WinRunner 7.0 使用说明（中文版）

第一版

WOZA

Email: lidaren@hotmail.com

目录

1. 简介

WinRunner (以下简称 WR)是基于 MS Windows 的功能测试工具。

由于 C/S 结构的软件功能增加越来越快，QA 部门测试难度越来越大，手工测试已经跟不上这种发展趋势。

WR 可以帮助你自动处理从测试开发到测试执行的整个过程。你可以创建可修改和可复用的测试脚本，而不用担心软件功能模块的变更。你只需要在下班后让计算机自动执行这些脚本，就能轻而易举的发现软件中的错误，从而确保软件的质量。

1.1 WinRunner 测试模式

当你在软件操作中点击 GUI（图形用户界面）对象时，WR 会用一种类 C 的测试脚本语言（TSL）生成一个测试脚本。你可以用手工编程的方法编辑这个脚本。WR 包括的功能生成器（Function Generator）可以帮助你快速简便的在已录制的测试中添加功能。WR 包括两种录制测试的模式：

1.1.1 环境判断模式（Context Sensitive mode）

这种模式根据你选取的 GUI 对象（如窗体、清单、按钮等）把你对软件的操作动作录制下来，并忽略这些对象在屏幕上的物理位置。每一次你对被测软件进行操作，测试脚本中的脚本语言会描述你选取的对象和你的操作动作。

当你进行录制时间，WR 会对你选取的每个对象做唯一描述并写入 GUI map(映射) 中。GUI map 和测试脚本被分开保存维护。当软件用户界面发生变化时，你只需更新 GUI map。这样一来，环境感应模式的测试脚本将非常容易地被重复使用。

执行测试只需要回放测试脚本。WR 模拟一个用户使用鼠标选取对象、用键盘输入数据。WR 从 GUI map 中读取对象描述，并在被测软件中查找符合这些描述的对象。WR 可以在同一个窗体中找到这些对象，即使它们的位置发生过变化。

1.1.2 模拟模式(Analog mode)

这种模式记录鼠标点击、键盘输入和鼠标在二维平面上（x 轴和 y 轴）的精确运动轨迹。执行测试时，WR 让鼠标根据轨迹运动。这种模式对于那些需要追踪鼠标运动的测试非常有用，例如画图软件。

1.2 WinRunner 测试过程

WR 的测试过程分六个步骤：

创建 GUI map

创建测试

调试测试

执行测试

查看测试结果

报告发现的错误

1.2.1 创建 GUI map

使用 RapidTest Script wizard(快速测试脚本巫师)回顾软件用户界面，并系统地把每个 GUI 对象的描述添加到 GUI map 中。你也可以在录制测试的时候，通过点击对象把对单个对象的描述添加到 GUI map 中。

注意：当使用 GUI map per test 模式，你可以跳过这一步骤。具体内容在 **WinRunner 如何识别 GUI 对象** 章节中。

1.2.2 创建测试

你可以通过录制、编程或两者同用的方式创建测试脚本。录制测试时，在你需要检查软件反应的地方插入检查点（Checkpoint）。你可以插入检查点来检查 GUI 对象，位图(Bitmap)和数据库。在这个过程中，WR 捕捉数据，并作为期望结果（被测软件的期望反应）储存下来。

1.2.3 调试测试

你可以先在调试模式（Debug mode）下运行脚本。你也可以设置中断点(Breakpoint)，监测变量，控制 WR 识别和隔离错误。调试结果被保存在 Debug folder，一旦调试结束就可以删除。

1.2.4 执行测试

你在检验模式(Verify mode)下测试被测软件。WR 在脚本运行中遇到检查点后，就把当前数据和前期捕捉的期望值进行比较。如果发现有不符合，就记录下来作为实测结果。

1.2.5 查看测试结果

测试是成功还是失败由你来认定。每次测试结束，WR 会把结果显示在报告中。报告会详述测试执行过程中发生的所有主要事件，如检查点、错误信息、系统信息或用户信息。

如果在检查点有不符合被发现，你可以在 Test Results（测试结果）窗口查看预期结果和实测结果。如果是位图不符合，你也可以查看用于显示预期值和实测结果之间差异的位图。

1.2.6 报告发现的错误

如果由于测试中发现错误而造成测试运行失败，你可以直接从 Test Results 窗口报告有关错误的信息。这些信息通过 EMAIL 发送给测试经理（QA Manager），用来跟踪这个错误直到被修复。

1.3 样本软件

本教程中很多例子使用 WR 附带的 Flight Reservation(航班预订)软件。

1.3.1 开始样本软件

样本软件位于：***开始>程序>WinRunner>Sample Application***。该程序有 2 个版本 Flight 1A 和 Flight 1B。

1.3.2 样本软件的多个版本

Flight 1A 这个版本是正常的软件，Flight 1B 有一些故意加入的错误。在 WinRunner Tutorial(WR 教学)中，两个版本都被使用来互相比较。本教程中的例子在两个版本中都可以使用。

如果 WR 中安装了 Visual Basic 支持，VB 版本的 Flight 1A 和 Flight 1B 将被安装到常规样本软件中。

1.3.3 登陆

使用任意用户名（长度必须至少 4 个字符）登陆 Flight Reservation(航班预订)软件，密码为：Mercury。

1.3.4 WEB 版样品软件

http://MercuryTours.mercuryinteractive.com；或点击***开始>程序>WinRunner>Sample Applications> Mercury Tours site***

1.4 使用 TestSuite(测试套件)

WR 和测试套件的其他工具一起提供整个测试流程的解决方案：测试计划、测试开发、GUI 负载测试、错误跟踪和多用户系统客户端负载测试。

1.4.1 TestDirector

TestDirector 是测试管理工具，帮助测试人员计划和组织测试活动。用 TD 可以创建手工和自动控制测试的数据库、建立测试流、执行测试、跟踪和报告错误。

1.4.2 LoadRunner

LoadRunner 是用于 Client/Server 结构软件(Browser/Server 结构也可以使用)的测试工具。可以模拟多个用户登陆到一台服务器的情况。

