个柱状图。

在你定制了一个报表和保存它之后，你可以重复地使用它来快速地分析你的数据。

过滤报表数据（Filtering Report Data）

TestManager通过预先定义的设置和选项，提供一套默认的报表。

当然，你可以定制报表来过滤某些数据。

例如，292页的Performance报表包含许多的命令ID的信息，且图形复杂。要查看少量的命令ID，将图形拉近放大，像304页解释的那样。交替地，右键点击该报表，点击**Settings**，然后点击**Select**

Command IDs。

然而，代替过滤处理的报表，过滤报表的预先定义，以使结果报表中只含有你感兴趣的信息是更为容易的。你可以过滤一个报表使其只含有某几个虚拟测试者，只包含测试脚本，或者只含有某几个命令。

当你设置一个报表中的过滤时，你必须指定以下的信息，依赖于报表的类型：

- Build和log信息

  Build和log文件夹针对要寻找的日志文件，以及指定日志文件在报表中要过滤的数据。

- 虚拟测试者

  虚拟测试者和/或组（测试机或用户）关联于要过滤数据的指定日志。

- 脚本

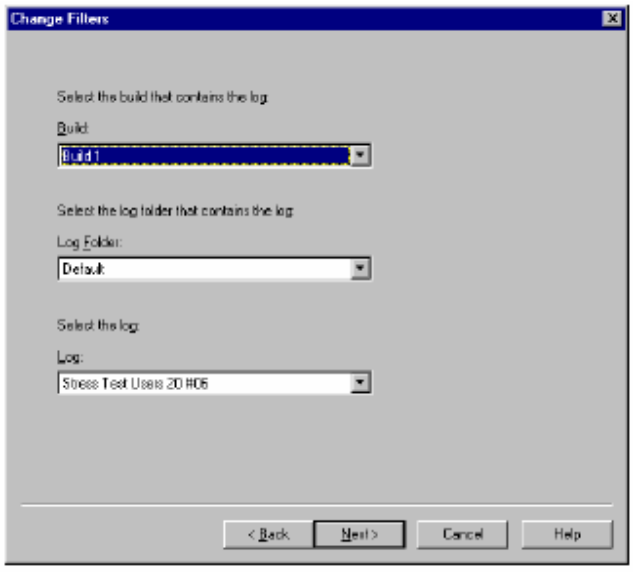
  测试脚本关联于要过滤数据的被选择虚拟测试者。

- 命令ID

  在被选择的测试脚本中指定要过滤数据的命令ID。

在Performance, Response vs. Time, Command Status, 和Command Usage报表中设置过滤：

- 打开或者创建该类型的一个新的报表，然后点击**Change Filters**。



注意事项：如果你过滤虚拟测试者，那么你通常要选择具有最大数量的虚拟测试者的日志。这样保证你的报表可以过滤所有的虚拟测试者。

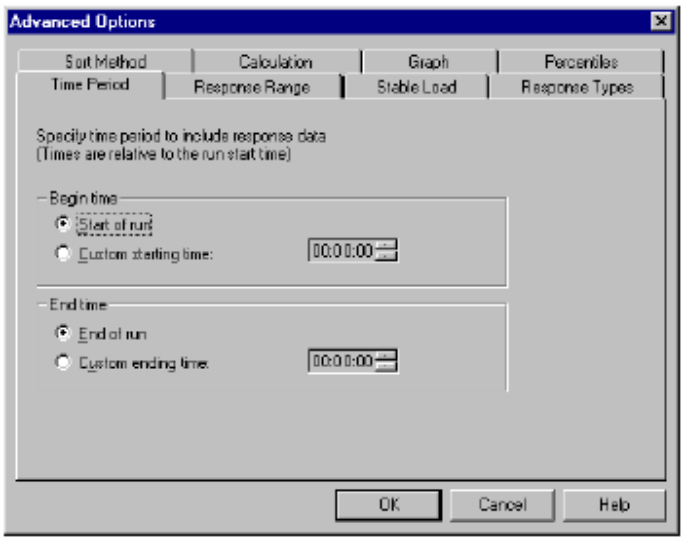
你也可以在执行报表之后，过滤该报表。更多的信息，参阅305页*Filtering Command IDs that Appear in a Graph*的内容。

设置高级选项（Setting Advanced Options）

所有的TestManager报表都具有高级选项，以确定报表如何被计算和展现。指定的高级选项对于每一个报表都是不同的。要调试一个报表，改变高级选项。

查看一个报表的高级选项：

- 打开或者创建一个新的报表，然后点击**Change Options**。



注意事项：有关高级选项的信息，参阅TestManager的帮助。

下表总结了每个高级选项，并列出使用这些选项的报表：

Option	Description	Reports
Graph	Display the report as a graph, a table, or both, change the type of graph displayed, change the labels for the graph axes, and add headers and footers.	Command Status, Performance, Response vs. Time, Compare Performance
Response Range	Include only responses that fall between a maximum and minimum time. The default includes all response times. You might want to set a maximum response time to eliminate outliers. If you change this option for one report, change the other reports, too, so that the reports reflect the same information. For more information, see *Eliminating Outliers* on page 299.	Command Status, Performance, Response vs. Time, Compare Performance
Response Types	Include only HTTP responses or responses with timers. The default includes all responses. The Command Status and Response vs. Time reports also let you filter responses that contain verification points.	Command Status, Performance, Response vs. Time

Option	Description	Reports
Sort Method	Sort command IDs numerically or in the order in which they were run. The default is to sort command IDs alphabetically.	Command Status, Performance, Response vs. Time
Stable Load	Specify a number of virtual testers that must be logged on before results are reported. The default is to report results when any number of virtual testers are logged on. You might want to change this option so that a certain number of virtual testers, or all virtual testers, must be logged on. For more information, see *Reporting on a Stable Load* on page 299. If you dynamically added virtual testers when you monitored the suite, you might want to change this option to correspond with the numbers of virtual testers you added. For more information, see *Reporting on a Dynamic Number of Virtual Testers* on page 300. If you change this option for one report, change the other reports, too, so that the reports reflect the same information.	Command Status, Performance, Response vs. Time
Time Period	Report on a specific portion of the suite run. The default is to report on the entire run.	Command Status, Performance, Response vs. Time
Calculation	Change how response times are calculated. The default measures the time from the end of the last send command until the last byte of the response is received. If you change this option for one report, change the other reports, too, so that the reports reflect the same information.	Performance, Response vs. Time
Response Status	Include only passed responses, or only failed responses. The default is to include all responses.	Response vs. Time
Summary	Summarize data by virtual tester, test script, command ID (Command Status), or run (Command Usage). The default for the Command Status report is detailed by command ID; the default for the Command Usage report is by run.	Command Status, Command Usage
Percentiles	Change how the response times are grouped. Generally, the defaults of 50, 70, 80, 90, and 95 are adequate.	Performance

Option **（选项）**	**Description** **（描述）**	**Report** **（报表）**
Graph **（图形）**	将报表展现为一个图，一个表，或者两者皆有，改变展现的图类型，改变图形的标号，以及添加标题和脚注。	Command Status, Performance, Response vs. Time, Compare Performance

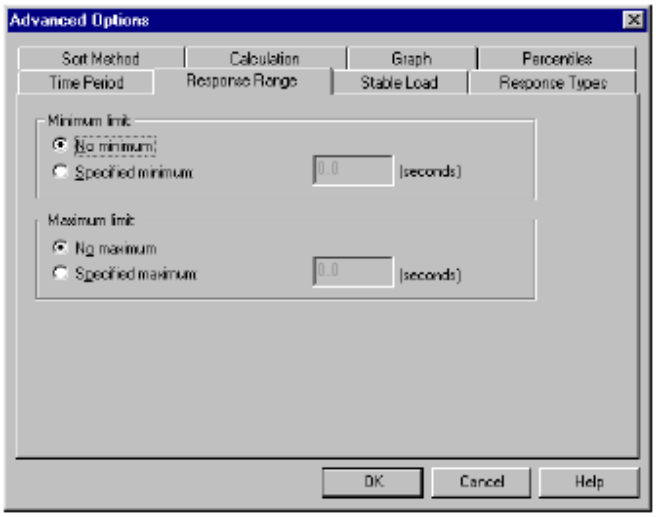
Response Range（响应范围）	只包含全部的在最小和最大之间的响应。默认包含全部的响应时间。 你可能希望设置最小响应时间来消除外露的部分。如果你针对一个报表改变改选项，那么其他的报表也要改变，以使报表反映出相同的信息。更多的信息，参阅299页Eliminating Outliers的内容。	Command Status, Performance, Response vs. Time, Compare Performance
Response Types（响应类型）	只包含HTTP响应或定时器响应。默认包含这里所有的响应。Command Status和Response vs. Time报表也允许你过滤包含了查证点的响应。	Command Status, Performance, Response vs. Time
Sort Method（分类方法）	数字分类或者已他们执行的顺序分类命令ID。默认的命令ID的分类方法是按字母分类的。	Command Status, Performance, Response vs. Time
Stable Load（稳定负载）	指定一定数量的虚拟测试者在结果被记录前必须登陆在系统上。默认设置是在有任意数量的虚拟测试者登陆到系统之上时报表结果。你可能希望改变该选项，以使确定数量的虚拟测试者，或所有的虚拟测试者，必须登陆在系统上。更多的信息，参阅299Reporting on a Stable Load页的内容。 如果你在监控suite时，动态地添加虚拟测试者，那么你可能希望改变此选项以符合你添加的该数量的虚拟测试者。更多的信息，参阅300页Report on a Dynamic Number of Virtual Testers的内容。 如果你针对一个报表改变该选项，那么也要改变其他的报表，以使报表可以反映出相同的信息。	Command Status, Performance, Response vs. Time
Time Period（时间周期）	在suite执行的一个指定部分上的报表。默认设置为整个执行期的报表。	Command Status, Performance, Response vs. Time
Calculation（计算）	改变如何计算响应时间。默认时间尺度为从最后一个发送命令结束到接受响应的最后一个字节止。如果你针对一个报表改变此选项，那么也要改变其他的报表，以使报表可以反映出相同的信息。	Performance, Response vs. Time
Response Status（响应状态）	仅包含通过的响应，或者只有失败的响应。默认设置包含所有的响应。	Response vs. Time
Summary（摘要）	通过虚拟测试者，测试脚本，命令ID（命令状态），或者执行（命令应用）来总结数据。默认设置是：针对Command Status报表，通过命令ID描述细节；对于Command Usage报表，通过执行描述。	Performance
Percentiles（百分比）	改变如何对响应时间进行分组。 一般地，合适的默认设置为50，70，80，90和95。	Performance

消除外露层（Eliminating Outliers）

报表可能包含一些数值，称为“外露层”，它们完全在正常范围之外。例如，假设你执行一个有1000个虚拟测试者的性能报表，且大多数的响应时间的范围在2到7秒之间。该响应对于一个命令ID为30秒—远远超出了正常范围。这种只发生一次的响应，可以当作不具备代表性的数据。在一些情况中，你可能希望从报表中消除这样的数据点，因为他们可能不准确地偏离了累积数据。

消除“外露层”：

- 在Advanced Options对话框的**Response Range**标签中，指定一个最小的响应时间限制。



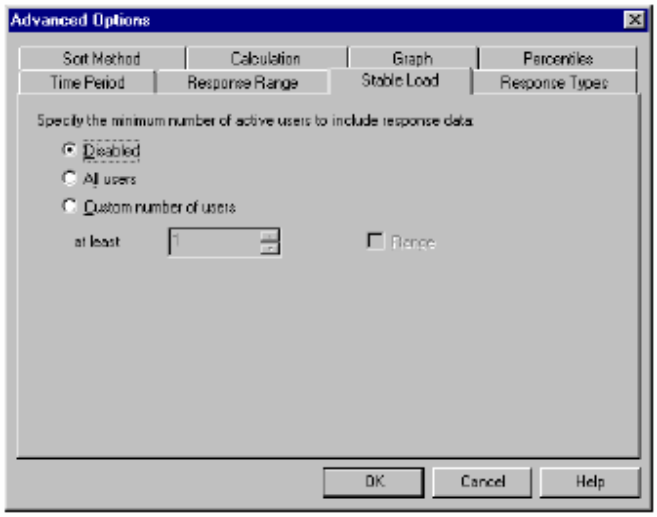
注意事项：认真地考虑是否需要从图形中移除数据点。虽然偏离中心的无关数据可以不具备代表性，但在其他一些情况中，外露层可能表示为其他性能的结果。