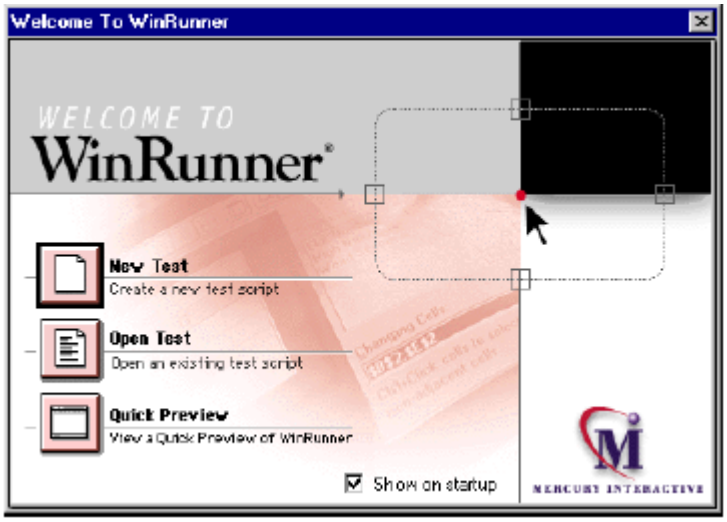
2. WinRunner 使用概述

本章介绍如何开始使用 WR，介绍 WR 的各个窗口。

2.1 启动 WinRunner

点击*开始>程序>WinRunner>WinRunner* 启动 WR。WR 的 Record/Run Engine(记录/执行引擎)的图标出现在 Windows 的任务条上。这个引擎设立和维护 WR 和被测软件之间的连接。

第一次启动 WR 会看到欢迎窗口，你可以选择新建测试、打开已有测试或快速预览 WR。如果不希望下次启动看到这个窗口，可以把 ***Show on startup*** 前面的勾去掉。



2.2 WinRunner 主窗口

主窗口包括以下部分：

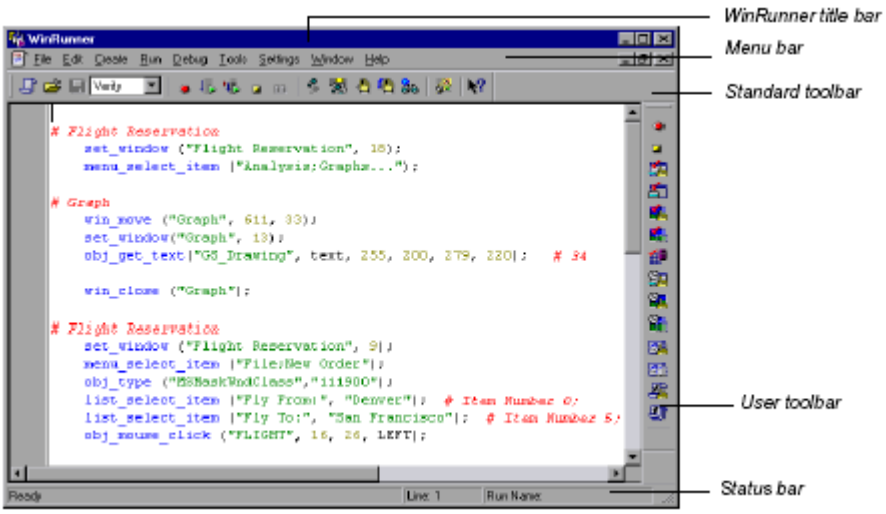
a). WinRunner title bar 标题栏

b). Menu bar 菜单栏

c). Standard toolbar 标准工具栏，包含运行测试时常用的命令

d). User toolbar 用户工具栏，包含创建测试时常用的命令

e). Status bar 状态栏，



2.3 测试窗口

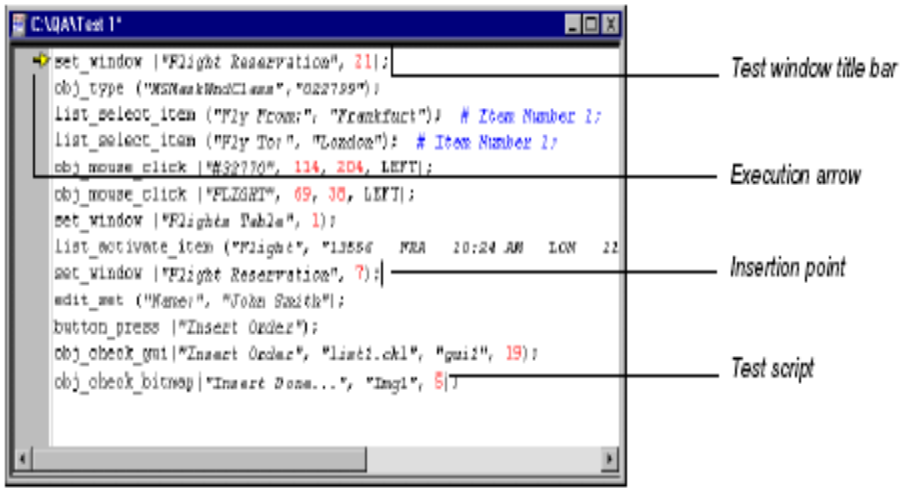
在测试窗口创建和执行测试，窗口包含以下部分：

a). Test Window title bar 测试窗口标题栏，显示当前打开的测试名称

b). Test script 测试脚本, 通过录制或编写代码方式生成

c). Execution arrow 执行箭头，指明当前正在执行的那一行脚本，如果想要移动这个标志到某一行，只需要在该行左侧空白处点击鼠标左键

d). Insertion point 插入点，支出你可以插入或编辑文本的地方



2.4 加载 WinRunner 插件

在 WinRunner 启动时，可以选择支持 ActiveX control、PowerBuilder、VisualBasic 或 WebTest 的插件。其他插件需要单独再向 MI 公司购买，建议不要同时载入所有的插件，不必要的插件可能会对录制或执行脚步造成问题。

把 ***Show on startup*** 前面的勾去掉，这个 Add－In Manager 的窗口就不会在 WR 启动的时候出现。你也可以在进入 WR 后在 ***Settings>General Options>Enviroment*** 里面设置是否在开始时显示这个窗口以及等待时间等。

3. WinRunner 如何识别 GUI 对象

3.1 关于识别 GUI 对象

当使用 Context Sensitive 模式时，这些 GUI 对象（如 Windows、Menus、Buttons、Lists）可以像用户看到的那样去测试。每个对象都有一组被定义的属性来决定它的行为和外观。WR 通过学习这些属性来识别和定位 GUI 对象，而不需要知道对象的物理位置。你可以使用 GUI spy 查看桌面上任何 GUI 对象的属性。想知道如何把属性教给 WR 请看有关**理解 GUI map** 的章节。