关于一个稳定负载的报表（Reporting on a Stable Load）

使一个报表只包含具有一个稳定的虚拟测试者负载的时间，需要有效地限制你的报表。例如，你很可能对这样的响应时间不感兴趣，比如只有很少的几个虚拟测试者登陆在系统上，或者大多数的虚拟测试者都退出系统了。

指定一个稳定的负载：

- 在Advanced Options对话框的**Stable Load**标签中，指定你认为可以算作一个稳定负载的虚拟测试者的数量。



关于动态数量虚拟测试者的报表（Reporting on a Dynamic Number of Virtual Testers）

如果你在执行一个suite时，动态地添加虚拟测试者，你可能希望知道这些添加如何影响你的结果。例如，如果你的suite是以50个虚拟测试者开始的，然后再动态地添加3组（50个一组）更多的用户，你的报表应该展现的范围是50—100，101—150，和151—200。

关于一个动态数量的虚拟测试者的报表：

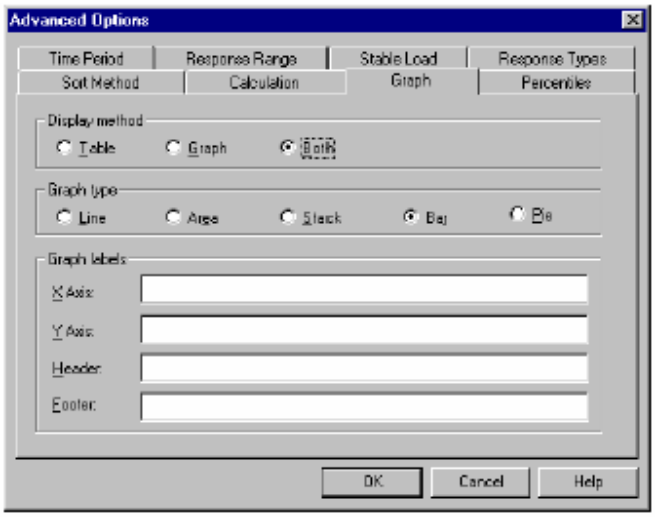
- 在Advanced Options对话框的**Stable Load**标签中，指定该数量的虚拟测试者。
- 在Advanced Options对话框的**Response Range**标签中，指定具有动态数量的虚拟测试者的范围。

关于特殊的命令ID的报表（Reporting on a Particular Command ID）

默认的Response vs. Time报表可以看起来比较混乱，因为它包含了每一个命令ID的有关信息。这个信息对于评估数据的趋势是有效的。然而，你可能希望有一个关于特殊的命令ID的报表或者一小组的命令ID，以及用一个线形直方图展现该报表，这样是比较容易阅读。

关于特殊的命令ID的报表，然后展现为一个线形直方图：

- 在Advanced Options对话框的**Graph**标签中，在过滤关于命令ID的报表之后，指定图形类型。



将测试机资源映射到响应时间之上（Mapping Computer Resource Usage onto Response Time）

监控测试机资源在性能测试中是必要的。如果有性能问题，那么你需要确定是由大数量的虚拟测试者引起的还是由于硬件本身的性能瓶颈。Response vs. Time报表使你在响应时间上覆盖测试机的资源统计。如果你的响应时间增加，你可以确定这是否是由于测试机的资源问题引起的。

注意事项：在你记录他们之前，TestManager必须设置来收集测试机资源的信息。当你执行一个suite时，Run Suite对话框出现，你必须选择**Monitor resources**检查框。

将测试机资源映射到响应时间之上：

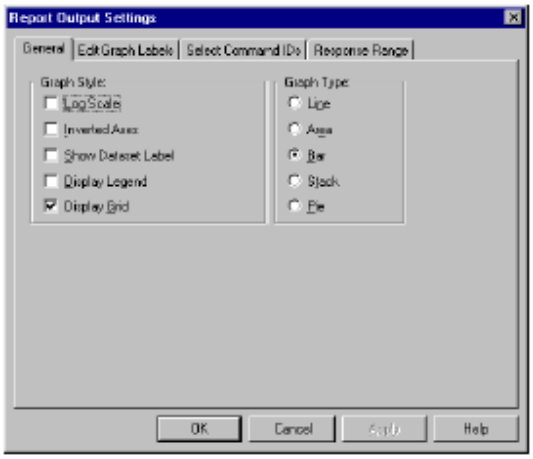
- 在Response vs. Time报表的图形上右键点击，然后点击**Show Resources**。

改变图形的外观或类型（Changing a Graph's Appearance or Type）

TestManager可以将Compare Performance，Performance，Response vs. Time，和Command Status报表展现为图和表风格的报表。Report Output Settings（报表的输出设置）对话框使你能够改变图形展现的类型，以及增强它的显示。

改变一个图形的类型或外观：

- 从一个打开的报表中，点击**View > Settings**。



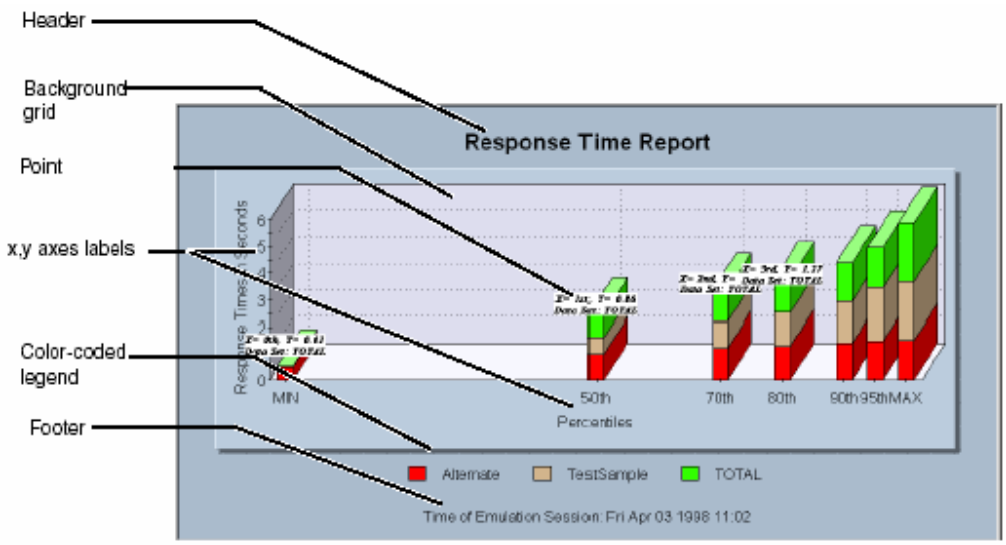
注意事项：对于改变一个图形的类型或外观的可用选项，其变化是围绕着你选择的报表类型的。从这个对话框中，你可以：

- 改变一个图形的外观。
- 改变一个图形的标号。
- 过滤信息，比如命令的ID号。

在一个性能测试报表中，你也可以改变图形中显示的响应范围。

改变一个图形的外观（Changing a Graph's Appearance）

TestManager使你可以控制一个图形的格式和外观。你可以展现或者清除那些不影响该图形的累积数据的选择点和数据集。下图展现一个簇状柱形图，有标题，背景格栅，和各种各样的其他选项：



你可以改变的图形选项包括：

- **Log Scale**—衡量任意的图形显示类型到它的对数等价。
- **Inverted Axes—**交换图形axes的相对位置。
- **Show Dataset Label—**添加数据集标签到图形。
- **Display Legend—**为所有的图形组件展现一个彩色代码图例符号（在Response vs. Time报表中不可用）。
- **Display Grid—**对于可视化的比较，展现一个格栅是很有效的（在百分图中不可用）。

展现和清除数据点信息（Displaying and Clearing Data Point Information）

在使用图形时，你可能希望展现图形中一个指定点的值。

要展现有关一个数据点的信息：

- 移动鼠标选择图形中期望的区域，点击CTRL-SHIFT-BUTTON1。

清除数据点信息：

- 右键点击该图形，然后点击**Clear Point Information**。

改变一个图形的类型（Changing a Graph's Type）

在使用图形时，你可以改变TestManager展现的图形类型。

改变一个图形的类型：

- 在你希望变更的该图形中，点击**View > Settings**，然后选择一个图形类型。

放大和旋转图形（Enlarging and Rotating a Graph）

通过点击的**SHIFT/CONTROL**组合键和鼠标，你可以进一步操作一个图形的外观。下表列出了一些你可以利用的方法：

Action	Mouse/Key sequence	Other required action
Enlarge a graph's size.	CTRL-BUTTON1 BUTTON2	Drag the mouse toward the bottom of the graph.
Change a graph's position.	SHIFT-BUTTON1 BUTTON2	Move the mouse to reposition the graph.
Zoom in on a graph's axes.	SHIFT-BUTTON1	Draw a box around the area to zoom, and then release BUTTON1.
Zoom in on a graph's data.	CTRL-BUTTON1	Draw a box around the area to zoom, and then release BUTTON1.
Rotate the view of a graph (stack and pie graphs only).	BUTTON1 BUTTON2	Move the mouse up and down to change the inclination angle. Move the mouse left and right to rotate the graph (stack only).
Reset a graph to its original size.	The lowercase letter "r"	None.

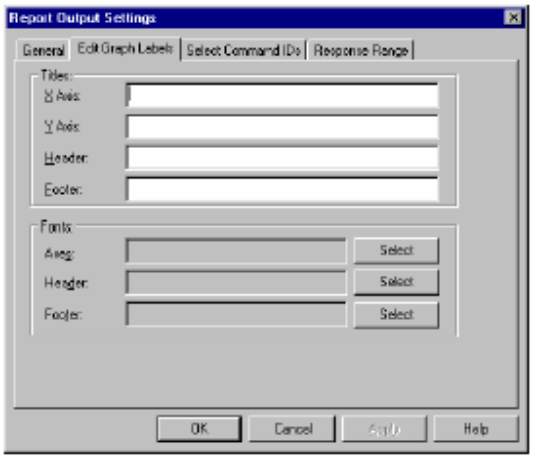
Action （活动）	Mouse/Key顺序	Other required action （其他的活动）
放大图形的尺寸。	CTRL-BUTTON1 BUTTON2	拖动鼠标接近该图形的底部。
改变图形的位置。	SHIFT-BUTTON1 BUTTON2	移动鼠标，重新定位该图形。
将图形的axes拉近。	SHIFT-BUTTON1	拉动一个围绕该区域的框来进行缩放操作，然后释放BUTTON1
将图形的数据拉近。	CTRL-BUTTON1	拉动一个围绕该区域的框来进行缩放操作，然后释放BUTTON1
旋转一个图形的视角（只能是簇状图和百分比图）。	BUTTON1 BUTTON2	向上和向下移动鼠标，改变倾斜角度。
重置图形，恢复初始大小。	小写字母“r”	无

改变图形的标号（Changing a Graph’s Labels）

在使用Command Status，Performance，或者Compare Performance图形时，你可以改变图形的标号，包括文本，字体，风格，和标号尺寸。

改变图形的标号：

- 在该图形中，点击**View > Settings**。



过滤图形中出现的命令ID（Filtering Command IDs that Appear in a Graph）

在你处理报表之前或之后，TestManager允许你过滤命令ID。

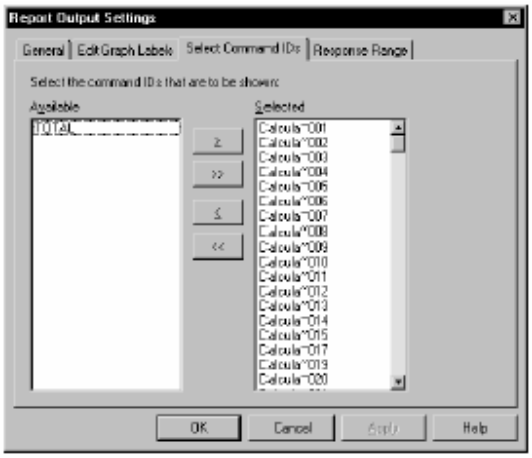
如果图形中的报表结果是复杂的，而且你希望检查某一部分的更多细节，那么在执行该报表之后过滤命令ID是很有效的。

To filter the command IDs in a graph:

. In the graph in which you want to filter IDs, click **View > Settings**.

过滤图形中的命令ID：

- 在该图形中，点击**View > Settings**。



输出报表（Exporting Reports）

Performance，Command Status，Compare Performance，Response vs. Time报表展现图形化的数据。你可以输出这个图形数据到一个.csv文件，以进一步处理。

要输出报表：

- 打开该报表，并点击**File > Export to File**。

改变报表的默认（Changing Report Defaults）

TestManager在suite执行结束时自动地生成Performance和Command Status报表。此外，你可以在Report栏中点击一个报表的名称，TestManager执行你点击的报表。

你可以指定报表。例如，除了Performance和Command Status报表之外，TestManager可以自动地展现一个Command Usage报表。或者TestManager可以生成一个Performance报表，这个报表基于你定义的，以替代默认Performance报表的一个报表。

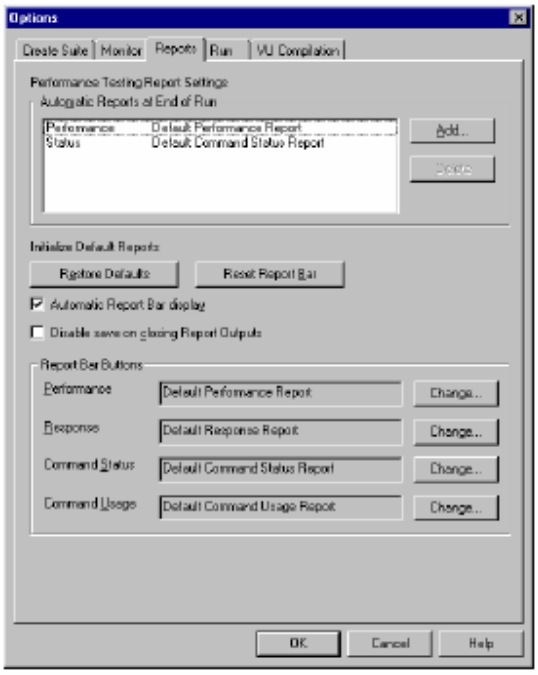
在点击Report栏的一个按钮时，你也可以改变TestManager执行的报表。例如，替代TestManager执行的默认Performance报表，你可以使它执行一个你定义的Performance报表。

改变自动执行的报表（Changing the Reports that Run Automatically）

TestManager在suite执行结束时，自动地展现Performance和Command Status报表。然而，你可以变更这些TestManager自动展现的报表。

改变该报表：

- 点击**Tools > Options**，然后点击**Reports**标签。



从Report栏中改变执行的报表（Changing the Reports that Run from the Report Bar）

Report栏允许你通过点击按钮执行报表。TestManager自动地执行默认报表，除非你指定。例如，你可能已经定义了一个新的报表以替代默认报表。

指定TestManager从Report栏执行的报表：

- 点击**Tools > Options**，然后点击**Reports**标签。

注意事项：重置Report栏，以使其生成默认报表，点击**Tools >Options**，点击**Reports**标签，然后点击**Reset Report Bar**按钮。

报表的类型（Types of Reports）

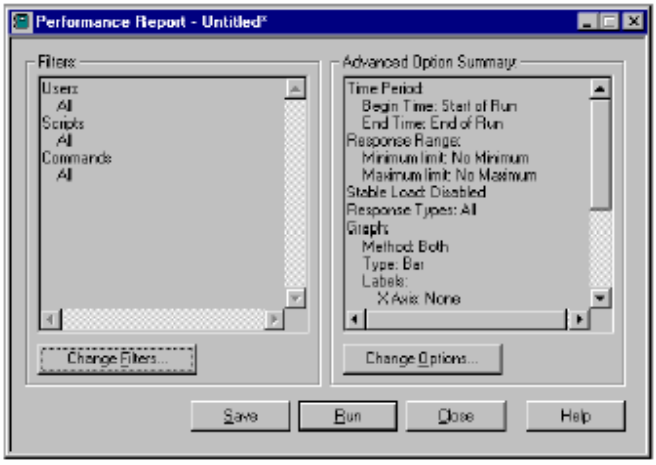
这一节讨论TestManager中可利用的五种类型的性能报表。

性能报表（Performance Reports）

你可以使用Performance报表来展现选择的命令在suite执行期间纪录的响应时间。Performance报表也提供响应时间平均值，标准偏差，和百分比。

要定义一个新的Performance报表：

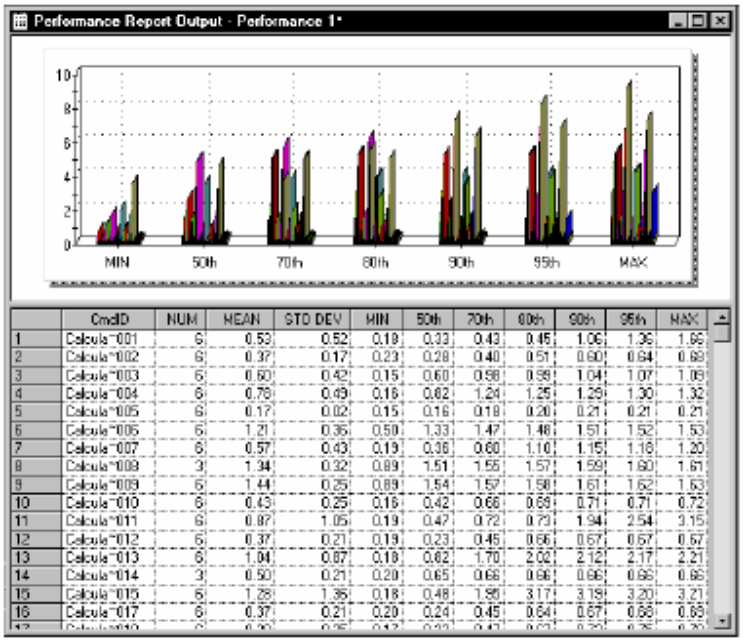
- 点击**Reports > New > Performance**。



TestManager中的Performance报表是纪录性能关联结果的基础。他们可以显示应用程序是否符合在测试计划和/或测试用例中定义的基本标准。例如，一个Performance报表告诉你，是否有95%的虚拟测试者接收到来自系统的响应，时间为8秒或者更少，或者有百分之多少的虚拟测试者在那时不能接受到来自于系统的响应。

Performance报表使用于Response vs. Time报表相同的输入数据，并且他们的分类和数据过滤也是相似的。然而，Performance报表以相同的命令ID将响应分组，同时，Response vs. Time报表单独地展示每一个命令ID。

下图展现一个报表的例子。这个图形为每个百分比种类展现了15栏，因为有15个命令被描绘。



	CmdID	NUM	MEAN	STD DEV	MIN	50th	70th	80th	90th	95th	MAX
1	Calcula~001	6	0.53	0.52	0.19	0.33	0.43	0.45	1.06	1.36	1.66
2	Calcula~002	6	0.37	0.17	0.23	0.28	0.40	0.51	0.60	0.64	0.68
3	Calcula~003	6	0.60	0.42	0.15	0.60	0.58	0.58	1.04	1.07	1.09
4	Calcula~004	6	0.78	0.49	0.16	0.62	1.24	1.25	1.29	1.30	1.32
5	Calcula~005	6	0.17	0.02	0.15	0.16	0.18	0.20	0.21	0.21	0.21
6	Calcula~006	6	1.21	0.35	0.50	1.33	1.47	1.48	1.51	1.52	1.53
7	Calcula~007	6	0.57	0.43	0.19	0.36	0.60	1.10	1.15	1.18	1.20
8	Calcula~008	3	1.34	0.32	0.89	1.51	1.55	1.57	1.59	1.60	1.61
9	Calcula~009	6	1.44	0.25	0.89	1.54	1.57	1.58	1.61	1.62	1.63
10	Calcula~010	6	0.43	0.25	0.16	0.42	0.66	0.69	0.71	0.71	0.72
11	Calcula~011	6	0.87	1.05	0.19	0.47	0.72	0.73	1.94	2.54	3.15
12	Calcula~012	6	0.37	0.21	0.19	0.23	0.45	0.66	0.67	0.67	0.67
13	Calcula~013	6	1.04	0.87	0.19	0.62	1.70	2.02	2.12	2.17	2.21
14	Calcula~014	3	0.50	0.21	0.20	0.65	0.66	0.66	0.66	0.66	0.66
15	Calcula~015	6	1.28	1.35	0.18	0.48	1.56	3.17	3.19	3.20	3.21
16	Calcula~017	6	0.37	0.21	0.20	0.24	0.45	0.64	0.67	0.68	0.69

- 该图形绘制了针对前置百分比的第二响应时间。
- MIN类展现每个命令ID的最小响应时间。
- 50th展现每个命令ID的第50百分比的响应时间。

  有一半的命令ID具有较短的响应时间，另外的一半具有较长的响应时间。
- MAX类展现每个命令ID的最大响应时间。

在性能报表中有什么（What's in Performance Reports?）

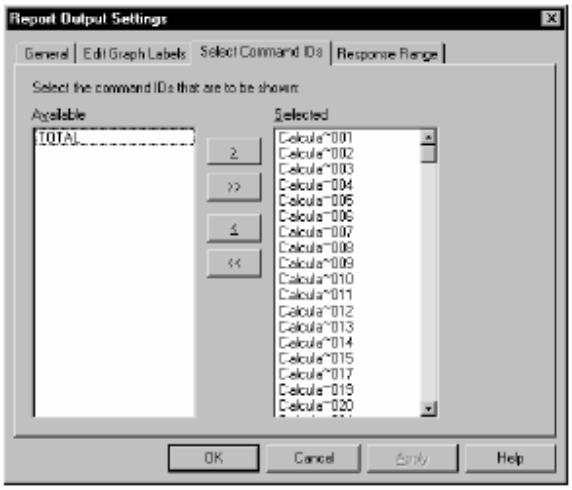
报表中包含以下的信息：

- **Cmd ID—**关联与响应时间的命令ID.
- **NUM—**针对每个命令ID的响应数量。
- **MEAN—**针对每个命令ID所得的全部响应时间的算术平均值。
- **STD DEV—**针对每个命令ID的所有响应时间的标准偏差。
- **MIN—**针对每个命令ID的所有响应的最小响应时间。
- **50th**, **70th**, **80th**, **90th**, **95th—**针对每个命令ID的所有响应的百分比响应时间。

  例如，如果添加Ne002的95th的百分比是0.53，那么说明95%的响应时间会少于0.53秒。
- **MAX—**针对每个命令ID的所有响应的最大响应时间。

例如，要在图中和报表的最后一行展现总的响应时间：

1 从一个Performance报表中，在图形上右键点击，并选择**Settings**。



2 点击**Select Commands IDs**标签，选择**TOTAL**。

有关Performance报表中的百分比（About Percentiles in Performance Reports）

一个Performance报表中的百分比代表最长量的时间，它定义了完成一个测试脚本的所有虚拟测试者总数的百分比。

百分数对于测试脚本中的每个命令ID和时间总数是给定的，所有的虚拟测试者去完成整个命令的列表。

例如，假定你有总共100个虚拟测试者，每次执行一个命令。在50th百分比栏中的时间说明有50%的虚拟测试者在那段时间里完成命令。一些虚拟测试者需要2秒的时间完成测试脚本中的一个命令，有一些是3秒，还有一些是5秒。出现在Performance报表上的50th百分比时间会是3秒，在50th百分比中的所有虚拟测试者完成命令的平均值少于或等于3秒。

Performance报表展现完成一个命令的最小和最大量时间。百分比的时间范围在最小和最大时间之内。

当你执行一个性能测试时，有些虚拟测试者执行时，完成命令的时间是反常的（比较长的时间），这种情况比较普通。例如，服务器可能在你执行性能测试时，正在下载一个应用程序，以至于在下载期间，虚拟测试者完成其任务的时间要长于其他的虚拟测试者。这被创建外露层—数据在一个比率的结束点是极端的，并且不能准确地代表所有虚拟测试者的趋势。你可以使用百分数用消除外露层的方法来评估测试结果，因此，会提供给你一个更为准确的真实响应时间的图形。

例如，假设你有一个服务器，它包含了一个内部的WEB站点，该站点为80名技术支持员工的使用提供信息。你需要做以下的工作：

- 确定获得的访问站点的时间一般不超过8秒。
- 创建一个suite，其中具有一个脚本，访问内部提供的WEB站点。
- 考虑预期的负载为80个用户，并确定为100个用户的测试，以消除外露层。
- 定义一个有100个虚拟测试者的用户组，执行suite，然后执行一个Performance报表。

在报表上的百分数展现为50，60，70，80，90，95。虽然你执行的suite有100个虚拟测试者，但因为你仅仅需要针对80个用户的测试结果，所以你可以不考虑90th和95th百分率，以消除外露层数据，获得针对80个用户的准确结果。由于消除外露层，可测试更多的虚拟测试者，使你可以确定该服务器是否可以处理多于预期负载的更多的负载。