WR 把从 GUI map 上学来的信息储存起来。当执行测试时，WR 使用 GUI map 定位对象：先从 GUI map 读取有关对象的描述，然后寻找有相同属性的对象。你可以通过查看 GUI map 获得对象的全面图片。

GUI map 是一个或多个 GUI map 文件的总合。有两种方式组织 GUI map 文件：

- 你可以为整个软件创建一个 GUI map 文件，或者为每个窗体创建一个 GUI map 文件。多重测试可以参考同一个 GUI map 文件。这是 WR 的默认模式。对有经验的用户来说，这是最有效率的方式。具体内容请参考 **Global GUI map file 模式下工作**。
- WR 可以在每次创建新的测试时自动创建相关的 GUI map 文件，你无须担心有关 GUI map 文件的创建、保存或读取的问题。如果你是 WR 新手，这是最简单的使用方法。具体内容请参 **GUI map file per test 模式下工作。**

在测试过程的任何阶段，都可以在两种模式之间切换。具体内容请参考**合并 GUI map 文件。**

当 GUI 被修改了之后，你仍然可以使用先前的脚本。你只需要在 GUI map 上添加、删除或编辑相关的对象的描述，WR 就可以在修改后的软件上找到这些对象。具体内容请参考**编辑 GUI map**。

你可以指定 WR 使用某些属性去识别特定的一类对象。你可以教 WR 去识别自定义的对象，也可以把这些对象映射到标准对象上去。具体内容请参考**配置 GUI map**。

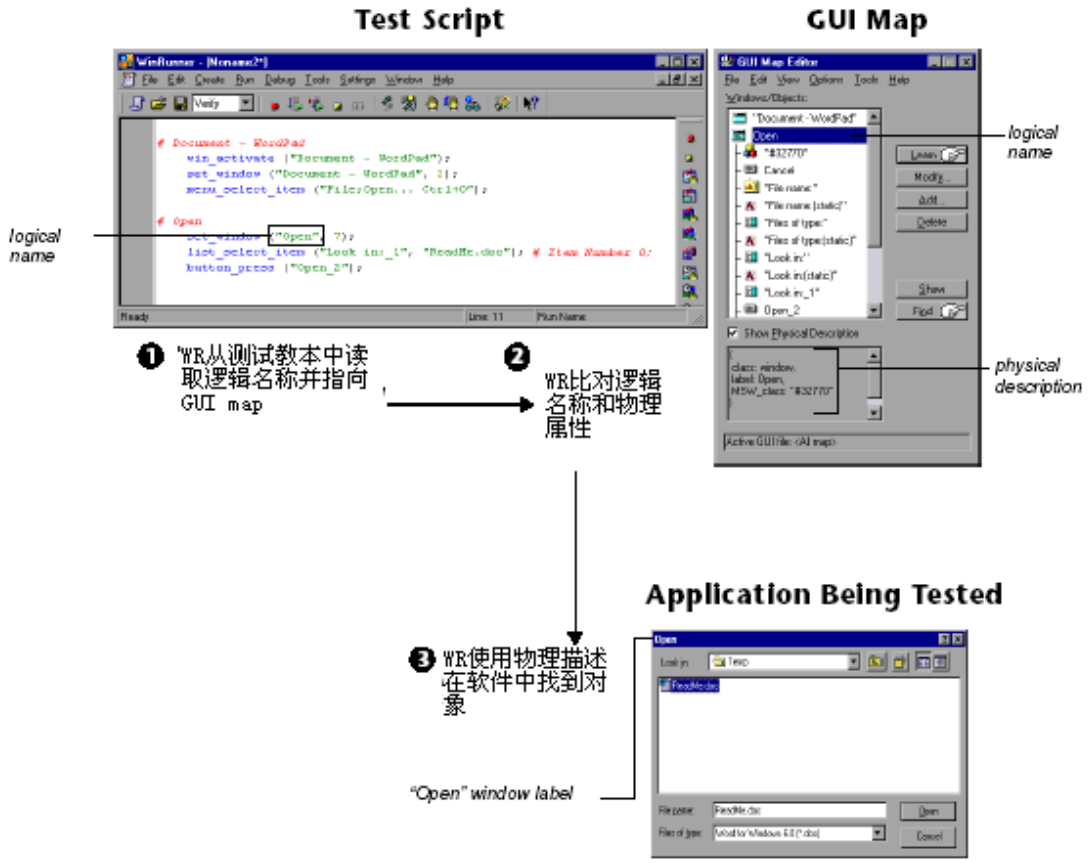
你也可以教 WR 在窗体上通过把位图定义为虚拟对象的方法识别为 GUI 对象。具体内容请参考**学习虚拟对象**。

3.2 测试中如何识别 GUI 对象

你通过录制或编程教本的方式创建测试，测试教本语言（TSL）重现了鼠标和键盘对被测软件的操作。具体内容请参考**创建测试**。

WR 使用逻辑名(Logic name)定义对象：如“Print”定义 Print dialog box, “OK”定义 OK button。逻辑名实际上是对象物理描述的昵称。物理描述是包含一个对象物理属性的清单，如：Print dialog box 被定义成一个有“Print”卷标（Label）的窗体。逻辑名和物理描述一起作用，确保每个 GUI 对象有自身唯一的标识。

3.3 物理描述（Physical Description）



WR 使用物理描述识别被测软件的 GUI 对象。物理描述包括：物理属性清单和每个属性的值。这些属性－值的配对在 GUI map 中以下面的格式出现：{属性 1：值 1,属性 2：值 2，property3：value3,….}

例如：对于“Open”window 的描述包含两个属性：类（Class）和卷标(Label)。类的属性的值是“window”，卷标的属性的值是“Open”：{class:window,label:Open}