Compare Performance报表（Compare Performance Reports）

Compare Performance报表比较通过Performance报表衡量的响应时间。在你生成若干个Performance报表之后，你可以使用一个Compare Performance报表来比较每个报表中专用字段的值。你也可以比较显示不同数量的虚拟测试者的报表，或者比较执行在不同系统配置下的报表。Compare Performance报表可以使你查看到一个特殊的应用程序针对各种各样的负载配置具有不同的基准信息。这可以帮助你识别，例如，在测试的应用程序中需要协议的改进。例如，你可以执行一个具有各种虚拟测试者负载的应用程序的测试，然后比较各种测试执行Performance报表，来查看应用程序是如何在一个不断增加的虚拟测试者负载下进行管理的。

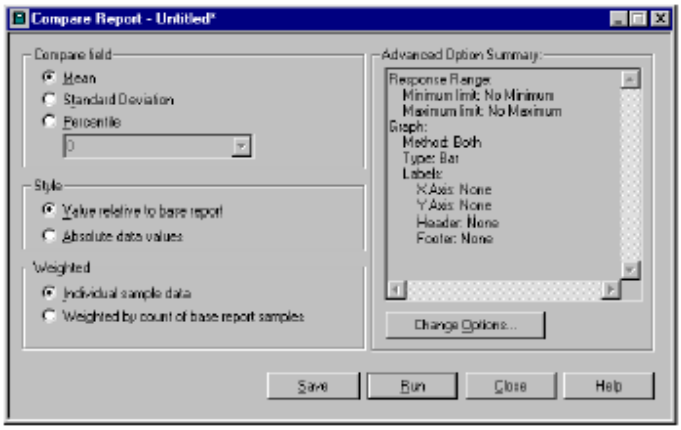
当你执行一个Compare Performance报表时，你指定一个基本的Performance报表和其他一些Performance报表，除了几本报表之外，其他的报表可多达6个。

定义一个Compare Performance 报表（Defining a Compare Performance Report）

定义一个Compare Performance报表与定义其他的报表是相似的。

要定义一个新的Compare Performance报表：

- 点击**Reports > New > Compare Performance**。



当你定义一个Compare Performance报表时，你必须定义以下的内容：

- 在Performance报表中选择的要比较的字段：
  - □ **Mean—**比较响应时间的平均值。
  - □ **Standard Deviation—**针对响应时间，比较标准偏差。
  - □ **Percentile—**比较基于你选择的百分数的响应时间。百分数必须在Performance报表中。例如，如果Performance报表计算50，70，80，90，和95的百分数，那么Compare Performance报表必须使用其中的一种。
- 相对于基本Performance报表的比较风格：
  - □ **Value relative to base report**（相对于基本报表的值）**—**比较相对于基本Performance报表的响应时间。利用该选项，报表（基本Performance报表）中的第一栏一般为1，并且其他栏是关联与它的。例如，如果基本报表列出的响应时间为2.5，其他报表的列出的响应时间为5，那么在Compare Performance报表中列出的值为1和2。
  - □ **Absolute data values**（绝对数值）**—**指明的响应时间出现在报表中。例如，如果基本报表列出的响应时间为2.5，其他报表的响应时间为5，那么在报表中列出的值为2.5和5。
- 发生最为频繁的响应时间的重量：
  - □ **Individual sample data**（单个的实例数据）**—**响应时间不论重量的。一个发生了10次的命令ID和一个发生了100次的命令ID对响应时间统计的影响是相同的。
  - □ **Weighted by count of base report samples**（根据基本报表的实例计数负重）**—**响应时间是加重量的，以映射与他们相符的命令ID的发生频率。发生越是频繁的命令ID越是会影响响应时间的统计，当然，发生频率越低的命令ID对统计的影响也会比较低。

注意事项：有关高级选项的信息，参阅TestManager的帮助。

Compare Performance报表中的内容（What's in Compare Performance

Reports?）

一个Compare Performance报表可以用多种方法比较。它可以完全地的比较报表，或者可以相对于一个基本报表来比较报表。此外，响应时间是负重的，以使频繁发生的命令ID具有更多的影响，或者它们不用负重，以使每一个命令ID具有相同的影响。

Compare Performance报表有4个版本：

- 绝对的
- 负重绝对的
- 相对的
- 负重相对的

绝对的Compare Performance 报表（Absolute Compare Performance Reports）

绝对报表展现响应时间的实际值，而且报表的最后一行给出了响应时间的算术和。

定义一个绝对报表：

- 依照312页*Defining a Compare Performance Report*的步骤，选择**Absolute data values**选项。

下图展示了一个绝对Crmompare Perfoance报表的最后几行的内容：

	CmdID	Build 1	Build 1	Build 1	Build 1
		sample schedule	sample schedule	sample schedule	sample schedule
		Users 5 #02	Users 10 #01	Users 15 #03	Users 20 #09
		Performance 1	Performance 1	Performance 1	Performance 1
149	READ R017	0.01	0.04	0.01	0.01
150	READ R018	0.01	0.02	0.01	0.01
151	READ R019	0.00	0.01	0.00	0.00
152	READ R020	0.00	0.01	0.00	0.01
153	READ R021	0.00	0.01	0.00	0.00
154	READ R022	0.00	0.00	0.00	0.00
155	SUM	2.73	4.79	5.11	12.05

负重的绝对Compare Performance报表（Weighted Absolute Compare Performance Reports）

负重的绝对报表通过他们发生的频率来给响应时间加重，这对于比较总的响应时间是有效的。

定义一个负重的绝对报表：

- 依照312页*Defining a Compare Performance Report*的步骤，选择**Absolute data values**选项和**Weighted by count of base report samples**选项。

重量的应用相当于在每个报表中针对那个命令ID的有效响应的数量。如果报表中的命令ID有着不同数量的响应，那么TestManager使用最小的非零数量作为其重量。

负重的绝对值是这个重量的乘积，且该绝对值是针对响应时间的。负重的绝对Compare Performance报表的最后一行，针对每个报表给出了负重响应次数的算术和。

下图展示了负重的绝对Compare Performance报表的最后几行：

	CmdID	Build 1	Build 1	Build 1	Build 1
		sample schedule	sample schedule	sample schedule	sample schedule
		Users 5 #02	Users 10 #01	Users 15 #03	Users 20 #09
		Performance 1	Performance 1	Performance 1	Performance 1
149	READ R017	0.03	0.12	0.03	0.03
150	READ R018	0.03	0.06	0.03	0.03
151	READ R019	0.00	0.03	0.00	0.00
152	READ R020	0.00	0.03	0.00	0.03
153	READ R021	0.00	0.03	0.00	0.00
154	READ R022	0.00	0.00	0.00	0.00
155	WEIGHTED SUM	9.11	14.61	17.45	41.53

相对Compare Performance报表（Relative Compare Performance Reports）

相对报表将1.00作为基响应时间，且其他的响应时间相对于基响应时间。

定义一个相对报表：

- 依照312页*Defining a Compare Performance Report*中的步骤，并选择**Value relative to base report**项。

报表的最后一行给出了每个报表相对响应时间的几何平均值。要确定集合平均值，TestManager乘响应次数，然后获得的乘积根目录与响应的次数相等。

例如，如果响应为5次，那么TestManager将他们相乘，并获得第五根目录的乘积。

在数学中，几何平均值的含义是：

数值x1，x2，…，xk。

表达式为 $(x_1x_2...x_k)^{1/k}$ [（x1x2...xk）的（1 / k）次幂]。

下图展示了一个相对Compare Performance报表的最后几行：

	CmdID	Build 1	Build 1	Build 1	Build 1
		sample schedule	sample schedule	sample schedule	sample schedule
		Users 5 #02	Users 10 #01	Users 15 #03	Users 20 #09
		Performance 1	Performance 1	Performance 1	Performance 1
149	READ R017	1.00	4.00	1.00	1.00
150	READ R018	1.00	2.00	1.00	1.00
151	READ R019	1.00	10.00	1.00	1.00
152	READ R020	1.00	10.00	1.00	10.00
153	READ R021	1.00	10.00	1.00	1.00
154	READ R022	1.00	1.00	1.00	1.00
155	GEO MEAN	1.00	1.51	1.07	1.44

负重的Compare Performance报表（Weighted Relative Compare Performance Reports）

该报表与相对报表是相同的，只是需要列出负重的几何平均值。

.

定义一个负重的相对报表：

- 依照312页*Defining a Compare Performance Report*的步骤，选择**Value relative to base report**和**Weighted by count of base report samples**选项。

负重的几何平均值不同于一般的几何平均值，它需要考虑不同的命令ID发生的频率。经常使用的命令ID比起不经常使用的，对负重的几何平均值，有着更大的影响—和一般的几何平均值相比，所有的命令ID具有相同的影响。

重量应用于为每个命令ID计算负重的几何平均值时，在每个已经比较的报表中，针对于那个命令相等数量的有效响应。如果针对一个命令ID的有效响应时间的数量在这些报表中是不相同的话，那么有一个最小的非零数被用来作为它的重量。

在数学中，负重的几何平均值的含义是：

数值x1，x2，…，xk，每个数值具有各自的频率（重量）f1，f2，…，fk，且有f1+f2+…+fk=N。

表达式为 $(x_1^{f_1}x_2^{f_2}\ldots x_k^{f_k})^{1/N}$ 。

下图展示了一个负重的Compare Performance报表的最后几行：

	CmdID	Build 1 sample schedule Users 5 #02 Performance 1	Build 1 sample schedule Users 10 #01 Performance 1	Build 1 sample schedule Users 15 #03 Performance 1	Build 1 sample schedule Users 20 #09 Performance 1
149	READ R017	1.00	4.00	1.00	1.00
150	READ R018	1.00	2.00	1.00	1.00
151	READ R019	1.00	10.00	1.00	1.00
152	READ R020	1.00	10.00	1.00	10.00
153	READ R021	1.00	10.00	1.00	1.00
154	READ R022	1.00	1.00	1.00	1.00
155	GEO MEAN	1.00	1.51	1.07	1.44
156	WGHT GMEA	1.00	1.30	1.08	1.47

N/A 与未定义响应（N/A and Undefined Responses）

偶然的情况，你可能会在一个Compare Performance报表中看到有`n/a`和`Undefn`这样的字符串。下表就为你描述了在何种情况下，`TestManager`会展现这些字符串：

If	Then the Compare Performance report
A command ID is in the base report but does not exist in the other reports.	Lists **n/a** for that command ID in the table and does not include information for that command ID in the report graph.
A command ID is in the report but does not occur in the base report.	Ignores that command ID.
You are producing a relative report, and some command IDs have a response time of 0.	Lists the response time as **0** in the base report, and lists the other results corresponding to that command ID as **Undefn**.
All the response times for a report are listed as **n/a** or **Undefn**.	Lists the geometric mean or sum as **n/a**.