类的属性标识对象的类型。每个对象根据功能不同属于不同的类：window, push button, list, radio button, menu 等。

每个类有默认的属性。有关各种属性的详细描述请参考 **GUI map 配置**。

注意：WR 总是在对象出现的窗体的环境中学到该对象的物理描述。因此给每个对象创建了一个唯一的物理描述。具体内容请参考**设定窗体环境**。

3.4 逻辑名（Logic Names）

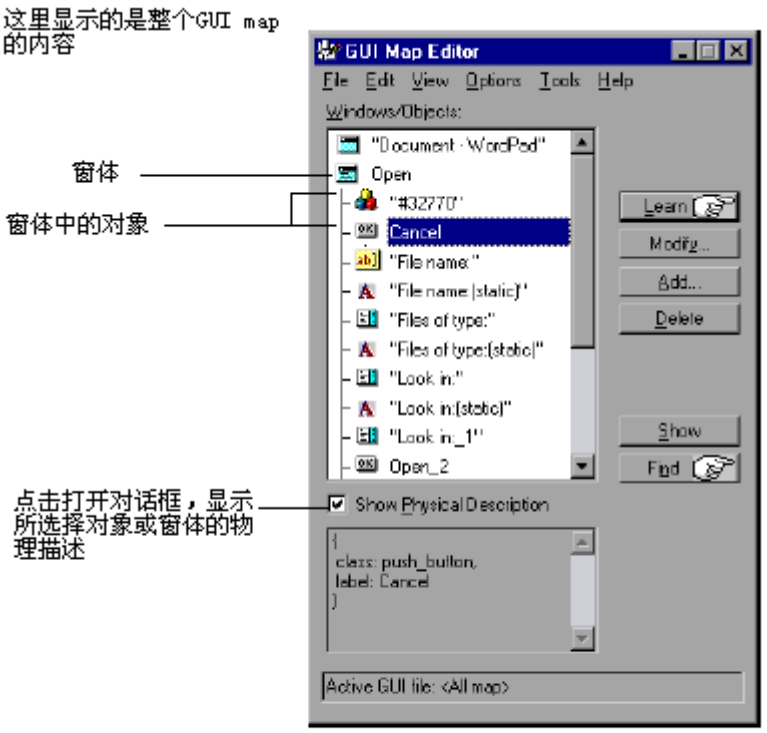
在测试脚本中，WR 不使用全部的物理描述来表示对象，而是给每个对象一个逻辑名。对象的逻辑名由它的类决定。多数情况下，逻辑名是对象的卷标：一个 button 的逻辑名就是它的卷标，如“OK”或“Cancel”；一个 window 的名称就是它标题栏上的文本；一个 list 的名称是 list 上方或旁边的文本。

对于一个 static text（静态文本）对象，逻辑名是 text 和字串（string）“（static）”的结合。如 static text “File Name”的逻辑名是：“File Name (static)”。

在特定情况下，一些在同一窗体中的对象会有相同的逻辑名加上一个位置选择符（location selector），如：LogicName_1, LogicName_2)。目的是给对象一个唯一名称。

3.5 GUI map

选择 ***Tools>GUI Map Editor*** 可以查看 GUI map 的内容。GUI map 实际上是一个或多个 GUI map 文件的总和。在 ***GUI Map Editor*** 中，你可以查看整个 GUI map 或单个 GUI map 文件的内容。GUI 对象按照他们在软件中出现时所在的窗体分组。有关 ***GUI Map Editor*** 的具体内容请参考**编辑 GUI map**。



3.6 设定窗体环境（Window Context）

WR 在对象出现的窗体环境中学习和实现操作。当你录制脚本时，一旦当前窗体（Active window）改变而且有 GUI 对象被操作，WR 会自动在测试脚本中插入一条 set_window 语句。然后所有的对象就在那个窗体环境中被识别。例如：

Set_window(“Print”,12);

Button_press(“OK”);

Set_window(“Print”,12)指明 Print window（打印窗口）是当前窗体。OK button 是在这个窗体环境里学到的。

如果你手工编写脚本，则需要你写入 set_window 语句以对应当前窗体的变化。同样在编辑脚本时注意不要随意删除必要的 set_window 语句。

4. 理解 GUI map

4.1 关于 GUI map

当 WR 运行测试时，它模拟一个真实的用户对软件的 GUI 对象用鼠标键盘进行操作。因此，WR 必须学习软件的 GUI。

WR 学习软件的 GUI 对象和对象的属性。你可以用 GUI Spy 查看任意 GUI 对象的属性，了解 WR 是如何识别它们的。

WR 通过以下方式学习软件的 GUI：

- 使用 RapidTest Script wizard 学习软件每个窗体中所有 GUI 对象的属性
- 通过录制脚本的方法学习被录制的那部分软件中所有的 GUI 对象的属性
- 使用 GUI Map Editor 学习单个 GUI 对象、窗体或某个窗体中所有 GUI 对象的属性

如果软件开发过程中 GUI 改变了，你可以使用 GUI Map Editor 更新 GUI map。

在你开始让 WR 学习软件的 GUI 之前，你需要确认你组织 GUI map 文件的方式：

- GUI Map per Test mode，为每个新建的测试创建一个新的 GUI map 文件
- Global GUI map File mode, 相关测试共享同一个 GUI map 文件

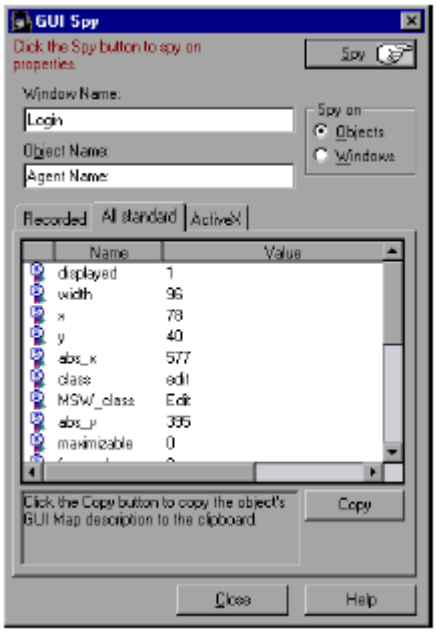
相关的具体内容请参考 **GUI map file 模式的选取**。

4.2 查看 GUI 对象属性

WR 学习 GUI 对象的描述时，它观察对象的物理属性。每个 GUI 对象都有很多属性，如“class”、“label”、“width”、“height”、“handle”和“enabled”。WR 只学习这些属性中的一组来把某个对象和软件中的其他对象区别开。

当你想查看 GUI 对象的属性时，可以使用 GUI spy。把 Spy pointer 指向某个对象，GUI spy 就在 GUI spy 对话框中显示属性和属性的值。你可以选择查看对象的所有属性或只是选定的一组属性。