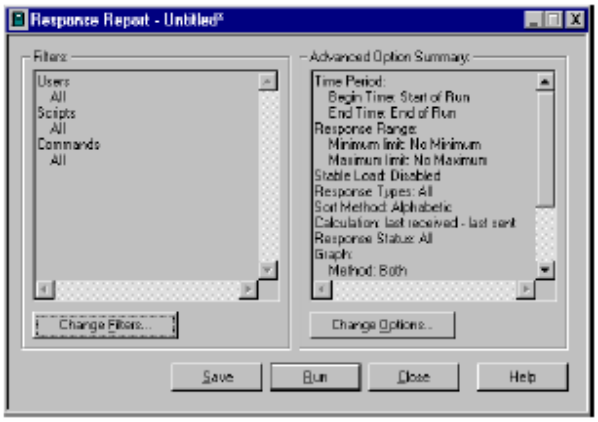
如果	**然后Compare Performance**
一个命令ID是在基报表中，而不存在于其他的报表中。	在表中为这个命令ID列出n/a，但在报表图形中不包括此命令ID的信息。
一个在报表中的命令ID，但不在基报表中发生。	忽略这个命令ID。
你在产生一个相对报表，且有一些命令ID的响应时间为0。	在基报表中将该响应时间列出，且为0，列出其他的结果，且与作为**Undefn**的命令ID相符合。
在一个报表中，所有的响应时间以**n/a**或**Undefn**列出。	将几何平均值或“和”以n/a列出。

响应vs.时间报表（Response vs. Time Reports）

响应vs.时间报表展现了单个的响应时间。Response vs. Time报表使用了与Performance报告相同的输入数据，并分类和过滤同样的数据。然而，Response vs. Time报表是单个地展现每一个命令ID，Performance报表则是将相同命令ID的响应分组。

定义一个新的Response vs. Time报表：

- 点击**Reports > New > Response vs. Time**。



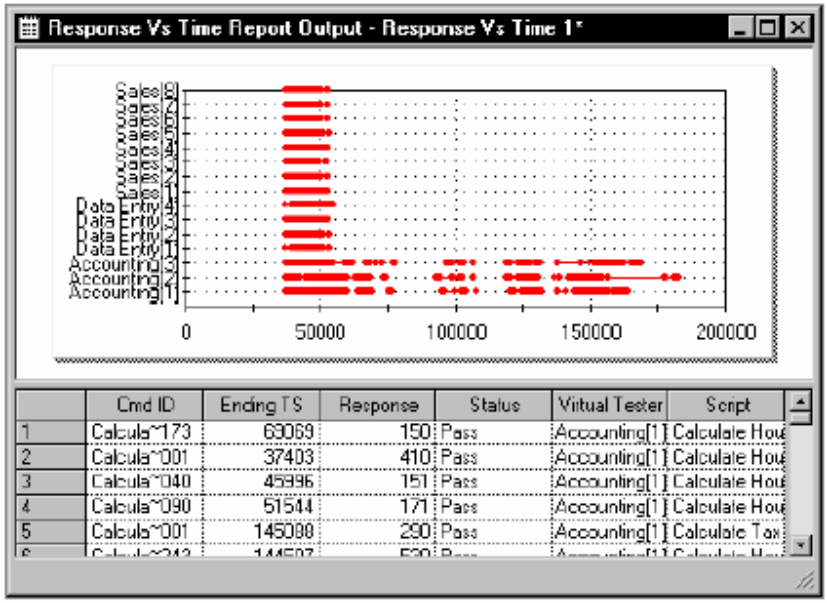
Response vs. Time报表主要执行以下的任务：

- 检查响应时间的趋势。Response vs. Time报表展现的响应时间与suite执行的经过时间相对应。响应时间应该是聚合在一个点的周围，而不是步进地获得更长或和更短的时间。如果趋势变化，检查你是否在结果中排除了登录和启动的时间。最糟糕的情况是，你可能需要去改变你的测试设计。
- 检查响应时间中任意峰值。如果响应时间除了1个或2个的峰值后，是相对平稳的，那么你可能希望去调查一下引起峰值的原因。
- 过滤数据以使其只包含一个命令ID，然后将这个命令ID做成一个直方图。
- 检查一台测试机在执行时的资源使用情况（可选）。

  要察看资源使用情况，在Response vs. Time报表上右键点击，并选择一个测试机。

The following figure shows a Response vs. Time report. This graph shows that the first virtual tester in the accounting group (Accounting 1) executed two commands.

下图展示了一个Response vs. Time报表。这个图形显示了accounting组中的第一个虚拟测试者执行了两个命令。



该图形绘出了每个虚拟测试者相对的响应时间，单位为毫秒（ms）。图形中包含了许多很像点的短线段。这些线段表明对于所有的虚拟测试者，响应时间是很短的。在X轴上的较长的线段，表示了较长的响应时间，因为X轴描绘的就是响应时间。

Response vs. Time 中的内容（What's in Response vs. Time Reports?）

一般地，Response vs. Time报表包含了两个部分，一个针对期望的响应时间，一个针对异常的响

应时间。每一部分中的响应根据命令ID来分类。在每个命令ID中，响应根据结束时间戳来分类。

Response vs. Time报表包含了以下的信息：

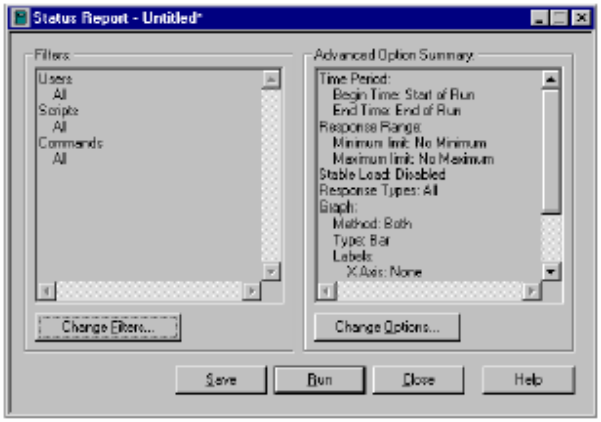
- **Cmd ID—**关联与响应的命令ID。
- **Ending TS—**响应的结束时间戳。这个时间戳与该响应的只读时间戳变量的值是相符合的。该时间戳是通过Time Period报表选项定义的间隔结束时间戳。
- **Response—**响应时间，单位毫秒（ms）。
- **Status—**为**P**或**F**，表示该响应是通过还是失败。
- **Virtual Tester—**对应于响应的虚拟测试者。
- **Script—**对应于响应的测试脚本名称。

命令状态报表（Command Status Reports）

Command Status报表展现实际响应与期望响应的符合情况。如果你收到的响应是一样或者预期的，那么TestManager认为它是通过的；否则，TestManager认为它是失败的。

要定义一个新的Command Status报表：

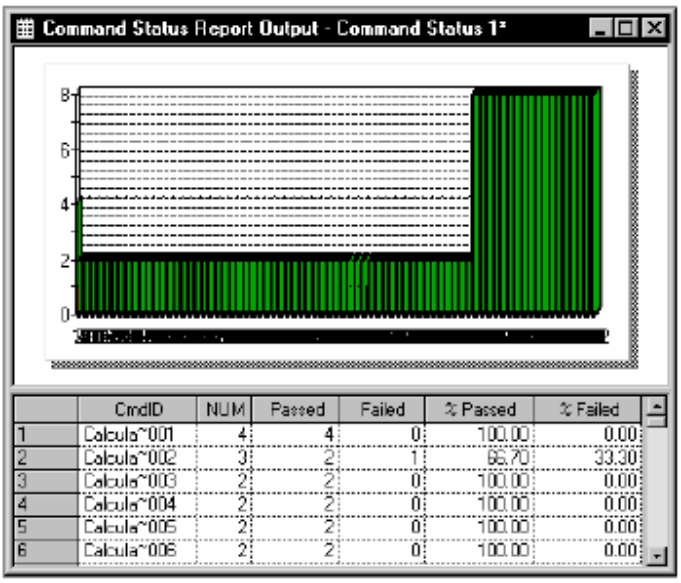
- 点击**Reports > New > Command Status**。



Command Status报表反映了一个suite执行的总的状况。他们类似Performance报表，但是他们更关注在suite中执行的命令的数量。

Command Status报表是调试测试过程的优秀工具，因为你可以容易地察看到命令的反复失败，以及测试脚本对应的地址。

下图展示了一个Command Status报表的例子。这个图形显示了命令1（命令ID为Add Ne01）执行了24次且没有失败，命令6（命令ID为Calcul001）执行了8次，且没有失败。



	CmdID	NUM	Passed	Failed	% Passed	% Failed
1	Calcula~001	4	4	0	100.00	0.00
2	Calcula~002	3	2	1	66.70	33.30
3	Calcula~003	2	2	0	100.00	0.00
4	Calcula~004	2	2	0	100.00	0.00
5	Calcula~005	2	2	0	100.00	0.00
6	Calcula~006	2	2	0	100.00	0.00

该图形绘出了和测试脚本执行次数相比的命令数量。

通过的命令标示为绿色，失败的标示为红色。

Command Status报表的内容（What's in Command Status Reports?）

- **Cmd ID—**关联与响应的命令ID。
- **NUM—**符合于每个命令ID的响应数量。这个数量是通过（**Passed**）与失败（**Failed**）两列的和。
- **Passed—**针对每个命令ID响应的通过数量（那些不是超时）。
- **Failed—**针对超时的每个命令ID响应的失败数量（预期的响应没有收到）。
- **% Passed—**针对那个命令ID通过的响应百分比。
- **% Failed—**针对那个命令ID失败的响应百分比。

报表的最后一行是每一列的总和。

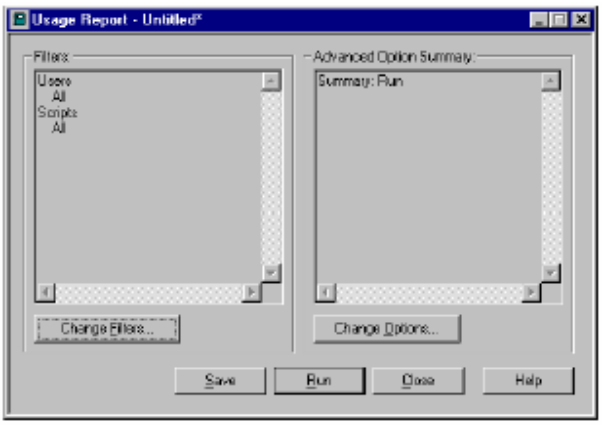
命令应用报表（Command Usage Reports）

Command Usage报表展现所有仿真命令和响应之上的数据。

该报表描述了suite执行期间的吞吐量和虚拟测试者的特性。

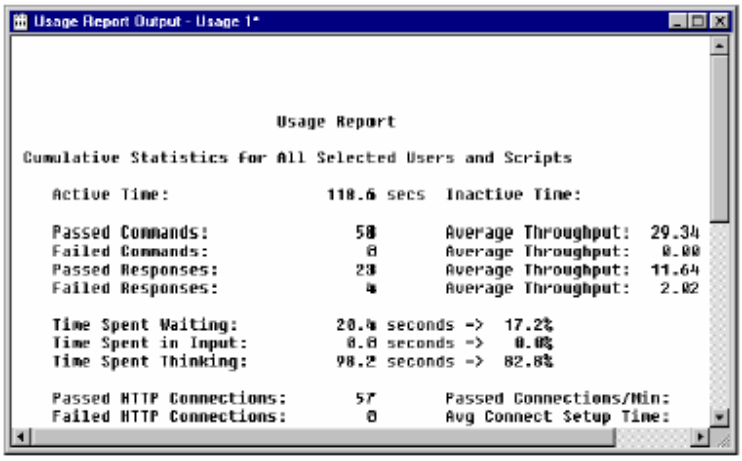
定义一个新的Command Usage报表：

- 点击**Reports > New > Command Usage**。



Command Usage报表中的摘要信息给出了一个在测试执行中活动分配的高层视图。通过虚拟测试者执行命令，思考，或者等待一个响应花去的累计时间，可以快速的告诉你在测试的应用程序中，哪里才是它的性能瓶颈。Command Usage报表也可以为协议提供摘要信息。

下图展示了一个Command Usage报表的例子：



Command Usage 报表的内容（What's in Command Usage Reports?）

Command Usage报表包含一个累计统计部分，和一个摘要统计部分。

累计统计（Cumulative Statistics）

- **Active Time**（有效时间）–所有的虚拟测试者的有效时间的总和。一个虚拟测试者的有效时间是虚拟测试者思考（包括了该虚拟测试者的第一个记录命令之后的延迟），执行命令，以及等待响应所花去的时间。
- **Inactive Time**（无效时间）–所有的虚拟测试者和测试脚本的无效时间的总和。一个虚拟测试者的无效时间是该虚拟测试者第一个仿真命令（包括需要启动和初始化执行的内务操作时间），和可能的内部脚本延迟（前一个测试脚本最后一个仿真命令与当前测试脚本开始的之

间的时间）。

- **Passed Commands**（通过的命令）–sqlexec,，sqlprepare，sql*_cursor，TUXEDO，http_request，sock_send，emulate，以及DCOM方法调用命令执行通过的总数量。
- **Failed Commands**（失败的命令）–sqlexec,，sqlprepare，sql*_cursor，TUXEDO，http_request，sock_send，emulate，以及DCOM方法调用命令失败的总数量。
- **Passed Responses**（通过的响应）–输入命令匹配与通过的接收命令（sqlnrecv，sqllongrecv， http_nrecv，http_recv，sock_nrecv，和sock_recv）的响应总数量。它和预期接收的命令总数是不相同的，因为一个响应可能与一个任意的接收命令相匹配。如果所有的接收命令被用来匹配一个响应，它有一个期望的状态，那么这个响应被认为是期望的响应。
- **Failed Responses**（失败的响应）–匹配与失败的接收仿真命令的响应总数。它和异常的接收命令总数是不相同的，因为一个响应可能与一个任意的接收命令相匹配。如果有任意的接收命令被用来匹配一个响应，它有一个异常的状态，那么这个响应被认为是异常的响应。
- **Average Throughput**（平均吞吐量）–提供了平均吞吐量的四种量度：通过的命令吞吐量，失败的命令吞吐量，通过的响应吞吐量，以及失败的响应吞吐量。这代表了一个平均的虚拟测试者的吞吐量。
- **Time Spent Waiting**（等待时间）–等待响应花去的总时间，既包含了秒的表示，也包括一个有效时间的百分比。等待时间是从输入的命令被提交到服务器，直到服务器接收完整的响应所经过的时间。例如，一个http_request等待一个链接被建立所计算的时间作为一个等待时间。
- **Time Executing Commands**（执行命令的时间）–用在sqlexec,，sqlprepare，sql*_cursor，TUXEDO， http_request，sock_send，emulate，以及DCOM方法调用命令执行的总时间。它的度量既包含了秒的表示，也包括一个有效时间的百分比。例如，SQL命令的执行时间被定义为，从SQL声明被提交到服务器，直到这些声明完成所经过的时间。TUXEDO命令的执行时间被定义为，从执行具体的ATMI元素，直到它成功或失败所经过的时间。
- **Time Spent in Input**(用在输入的时间)–把虚拟测试者的输入发送到服务器所花去的总时间。它的度量既包含了秒的表示，也包括一个有效时间的百分比。例如，通过http_request和sock_send命令将输入发送到服务器所记录的时间作为输入时间。
- **Time Spent Thinking**（思考时间）–思考花去的总时间，它的度量既包含了秒的表示，也包

括一个有效时间的百分比。针对一个给定命令的思考时间是从前面的仿真命令的结束，直到当前的仿真命令被提交到服务器所经过的时间。思考时间的定义，符合在执行期间仅当测试脚本中的环境变量Think_def为默认的LR（最后的接收）时使用，假定思考时间开始于前一个响应的最后的接受数据的时间戳。

如果有任意的SQL仿真命令被执行，那么Command Usage报表包括：

- **Rows Received**（接受的行）–所有的被记录的sqlnrecv命令的接收行的数量。
- **Received Rows/Sec**（接受行/秒）–每秒接受行的平均数目。接受行的数量除以有效时间。
- **Average Rows/Response**（平均行/响应）–通过和失败的响应中的平均行数。接受的行数除以通过和失败的响应数。
- **Average Think Time**（平均响应时间）–只针对sqlexec和sqlprepare的平均响应时间，单位秒（S）。
- **SQL Execution Commands**（SQL执行的命令）–被记录的sqlexec命令的数量。
- **Preparation Commands**（预备命令）–被记录的sqlprepare命令的数量。

. **Rows Processed** – Number of rows processed by all reported sqlexec commands.

- **Rows Processed**（处理的行）–通过所有的被记录的sqlexec命令处理的行数。
- **Processed Rows/Sec**（处理行/秒）–每秒处理的平均行数。处理行的数量除以有效时间。
- **Avg Rows/Execute Cmd**（平均行/执行命令）–每个sqlexec命令处理的行的平均数量。处理的行的数量除以被纪录的sqlexec命令数。
- **Avg Row Process Time** （平均行处理时间）–一个sqlexec命令处理一行的平均时间，单位毫秒（ms）。Sqlexec命令花去的时间除以处理的行数。
- **Avg Execution Time**（平均处理时间）–执行一个sqlexec或DCOM方法调用命令的平均时间，单位毫秒（ms）。Sqlexec命令花去的时间除以sqlexec命令的数量。
- **Avg Preparation Time**（平均预备时间）–执行一个sqlprepare命令的平均时间。Sqlprepare命令花去的时间除以sqlprepare命令的数量。

如果有任意的HTTP仿真命令被执行，那么Command Usage报表包括：

- **Passed HTTP Connections**（通过的HTTP的连接）–所有被纪录的http_request命令，所建立起来的成功的HTTP连接的数量。
- **Failed HTTP Connections**（失败的HTTP的连接）–所有被纪录的http_request命令，建立HTTP连接时失败的数量。
- **HTTP Sent Kbytes**（HTTP发送的千字节数）–通过被纪录的http_request命令发送的千字节数据。

- **HTTP Received Kbytes**（HTTP接受的千字节数）–通过被纪录的http_nrecv和http_recv命令接受的千字节数据。
- **Sent Kbytes/Connection**（发送千字节/连接）–通过http_request命令的每连接所发送的千字节数。发送的千字节数据除以成功建立的HTTP连接数量。
- **Passed Connections/Min**（发送千字节/连接）–每分钟成功建立的HTTP连接数量。成功的HTTP连接数除以有效时间。
- **Avg Connect Setup Time**（平均的连接建立时间）–平均时间，单位毫秒（ms），需要建立一个成功的HTTP连接。针对所有的被纪录http_request命令的连接总时间除以成功的连接数量。
- **HTTP Sent Kbytes/Sec** （HTTP发送的千字节数/秒）–每秒发送的千字节数。通过所有的被纪录http_request命令发送的千字节数据除以有效时间。
- **HTTP Recv Kbytes/Sec**（HTTP接受的千字节数/秒）–每秒接受的千字节数。通过所有的被纪录http_nrecv和http_recv命令接受的千字节数据除以有效时间。
- **Recv Kbytes/Connection** （接受的千字节数/连接）–通过http_nrecv和http_recv命令的每连接的接受千字节数据。接受的千字节数据除以成功建立的HTTP连接数。

如果有任意的socket仿真命令被执行，那么Command Usage报表包括：

- **Passed Socket Connections** （通过的Socket连接）–通过所有的sock_connect函数建立的成功的socket连接的数量。
- **Socket Sent Kbytes** （Socket发送的千字节数）–通过sock_send命令发送的千字节数据。
- **Socket Received Kbytes**（Socket接受的千字节数）–通过被纪录的sock_nrecv和sock_recv命令接受的千字节数据。
- **Passed Connections/Min**（通过的连接/分钟）–每分钟成功建立的socket连接的数量。成功的socket连接的数量除以有效时间。
- **Socket Sent Kbytes/Sec**（Socket发送的千字节数/秒）–每秒发送的千字节数据。通过所有被纪录的sock_send命令发送的千字节数除以有效时间。
- **Socket Recv Kbytes/Sec**（Socket接受的千字节数/秒）–每秒接受的千字节数据。通过所有被纪录的sock_nrecv和sock_recv命令接受的千字节数除以有效时间。

如果有任意的TUXEDO仿真命令被执行，那么Command Usage报表包括：

- **Tuxedo Execution Commands**（Tuxedo的执行命令）–被纪录的TUXEDO命令数。
- **Avg Execution Time**（平均执行时间）–平均执行时间，单位毫秒（ms），一个TUXEDO命

令的。TUXEDO命令的执行时间除以TUXEDO命令数量。

如果有任意的start_time仿真命令被执行，那么Command Usage报表包括：

- **stop_time Commands**（停止时间的命令）–被纪录的stop_time命令数量。
- **stop_time Cmds/Min**（停止时间的命令/分钟）–每分钟的stop_time命令的数量。stop_time命令的数量除以有效时间。
- **start_time Commands**（开始时间的命令）–被纪录的start_time命令数量。
- **Avg Block Time**（平均阻塞时间）–针对被纪录的stop_time命令的平均响应时间，单位秒（s）。针对所有stop_time命令的响应时间的总和除以stop_time命令的数量。一个stop_time命令的响应时间是它和关联于它的start_time命令之间经过的时间。

如果有任意的emulate仿真命令被执行，那么Command Usage报表包括：

- **Passed emulate Commands**（通过的emulate命令）–记录状态为通过的emulate命令数量。
- **Passed emulate Time Spent**（通过的emulate所花的时间）–时间总计，从通过的emulate命令开始到他们结束。
- **Failed emulate Commands**（失败的emulate命令）–记录状态为失败的emulate命令数量。
- **Failed emulate Time Spent**（失败的emulate所花的时间）–时间总计，从失败的emulate命令开始到他们结束。

如果有任意的testcase仿真命令被执行，那么Command Usage报表包括：

- **Passed testcase Commands**（通过的testcase命令）–记录状态为通过的testcase命令数量。
- **Failed testcase Commands** （失败的testcase命令）–记录状态为失败的testcase命令数量。

摘要统计（Summary Statistics）

- **Duration of Run**（执行的持续时间）–从执行开始到结束所经过的时间。执行的开始是所有的虚拟测试者和测试脚本之间的第一个仿真活动的时间，而不是你针对这个报表过滤的那些虚拟测试者和测试脚本。同样的，执行的结束是所有的虚拟测试者和测试脚本之间的最后一个仿真活动的时间。
- **Passed Commands**（通过的命令）, **Failed Commands**（失败的命令）, **Passed Responses**（通过的响应）, **Failed Responses**（失败的响应）–对应与323页*Cumulative Statistics*，内容完全相同。

- **Total Throughput**（总吞吐量）–提供四种总吞吐量的度量：通过的命令吞吐量，失败的命令吞吐量，通过的响应吞吐量，和失败的响应吞吐量。通过的命令的总吞吐量是通过的命令数量除以执行的持续时间，具有适当的秒到分的转化。因此，由所有可选择的虚拟测试者在应用负载时，它代表的通过命令的总吞吐量，和虚拟测试者的平均吞吐量相反。失败命令的总吞吐量，以及通过和失败响应的吞吐量具有相似的计算。

  这些吞吐量的度量，以及测试脚本吞吐量，依赖于虚拟测试者和测试脚本的选择集。例如，仅当有三个虚拟测试者从一个10虚拟测试者的执行中被选择，那么该吞吐量并不代表服务器的吞吐量在一个10虚拟测试者的工作负载下。作为一个指导方针，摘要吞吐量的度量在所有虚拟测试者和测试脚本被选择时是最有意义的。
- **Number of Users**（用户的数量）–在suite执行中虚拟测试者的数量。
- **Number of stop_time Cmds**（`stop_time`命令的数量）–在suite执行中`stop_time`命令的数量。
- **Number of Completed Scripts**（完成的脚本数量）–在执行结束之前，如果所有关联于一个测试脚本的活动都完成，那么该测试脚本被认为是完成了。
- **Number of Uncompleted Scripts**（未完成的脚本数量）–当一个执行异常终止时，未结束执行的测试脚本的数量。如果你终止执行或将suite设置为在某一数量的虚拟测试者和测试脚本结束后终止，那么该测试脚本被认为是未完成的。
- **Average Number of Scripts Completed per User**（每用户完成的平均脚本数量）–已完成的测试脚本数量除以虚拟测试者的数量。
- **Average Script Duration for Completed Scripts**（针对已完成脚本的平均持续时间）–一个完成的测试脚本经过的平均时间。所有虚拟测试者和测试脚本的累积有效时间除以已完成的测试脚本的数量。
- **Script Throughput for Completed Scripts**（针对已完成测试脚本的脚本吞吐量）–在执行期间通过服务器每小时完成的测试脚本数量。已完成的测试脚本数量除以执行的持续时间，具有秒到小时的转换。如果你过滤了虚拟测试者和测试脚本，该值会发生变化。

如果有任意的`start_time`仿真命令被执行，那么Command Usage报表包括：

- **Avg Number of stop_time Commands**（`stop_time`命令的平均数量）–`stop_time`命令数除以虚拟测试者数量。
- **Average start_time/stop_time Duration**（`start_time/stop_time`命令的平均持续时间）–针对被纪录`stop_time`命令的平均响应时间，单位秒（s）。针对所有`stop_time`命

令的响应时间的总和除以stop_time命令的数量。一个stop_time命令的响应时间是它和关联于它的start_time命令之间经过的时间。

- **Stop_time Command Throughput for all Users**（针对所有用户的stop_time命令吞吐量）–在suite执行期间每分钟执行的stop_time命令的执行数量。stop_time命令的数量除以执行的持续时间。

如果有任意的emulate仿真命令被执行，那么Command Usage报表包括：

- **Passed emulate Commands**（通过的emulate命令）–记录状态为通过的emulate命令数量。
- **Passed emulate Time Spent**（通过的emulate所花的时间）–时间总计，从通过的emulate命令开始到他们结束。
- **Failed emulate Commands**（失败的emulate命令）–记录状态为失败的emulate命令数量。
- **Failed emulate Time Spent**（失败的emulate所花的时间）–时间总计，从失败的emulate命令开始到他们结束。

如果有任意的testcase仿真命令被执行，那么Command Usage报表包括：

- **Passed testcase Commands**（通过的testcase命令）–记录状态为通过的testcase命令数量。
- **Failed testcase Commands** （失败的testcase命令）–记录状态为失败的testcase命令数量。