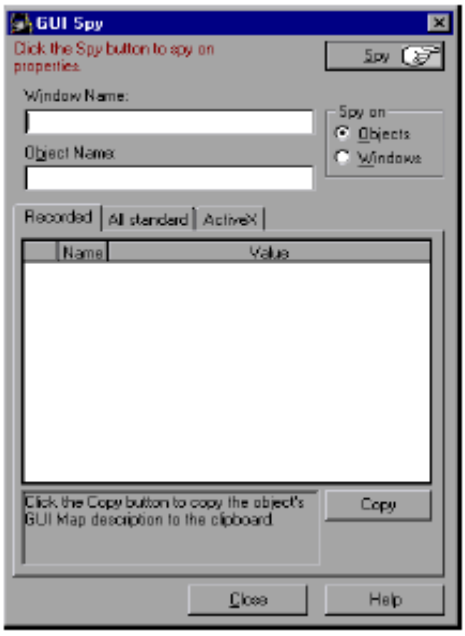
下面的例子是航班样品软件，当指向登陆窗口的 Agent Name edit box 时，在 GUI spy 里 All Standard Tab 里显示的内容：



注意：ActiveX tab 只有在 ActiveX 插件安装并加载后才能显示出来。

在 GUI 对象上使用 SPY：

1). 选择 Tools>GUI Spy 打开 GUI Spy 对话框



在 Spy on box 里点击 Windows 可以查看窗体的属性。

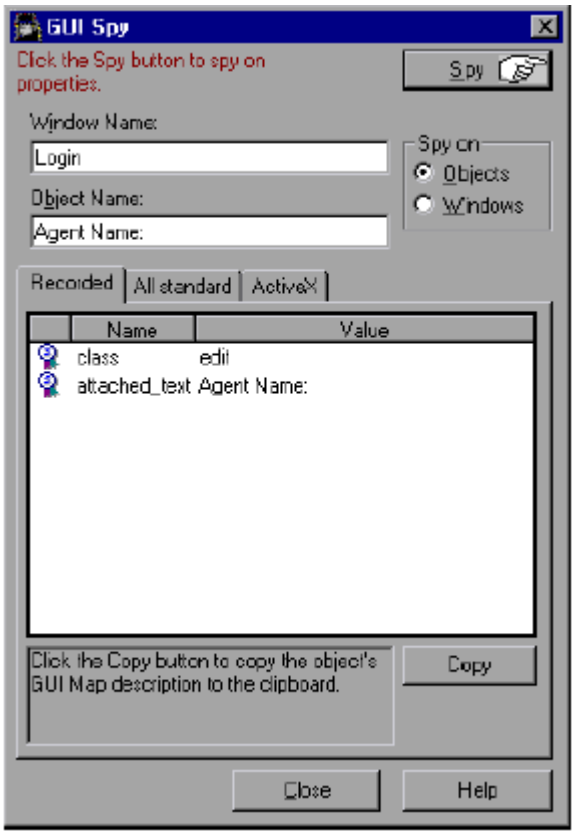
点击 All standard tab 查看所有标准窗体和对象的属性。

点击 ActiveX tab 查看所有 ActiveX 控制的属性和方法。

2). 点击 Spy，然后指向屏幕上的某个对象。对象会被加亮（highlight），而且当前窗口名称、对象名称和对象描述（属性和值）都显示在专用区域。

注意：当你指向其他对象时，每个对象都会被轮流加亮而且对象的描述会显示在 Description 格内。

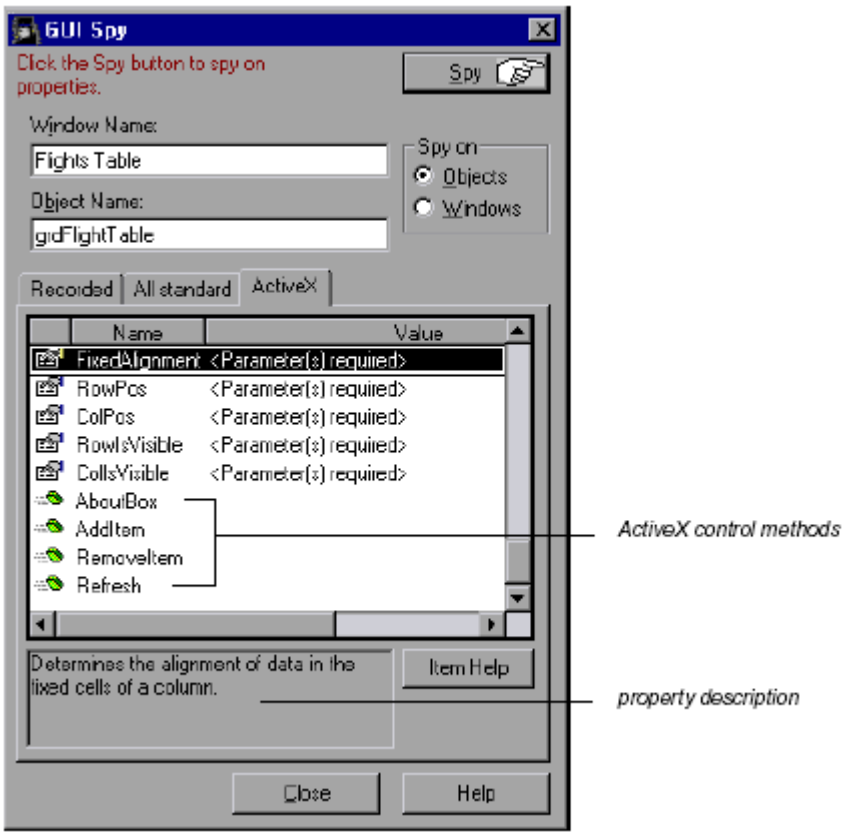
下面的例子是航班样品软件，当指向登陆窗口的 Agent Name edit box 时，在 GUI spy 里 Recorded tab 里显示的内容：



3). 想要把对象的描述捕捉到 GUI spy 对话框里，需要把鼠标指向目标对象并按下 STOP 热键（默认热键是左侧 Ctrl+F3）。

- 在 Recorded 和 All standard tabs 里，点击 Copy button 可以把对象物理描述复制到剪贴板里。在前例中如果复制的话，剪贴板里的内容将是：{class:"edit", attached_text:"Agent Name:", tag:"Agent Name:"}
- 在 ActiveX tab 里，当你加亮一个属性时，如果该属性包含描述时，这些描述会显示在底部的灰格里。