附录

本地和代理测试机的配置（Configuring Local and Agent Computers） A

如果你的Suite执行大量的虚拟测试者，那么为了使执行达到完全地成功，你必须设置确定的系统环境变量。该附录包含了以下的标题：

- 超过245个虚拟测试者的执行
- 超过1000个虚拟测试者的执行
- 在一台NT测试机上超过1000个虚拟测试者的执行
- 在一台UNIX代理机上超过24个虚拟测试者的执行
- 控制TCP端口数量
- 设置IP别名
- 为系统环境变量赋值

超过 245 个虚拟测试者的执行（Running More Than 245 Virtual Testers）

如果你的Suite执行虚拟测试者的总数超过245个，你必须在本地（Local）测试机上的NuTCRACKER操作环境中修改两项设置。要在一台NT代理（Agent）测试机上执行超过245个虚拟测试者，你必须在这台代理机上有相同的设置改变。

修改以下设置：

1 点击**Start （开始）> Settings（设置） > Control Panel（控制面板） > Nutcracker**。

2 点击**NuTC 4 Options**标签。

3 从种类（Category）列表中选择**Semaphore （信号）Settings**。

4 将**Max Number of Semaphores**修改为$N + S + 10$，N是你希望执行的虚拟测试者的数量，S是suite中被脚本使用的共享变量的数量。

5 针对**Max Number of Semaphores Per ID**，重复设置上值。

6 点击**OK**。

7 点击**Restart Later**。

8 重起NT。

超过 1000 个虚拟测试者的执行（Running More Than 1000 Virtual Testers）

如果你的Suite执行虚拟测试者的总数超过1000个，你必须创建一个环境变量，以此来设置本地（Local）测试机上共享内存的大小。要在NT代理（Agent）测试机上执行超过1000个虚拟测试者，你必须在这台代理机上有相同的设置改变。

要创建并设置这个环境变量：

1 点击**Start （开始）> Settings（设置） > Control Panel（控制面板） > System（系统）**。

2 选择**高级**标签，点击**Environment（环境变量）**。

3 创建一个名为RT_MASTER_SHM_MINSZ的环境变量，并设置它的值为700* N，N是你希望执行的虚拟测试者的数量。

在本地（Local）测试机上，N是针对整个执行的虚拟测试者的总数。

在代理（Agent）测试机上，N是执行在这台代理上的虚拟测试者的数量。

4 点击**Set**，然后点击**OK**。

5 重起NT。

在一台 NT 测试机上超过 1000 个虚拟测试者的执行（Running More Than 1000 Virtual Testers on One NT Computer）

如果是在一台NT测试机上，你的Suite有超过1000个虚拟测试者的执行，那么你必须在每一台执行

超过1000个虚拟测试者的NT测试机上创建并设置一个系统环境变量。

要创建并设置这个环境变量：

1 点击**Start（开始） > Settings（设置） > Control Panel（控制面板） > System（系统）**。

2 选择**高级**标签，点击**Environment（环境变量）**。

3 创建一个名为RT_MASTER_NTUSERLIMIT的环境变量，并将值设置为你希望执行的虚拟测试者的数量。

4 点击**Set**，然后点击**OK**。

5 为使新的设置对本测试机有效，重起（本地测试机上的）TestManager或者（代理测试机上的）test Agent。

在一台 UNIX 代理机上超过 24 个虚拟测试者的执行（Running More Than 24 Virtual Testers on a UNIX Agent）

如果在一台UNIX代理机上，你的Suite执行的虚拟测试者超过24个，那么你必须设置以下的系统环境变量：

System Environment Variable	Value
Total TestManager processes (NPROC, MAXUP)	The number of virtual testers on the Agent + 5.
Total open files (NFILE, NINODE)	*(6 * N) + (open_files * N) + (connections * N)* *N* is the number of virtual testers on the Agent. *open_files* is the number of files explicitly opened within test scripts. *connections* is the number of connections open concurrently.
Total system-wide shared memory (SHMALL/SHMMAX)	724 + 5609*N* + 16*S* + 13*G* + *group_names* bytes *N* is the number of virtual testers on the Agent. *S* is the total number of shared variables in all the test scripts in the suite. *G* is the total number of user groups in the suite. *group_names* is the total length of all user group names in the suite.
Semaphore set IDs (SEMMNI, SEMMAP)	1
Total semaphores (SEMMNS)	The number of virtual testers on the Agent.
Semaphores per set (SEMMSL)	The number of virtual testers on the Agent.

系统环境变量	**值**
TestManager进程总数（NPROC，MAXUP）	虚拟测试者数量，在代理机上+5。
打开文件的总数（NFILE，NINODE）	（6*N）+（open_files*N）+（connections*N） N是代理机上虚拟测试者的数量。 Open_files是在测试脚本中明确打开的文件的数量。 Connections是当前打开的连接数量。
全系统的共享内存总数（SHMALL/SHMMAX）	724+5609N+16S+13G+group_names字节数 N是代理机上虚拟测试者的数量。S是Suite中所有测试脚本共享变量的数量。G是Suite中用户组的总数。group_names是suite中所有用户组名称的长度总数。
信号设置IDs（SEMMNI，SEMMAP）	1
信号总数（SEMMNS）	代理机上的虚拟测试者的数量。
每套的信号数（SEMMSL）	代理机上的虚拟测试者的数量。

注意事项：这些值是其他系统进程或者应用程序需求之外的。当前系统值不应该被减少。例如，如果其他的系统进程要求SEMMNI=10，那么该值不能减少为1。

例如，对于一个Solaris代理机，执行2000-4000的虚拟测试者时，如下设置系统环境变量：

```
set semsys:seminfo_semmap=1024
set semsys:seminfo_semmni=4096
set semsys:seminfo_semmns=4096
set semsys:seminfo_semmnu=4096
set semsys:seminfo_semmsl=1024
set semsys:seminfo_semopm=50
set semsys:seminfo_semume=64
set semsys:seminfo_shmmni=1024
set semsys:seminfo_shmmax=100072000
set semsys:seminfo_shmseg=100
set semsys:seminfo_shmmin=1
```

TCP 端口数量的控制（Controlling TCP Port Numbers）

rtmstr_v和rtmstr_s网络服务控制本地测试机上的端口，代理机的通信软件连接到本地机。这些网络服务允许测试被执行在本地机与代理机上，由防火墙分开的不同网络中，通过对端口的控制，该端口对本地服务器绑定的进程进行监听。

在一个需要代理机的测试中，本地机与代理机之间具有多重socket连接。

本地机与代理机的连接，一般采用一个单一的熟知的端口，在这个端口上，代理机进行监听。该端口默认为8800。

有两个连接，其中一个的本地服务器，进程名为rtvsrv，另一个的本地服务器进程名为rtssrv。这两个服务器进程各自监听一个独立的端口。他们没有绑定到一个指定端口上，但是本地测试机上的操作系统会动态地选择一个端口。在测试执行的初始化期间，本地测试机会把这些端口值传送给代理机。

（注意事项：所有代理机连接的端口都是本地机上的这两个端口。）本地机上这两个动态选择的端口会引起防火墙的管理问题，因为这两个将被使用的端口不能事前被确定。

你可以控制这个问题，通过使用可选择的网络协议（传统的TCP/UDP网络协议被定义在一个/etc/services文件中，不能将其与一个NT服务相混淆。）在NT上，该服务文件在*Drive*\WINNT\system32\drivers\etc\services中。每行都有一个表目，列出了服务的名称，端口数量，以及相关协议（TCP或者UDP）。

具体地说，对端口的控制需要提供以下几点：

rtvsrv绑定到端口（按优先顺序）：

1 如果被定义了，TCP服务的值名为rtmstr_v。

2 如果没有定义，一个端口被系统动态地选择。

Rtssrv绑定到端口（按优先顺序）：

1 如果被定义了，TCP服务的值名为rtmstr_s。

2 如果没有定义，一个端口被系统动态地选择。

通过这两个服务定义的端口是独立的。就是说，它们不需要相邻，不用关联到那个熟知的测试代理机的8800端口。它们不需要设置成唯一。如果它们不是为本地机上相同的其他服务使用的话，我们建议使用端口8801和8802。

例如，如果你希望本地机上的端口是`8801`和`8802`，那么添加下面这两行到服务文件中：

```
rtmstr_s8801/tcp# TestStudio Master S server
rtmstr_v8802/tcp# TestStudio Master V server
```

此外，rtagent网络服务已经被添加到代理机监听的控制端口。如果代理机的8800端口已经在一台或者更多的代理测试机上被其他的应用程序所使用，那么一个交替的端口需要被指定为使用了rtagent服务。

Rtagent服务加进服务文件中的方法，与rtmstr_v、rtmstr_s网络服务的加进方法一样。不同之处在于，rtagent服务必须被定义在本地测试机上，且所有的代理机在测试执行时都用此服务，对所有的系统必须是同一的。在变更服务文件之后，代理机必须重起（重新引导）。

例如，如果你希望代理机在8888端口监听，那么在本地和代理机上添加下面的行到服务文件中：

```
rtagent8888/tcp# TestStudio Agent
```

设置 IP 别名（Setting Up IP Aliasing）

TestManager提供IP别名，可以使多个IP地址被分配到相同的物理系统。每一个虚拟测试者可以分配不同的IP地址真实地模仿你的虚拟测试者的团队。通过那些虚拟测试者产生的请求，利用定时特征与完整的合法性记录，接受来自于Web服务器的响应。

在任意一台特定的测试机上使用IP别名，系统管理员必须设置系统的IP地址。

对于Windows NT，可以这样做：

点击**Settings > Control Panel > Network > Protocols > TCP/IP Protocol > Properties > Advanced > IP Addresses > Add**按钮。

对于UNIX，与`ifconfig` (1)命令行的功能有关。参阅`ifconfig`手册，适合于该操作系统的

具体的细节内容。要设置大量的IP地址，使用Perl或者UNIX的shell脚本是较方便的。一个实例Korn shell脚本，为此目的命名为`ipalias_setup`，可以在UNIX代理机安装的“bin”目录中找到。（你必须具有root的权限，才能利用`ifconfig`来设置`IP`别名。）

由于你可能遇到诸如IP地址冲突或路由选择的考虑等问题，所以在分配IP地址到一台测试机时需要注意。我们推荐IP地址由一个限定的管理员来分配。

在IP地址被设置后，打开一个suite，点击**Suite > Edit Runtime**，并选择**Enable IP Aliasing**选择框。

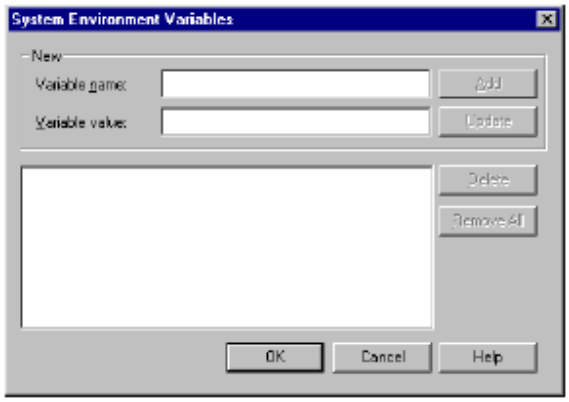
如果该suite中的IP别名被选择，那么在执行的开始时，每台测试机（本地或者代理）上的TestManager会询问所有的可用IP地址。在round-robin方式中，每个预先安排在那台测试机上执行的suite，都从那个表中分配一个IP地址。换句话说，如果一台测试机上的虚拟测试者多于IP地址，那么一个IP地址会被分配给到多个虚拟测试者。如果虚拟测试者少于IP地址，那么有一些IP地址不会被使用。任意特定的测试机上预先安排的虚拟测试者，不论其数量，IP地址的分布都接近最佳化，并且将你从IP地址与指定虚拟测试者相匹配的工作中释放出来。

为系统环境变量赋值（Assigning Values to System Environment Variables）

TestManager通过系统环境变量设置一台代理测试机作每个虚拟测试者。如果你使用虚拟测试者来测试一个数据库服务器，或者一个应用程序，你可以覆盖这些系统环境变量。

要覆盖系统环境变量的值：

- 点击**Suite > Edit Settings**，然后点击User Settings对话框的**Sys Environment Variables**栏中的按钮。



你可以在System Environment Variables对话框中改变一个值或者一个先前设置的系统环境变量。有关更多的信息，参阅TestManager帮助。