  下面的例子中，在 VB 版的航班样品程序里鼠标指向“Flights Table”，按下 STOP 热键后并选择 FixAlignment 属性，显示在 GUI Spy 中 ActiveX tab 里的的内容：



4). 点击 Close 关闭 GUI Spy。

4.3 教 WinRunner 被测软件的 GUI

WR 需要先学习软件的 GUI。

当使用 GUI Map File per Test mode 时，WR 会在录制脚本时自动学习软件的 GUI。

当使用 Global GUI Map File mode 时，你需要教给 WR 有关 GUI 对象属性的信息。

WR 可以通过以下方式学到信息：

- 使用 RapidTest Script wizard 学习软件每个窗体中所有 GUI 对象的属性
- 通过录制脚本的方法学习被录制的那部分软件中所有的 GUI 对象的属性
- 使用 GUI Map Editor 学习单个 GUI 对象、窗体或某个窗体中所有 GUI 对象的属性

注意：在 GUI Map File per Test 模式下，RapidTest Script wizard 将被禁用。

4.4 在 GUI map 中找到对象或窗体

当鼠标指针在测试脚本中和 GUI 对象或窗体对应的语句上时，你可以点击右键并选择 Find in GUI Map。WR 会在 GUI map(或 GUI map file)和对应被测软件（软件必须先打开）中找到并加亮这个对象或窗体。

注意：在 ***Tools>GUI Map Editor>***（打开 GUI Map Editor 后）***>View*** 里面选择 GUI Maps 或 GUI Files，将决定 WR 在 GUI map 或 GUI map file 里寻找对象。

- 如果窗体所在的 GUI map 文件已经被加载且窗体在被测软件中已经打开，那么 WR 打开 GUI Map Editor 且在 GUI map 和被测软件中加亮这个窗体。
- 如果对象所在的 GUI map 文件已经被加载且对象在被测软件中已经打开，那么 WR 打开 GUI Map Editor 且在 GUI map 和被测软件中加亮这个对象。
- 如果窗体或对象所在的 GUI map 文件已经被加载但是在被测软件中未打开，那么 WR 打开 GUI Map Editor 但只在 GUI map 中加亮这个窗体或对象。

4.5 GUI map files 使用概要

- 一个 GUI map 文件不能包含两个有相同逻辑名的窗体。
- 一个 GUI map 文件中的一个窗体不能包含两个有相同逻辑名的对象。
- 在 GUI Map Editor 中，可以使用 ***Options>Filter*** 命令打开 Filters(筛选)对话框，根据逻辑名、物理描述或 class 在 GUI map 中筛选对象。具体内容请参考**筛选显示的对象**。

4.6 GUI map file 模式的选取

计划和创建测试时，你需要确定 GUI map 的工作模式。

- 作为 WR 新手或 GUI map 只使用一次，可以使用 GUI Map File Per Test 模式。这种模式下，每次新建测试就自动新建一个 GUI map file。在你保存测试时，GUI map file 自动保存；在你打开测试时，GUI map file 自动加载。
- 作为 WR 老手或 GUI map 将被不同测试共享，建议使用更有效率的 Global GUI Map File 模式。这是 WR 的默认模式。WR6.02 或更低版本都是使用这种模式，且只能使用这种模式。

下面是两种模式的优缺点比较：

	GUI Map File Per Test	**Global GUI Map File**
优点	1. 每个测试都有自带的 GUI map file。 2. 使用方便，且避免忘记保存或加载 GUI map file。 3. 作为单次测试，容易维护和更新（就是重新录一次）。	1. 如果对象或窗体属性改变，只需要在 GUI map file 里把对应的属性修改。 2. 容易维护和更新（无需重新录制）
缺点	一旦软件 GUI 变更，每个测试的 GUI map file 都要重新录制。	要记住保存和加载 GUI map file。
建议	如果你对 WR 没有经验或被测软件的 GUI 已经固定，可以采用这种模式。	如果你是有经验的 WR 使用者或被测软件的 GUI 经常变化，最好使用这种模式。

注意：有时对象的逻辑名没有出现。如果在录制脚本前使用 GUI Map Editor 学习被测软件，你可以选择这个对象并点击 Modify 按键来修改它在 GUI map 中的名称。当 WR 录制脚本时，这个名称会出现在脚本中。具体内容请参考**修改逻辑名和物理描述**。

5. Global GUI Map File（共用 GUI 地图文件）模式的使用

5.1 关于 Global GUI Map File 模式

WR 最有效率的用法是把测试分组。一组中的测试（任务）都测试同一窗体上的 GUI 对象。这样这些任务就可以共享 GUI map file。当 GUI 发生变化，只需要修改一个 GUI map file，就可以让同组中的任务都正常工作。

在一个测试组中，某个测试（任务）可能只测试一个窗体内的特定几个 GUI 对象，而另外一个测试（任务）测试这个窗体中的上述对象中的一部分再加上其他对象（也必须在这个窗体中）。因此，如果只使用录制的方法让 WR 学习对象，WR 或许不能把窗体上所有的对象都学到（因为有对象没有被操作）。最好的方法是在录制前让 WR 全面学习被测软件中所有的 GUI 对象。

WR 有几种学习被测软件 GUI 的方法。通常使用 RapidTest Script wizard 在录制脚本前一次性的学习所有的 GUI 对象。这些 GUI 对象的物理描述保存在 GUI map 文件里。因为这些文件可以共享，其他用户就不需要再单独把 GUI 学习一次。

如果在软件开发过程中 GUI 发生变化，你可以用 GUI Map Editor 来学习单独的窗体或对象，并以此更新 GUI map。你也可以在录制脚本过程中让 WR 自动学习那些被操作的窗体或对象。这种方法比较快且简单，但不是系统性的方式，如果想做全面的测试最好不要用这种方法代替 RapidTest Script wizard。

注意：由于 GUI map file 独立于测试脚本，所以你关闭测试时系统不会自动保存。你一旦对 GUI map 作出修改要记住保存。同样在测试开始时，这些 GUI map file 也不会被自动加载。你可以用 GUI Map Editor 手工加载 GUI map 文件。有效的方法是在脚本开始部分插入 GUI_load 语句，这样脚本就会自动加载相关的 GUI map 文件了。