你可以在下表罗列的测试平台上设置系统变量：

Testing Platform	System Environment Variable Settings
Oracle on a UNIX Agent	Specify the directory that contains the client software in the variable `ORACLE_HOME`. Example: `ORACLE_HOME = /ora/app/oracle/product/8.0.5` If /var/opt/oracle does not contain tnsnames.ora, assign the pathname of the file to the variable `TNS_ADMIN`. Example: `TNS_ADMIN = /home/uname/oracletest`
Sybase on a UNIX Agent	Specify the directory that contains the client software in the variable `SYBASE`. Example: `SYBASE = /usr/local/sybasec` Specify the directory that contains the Sybase client libraries in the path of one of the following system environment variables: `PATH` (Windows) `LD_LIBRARY_PATH` (Solaris Agents) `SHLIB_PATH` (HP-UX Agents) `LIBPATH` (AIX Agents)

测试平台	**系统环境变量设置**
UNIX代理机 Oracle	在变量ORACLE_HOME中指定包含客户端软件的目录。 例如： ORACLE_HOME=/ora/app/oracle/product/8.0.5 如果/var/opt/oracle不包含tnsnames.ora，那么分配该文件的路径名到变量TNS_ADMIN。 例如：TNS_ADMIN=/home/uname/oracletest
UNIX代理机 Sybase	在变量SYBASE中指定包含客户端软件的目录。 例如：SYBASE=/usr/local/sybase 在以下的系统环境变量的一个路径中指定包含Sybase客户端库（libraries）的目录： PATH（Windows） LD_LIBRARY_PATH（Solaris Agents） SHLIB_PATH（HP-UX Agents） LIBPATH（AIX Agents）

Testing Platform	System Environment Variable Settings

Java on a UNIX Agent	Specify the directory that contains the Java libraries in the variable `LD_LIBRARY_PATH`.
	Example:
	`LD_LIBRARY_PATH=/usr/jre118/lib/linux/native_threads`
	When the Agent computer is also using third-party software (such as IBMWebSphere) you must specify the directory location of that software's libraries in the system environment variable `LD_LIBRARY_PATH` in addition to the Java libraries.
	In addition, for Java Developers Kit version1.1, you must also set the following variable:
	`JAVA_COMPILER=NONE`

Local or Agents running TUXEDO test scripts	Specify the directory that contains the client software in the variable `TUXDIR`.
	Set the `NLSPATH` environment variable to the path of the directory that contains the `TUXEDO` message file.
	Set the value of `$TUXDIR/lib` to one of the following system environment variables:
	`LD_LIBRARY_PATH` (Solaris Agents)
	`SHLIB_PATH` (HP-UX Agents)
	`LIBPATH` (AIX Agents)
	For Windows NT Local computers, these must be defined only for TUXEDO client-only installations. The TUXEDO full runtime installation process sets them automatically. For more information, see the TUXEDO installation instructions.
	Set one of the following:
	The Workstation Listener's address to `WSNADDR`.
	Example:
	`WSNADDR=//sparky:36001`
	`WSNADDR=00028CA1C0A8F0D6`
	The Workstation Listener's host name and port to `WSLHOST` and `WSLPORT`. These variables override `WSNADDR`, if set.
	Example:
	`WSLHOST=sparky`
	`WSLPORT=36001`

Testing Platform	System Environment Variable Settings
Local or Agents running TUXEDO test scripts that use FML typed buffer field names	Set a list of FML field table file names to FIELDTBLS. This variable is used by Agents running test scripts that contain FML typed buffer field name references. If this variable is not set, functions that use FML typed buffer field names that are not included in this list will fail, causing dependent commands to fail. Example: FIELDTBLS=ct.fldtbl,inv.fldtbl Set the absolute pathname of the directory containing the FML field table file to FLDTBLDIR. This variable is used by Agents running test scripts that contain FML typed buffer field name references. If this variable is not set, functions that use FML typed buffer field names that are not included in this list (for example, tux_setbuf_int()) will fail, causing dependent commands to fail. Example: FLDTBLDIR=/u1/tuxapp/dat
Local or Agents running TUXEDO test scripts that use FML32 typed buffer field names	Set a list of FML32 field table file names to FIELDTBLS32. This variable is used by Agents that run test scripts that contain FML32 typed buffer field name references. If this variable is not set, functions that use FML32 typed buffer field names that are not included in this list will fail, causing dependent commands to fail. Example: FIELDTBLS32=ct32.fldtbl,inv32.fldtbl Set the absolute pathname of the directory containing the FML32 field table files to FLDTBLDIR32. This variable is used by Agents running test scripts that contain FML32 typed buffer field name references. If this variable is not set, functions that use FML32 typed buffer field names that are not included in this list (such as tux_setbuf_int()) will fail, causing dependent commands to fail. Example: FLDTBLDIR32=/u1/tuxapp/dat
Local or Agents running TUXEDO test scripts that use VIEW, X_COMMON, or X_C_TYPE typed buffers	Set a list of view description file names to VIEWFILES. This variable is used by Agents running test scripts that use VIEW, X_COMMON or X_C_TYPE typed buffers. If this variable is not set, functions that use these typed buffers that are defined in view description files not in this list will fail, causing dependent commands to fail. Example: VIEWFILES=ct.V,inv.V Set the absolute pathname of the directory containing the view description files to VIEWDIR. If this variable is not set, tux_tpalloc() or tux_alloc_buf() calls that try to allocate a buffer of type VIEW, X_COMMON, or X_C_TYPE will fail, causing dependent commands or functions to fail. Example: VIEWDIR=/u1/tuxapp/dat:/u1/tuxapp/dat2

Testing Platform	System Environment Variable Settings
Local or Agents running TUXEDO test scripts that use VIEW32 typed buffers	Set a list of view description file names to VIEWFILES32. This variable is used by Agents running scripts that use VIEW32 typed buffers. If this variable is not set, functions that use VIEW32 typed buffers which are defined in view description files not in this list will fail, causing dependent commands to fail. Example: VIEWFILES32=ct32.V,inv32.V Set the absolute pathname of the directory containing the view description files to VIEWDIR32. This variable is used by Agents running test scripts that use VIEW32 typed buffers. If this variable is not set, tux_tpalloc() or tux_alloc_buf() calls that try to allocate a buffer of type VIEW32 will fail, causing dependent commands or functions to fail. Example: VIEWDIR32=/u1/tuxapp/dat
Solaris Agents running TUXEDO test scripts	Set the TLI network service provider pathname to WSDEVICE. This value is typically /dev/tcp. If not set, playback terminates with an error message. Example: WSDEVICE=/dev/tcp
INFORMIX on a UNIX Agent computer	Assign a valid entry in the $INFORMIXDIR/etc/sqlhosts file to INFORMIXSERVER. Assign a value to INFORMIXDIR. The value depends on your version of INFORMIX CLI and INFORMIX ESQL/C.

测试平台	**系统环境变量设置**
UNIX代理机 Oracle	在变量ORACLE_HOME中指定包含客户端软件的目录。 例如： ORACLE_HOME=/ora/app/oracle/product/8.0.5 如果/var/opt/oracle不包含tnsnames.ora，那么分配该文件的路径名到变量TNS_ADMIN。 例如：TNS_ADMIN=/home/uname/oracletest
UNIX代理机 Sybase	在变量SYBASE中指定包含客户端软件的目录。 例如：SYBASE=/usr/local/sybase 在以下的系统环境变量的一个路径中指定包含Sybase客户端库（libraries）的目录： PATH（Windows） LD_LIBRARY_PATH（Solaris Agents） SHLIB_PATH（HP-UX Agents） LIBPATH（AIX Agents）

UNIX代理机Java	在LD_LIBRARY_PATH变量中指定包含Java库（libraries）的目录。 例如： LD_LIBRARY_PATH=/usr/jre118/lib/linux/native_threads 当代理测试机也使用第三方软件（比如IBMWebSphere）时，在系统环境变量LD_LIBRARY_PATH中，除了Java库（libraries）之外，你必须指定那个软件库（libraries）的目录位置。 此外，对于Java Developers Kit 1.1（JDK1.1），你也必须设置下面的变量： JAVA_COMPILER=NONE
本地或代理机执行TUXEDO测试脚本	在TUXDIR变量中指定包含客户端软件的目录。 设置NLSPATH环境变量到包含了TUXEDO信息文件的目录路径。 设置值为$TUXDIR/LIB到下面的一个环境变量： LD_LIBRARY_PATH（Solaris Agents） SHLIB_PATH（HP-UX Agents） LIBPATH（AIX Agents） 对于Windows NT的本地测试机，这些只针对仅安装了TUXEDO客户端的测试机，必须被定义。TUXEDO完全运行安装的过程，会自动地设置它们。更多信息，参阅TUXEDO安装指南。 设置以下的一项： Workstation Listener（工作站接听者）的地址到WSNADDR。 例如： WSNADDR=//sparky：360001 WSNADDR=0028CAICA8F0D6 Workstation Listener（工作站接听者）的host名和端口到WSLHOST和WSLPORT。如果设置，这些变量会覆盖WSNADDR。 例如： WSLHOST= sparky WSLPORT=360001

本地或代理机执行TUXEDO测试脚本（使用FML类型的缓冲区字段名）	设置一列FML字段的表文件名到FIELDTBLS。该变量的使用，通过代理机执行包含了FML类型的缓冲区字段名索引的测试脚本。如果该变量不能设置，使用FML类型的缓冲区字段名（不算该列表）的函数将会失败，导致相关的命令失败。 例如：FIELDTBLS=ct.fldtbl，inv. fldtbl 设置包含FML字段的表文件目录的绝对路径到FLDTBLDIR。该变量的使用，通过代理机执行包含了FML类型的缓冲区字段名索引的测试脚本。如果该变量没有设置，使用FML类型的缓冲区字段名（不算该列表）的函数（例如，tux_setbuf_int（））将会失败，导致相关的命令失败。 例如：FLDTBLDIR=/u1/tuxapp/dat
本地或代理机执行TUXEDO测试脚本（使用FML32类型的缓冲区字段名）	设置一列FML32字段的表文件名到FIELDTBLS32。该变量的使用，通过代理机执行包含了FML32类型的缓冲区字段名索引的测试脚本。如果该变量不能设置，使用FML32类型的缓冲区字段名（不算该列表）的函数将会失败，导致相关的命令失败。 例如：FIELDTBLS=ct32.fldtbl，inv32. fldtbl 设置包含FML32字段的表文件目录的绝对路径到FLDTBLDIR32。该变量的使用，通过代理机执行包含了FML类型的缓冲区字段名索引的测试脚本。如果该变量没有设置，使用FML32类型的缓冲区字段名（不算该列表）的函数（例如，tux_setbuf_int（））将会失败，导致相关的命令失败。 例如：FLDTBLDIR32=/u1/tuxapp/dat
本地或代理机执行TUXEDO测试脚本（使用VIEW，X_COMMON，或X_C_TYPE类型的缓冲区字段名）	设置一列视图描述的文件名到VIEWFILES。该变量的使用，通过代理机执行包含了VIEW，X_COMMON，或X_C_TYPE类型的缓冲区的测试脚本。如果该变量不能设置，使用这些类型的缓冲区的函数（定义在视图描述文件而非此列表中）将会失败，导致相关的命令失败。 例如：VIEWFILES=ct.V，inv. V 设置包含视图描述文件目录的绝对路径到VIEWDIR。如果该变量没有设置，分配一个VIEW，X_COMMON，或X_C_TYPE类型缓冲区的函数tux_tpalloc（）或tux_alloc_buf（）的调用将失败，导致依赖的命令或函数也失败。 例如： VIEWDIR=/u1/tuxapp/dat:/u1/tuxapp/dat2

本地或代理机执行TUXEDO测试脚本（使用VIEW32类型的缓冲区）	设置一列视图描述的文件名到VIEWFILES32。该变量的使用，通过代理机执行使用了VIEW32类型缓冲区的测试脚本。如果该变量不能设置，使用VIEW32类型缓冲区的函数（定义在视图描述文件而非此列表中）将会失败，导致相关的命令失败。 例如：VIEWFILES32=ct32.V，inv32.V 设置包含视图描述文件目录的绝对路径到VIEWDIR32。如果该变量没有设置，分配一个VIEW32类型缓冲区的函数tux_tpalloc（）或tux_alloc_buf（）的调用将失败，导致依赖的命令或函数也失败。 例如：VIEWDIR32=/u1/tuxapp/dat
执行TUXEDO测试脚本的Solaris代理机	设置TLI网络服务供应商的路径到WSDEVICE。 一般情况下，该值为/dev/tcp。如果不进行设置，回放时跳出一个错误信息，同时该回防终止。 例如：WSDEVICE=/dev/tcp
一台UNIX代理测试机上的INFORMIX	在$INFORMIXDIR/etc/sqlhosts文件中分配一个合法表目到INFORMIXSERVER。 分配一个值到INFORMIXDIR。该值依赖于INFORMIX CLI和INFORMIX ESQL/C的版本。