注意：当你在 Global GUI Map File 模式下工作，如果加载在 GUI Map File Per Test 模式下创建的和 GUI 对象相关的测试，这些测试运行起来可能会有问题。

5.2 测试中共享 GUI Map File

如前文所述，你可以设计让多个测试任务共享同一个 GUI map。

举例来说，假设在一个 Open 对话框中 Open button 改成了 OK button。你只要在 GUI map 中修改 Open button 的物理描述，把 button 的 label property 从 Open 改成 OK,修改如下：Open button:{class:push_button,label:OK}

在测试过程中，当 WR 在测试脚本中遇到 Open 对话框中的逻辑名“Open”时，就搜索含有 label “OK”的 push button。

你可以使用 GUI Map Editor 随时编辑逻辑名和物理描述。你也可以用 Run wizard 在测试过程中更新 GUI map。如果在运行测试时，WR 找不到某个对象，就自动打开 Run wizard。详细内容请参考**编辑 GUI Map**。

5.3 让 WinRunner 学习 GUI

5.3.1 RapidTest Script wizard 的使用

RapidTest Script wizard 只能在 WR 6.02 以上版本中使用，因为它只能在 Globe GUI Map File 的模式下使用，而 WR6.02 或以下版本没有这种模式。

在录制脚本前使用 RapidTest Script wizard 一次性学习被测软件所有的 GUI 对象，生成并保存 GUI map 文件后在脚本开头部分使用 GUI_load 语句加载这个 GUI map 文件。

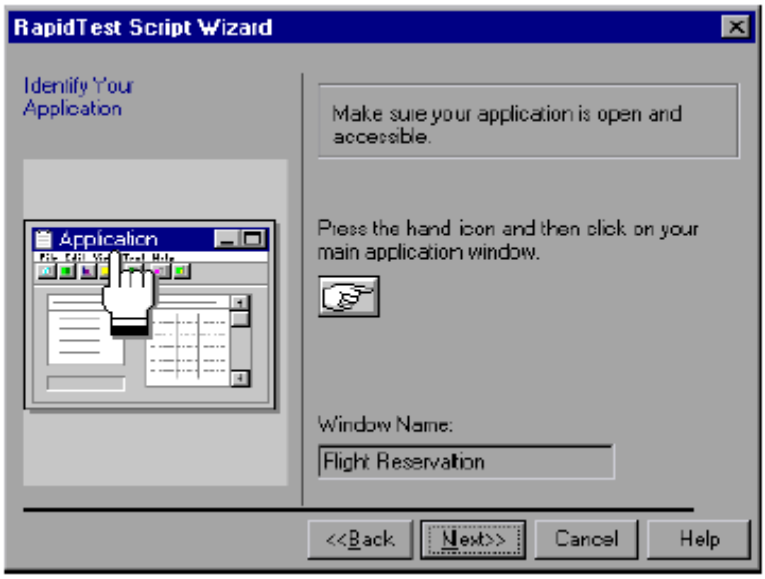
RapidTest Script wizard 的使用方法：

1). 选择 ***Create>RapidTest Script Wizard***

点击 ***Next***。

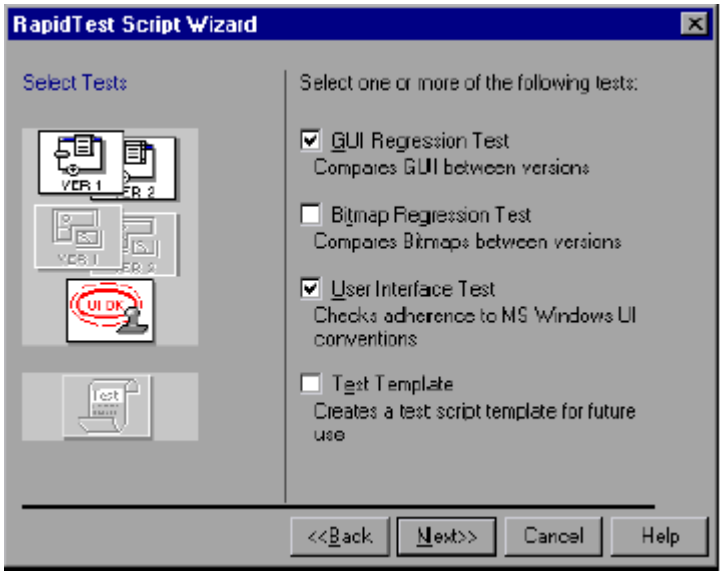
注意：当你载入 WebTest 插件或其他某些插件后，RapidTest Script wizard 将被禁用。

2). **标识被测软**屏幕打开



点击指向手（手型图标），然后点击被测软件。被测软件的窗体名称显示在 Window Name 框中。

3). **选择测试**屏幕打开



4). 选择你希望 WR 创建的测试类型。当 RapidTest Scipt Wizard 在被测软件中走查结束，你选择的测试就会被显示在 WR 窗口中。

你可以选择以下几种类型的测试：

- **GUI Regression Test**（界面回归测试）—用来比较软件不同版本中的 GUI 对象。例如检查一个 button 是否被禁用。创建这种测试时，WR 先捕捉 GUI 对象默认信息。在回归测试时，WR 把当前信息和默认比较，并报告不符合的地方。
- **Bitmap Regression Test** (位图回归测试)—用来比较软件不同版本中的位图图片。如果

被测软件没有 GUI 对象，则选择这种类型。创建这种测试时，WR 先捕捉被测软件每个窗体的一幅位图图片。在回归测试时，WR 把当前图片和以前捕捉的比较，并报告不符合的地方。

- **User Interface Test**(用户界面测试)－这种测试决定被测软件是否符合 Microsoft Windows 标准。它检查：

a). GUI 对象在窗体中的排列

b). 所有被定义的文本（text）在 GUI 对象上可见

c). GUI 对象上的卷标（Label）以大写字母写

d). 每个卷标包含一个有下划线的字母

e). 每个窗口有一个 **OK** button，一个 **Cancel** button，和一个系统菜单

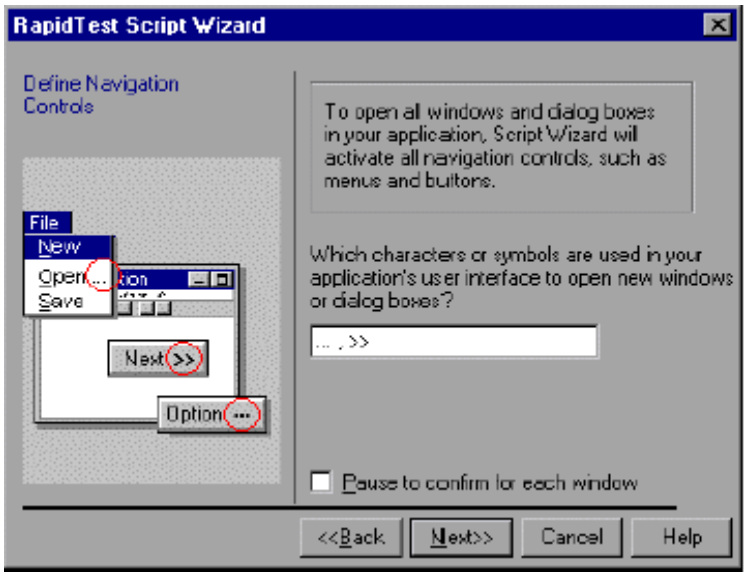
在这种测试中，WR 搜索软件 UI，把不符合 Microsoft Windows 标准的地方报告出来。

- **Test Template**(测试模板)－这种测试提供一个操作被测软件的自动测试的基本框架。它打开和关闭每个窗口，为你留下可以添加代码（手写或录制）的空间。

注意：即使你不想创建以上任何类型的测试，你仍然可以用 RapidTest Scipt Wizard 来学习被测软件的 GUI。

点击 ***Next***。

5). 定义导航控制（Navigation Control）

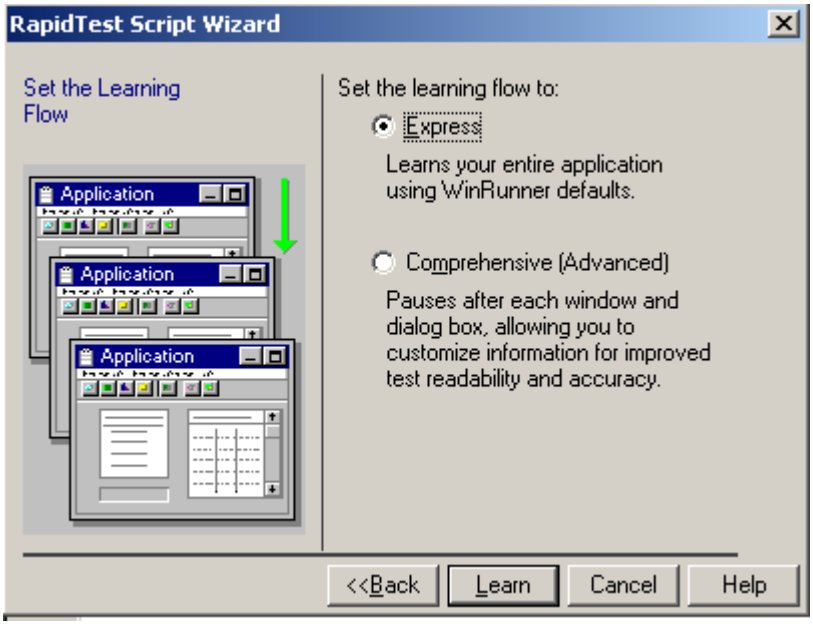


输入在被测软件中用作导航作用的字符。如果你需要在被测软件的每个窗口暂停以确

认用于打开其他窗口的对象，可以把 **Pause to confirm for each window** 的勾打上。

点击 ***Next***。

6). 选择 Express(快速)或 Comprehensive(全面)学习流程。点击 ***Learn***。WR 就开始系统地一个窗口一个窗口地学习被测软件。这个过程地时间长短取决于被测软件地复杂程度。



7). 选择 **Yes** 或 **No** 来告诉 WR 你是否希望在你使用 WR 时，让 WR 自动启动这个被测软件。

点击 Next。

8). 输入启动脚本和 GUI map 文件的保存路径和文件名，或使用默认值。

点击 Next。

9). 输入测试文件的保存路径和文件名，或使用默认值。

点击 Next。

10). 点击 **OK** 关闭 RapidTest Script wizard。你刚才创建的测试被显示在 WR 窗口中。

5.3.2 WinRunner 用录制的方式学习 GUI

WR 也可以通过在 Context Sensitive 模式（默认模式）下录制脚本的方法学习 GUI 对象。你只需要录制对被测软件的操作，WR 会自动学习操作中碰到的 GUI 对象。这个方法虽然快速简单，但是学习得不全面（你没有操作的对象就漏过不学了）。

当你开始录制时，WR 先检查对象是否已经存在于 GUI map 中；如果没有，就学习这个对象。

WR 先把学到的信息放在临时 GUI map 文件中。因此你在推出 WR 时要记住保存。详细内容请参考**保存 GUI map**。

如果你不希望 WR 把学到的信息添加到临时 GUI map 文件中，你可以在 General Options 对话框的 Environment 栏设置让 WR 不加载临时 GUI map 文件。详细内容请参考**设置 Global Testing 选项**。

总的来说，录制方式只用于小的或临时的测试。

5.3.3 WinRunner 用 GUI Map Editor 学习 GUI

1). 选择 ***Tools>GUI Map Editor*** 打开 GUI map 编辑器。

2). 点击 Learn。

- 想要学习一个窗体中所有的对象，就点击窗体的标题栏。当提示是否学习窗体中所有对象时，点击 **Yes**。
- 如果只想学习窗体，就点击窗体的标题栏。当提示是否学习窗体中所有对象时，点击 **No**。
- 如果只学习个别对象，就点击这个对象。

  （取消操作，电鼠标右键）

3). 把鼠标移动到对象上，点左键开始学习。WR 把学到的信息放在当前 GUI map 文件中。

5.4 保存 GUI Map

当你用录制的方式学习 GUI 对象时，对象的描述就被添加到临时 GUI map 文件中。临时文件总是打开的，因此它包含的任何对象都是 WR 认识的。你启动 WR 时，包含前一次测试内容的临时文件就被加载。

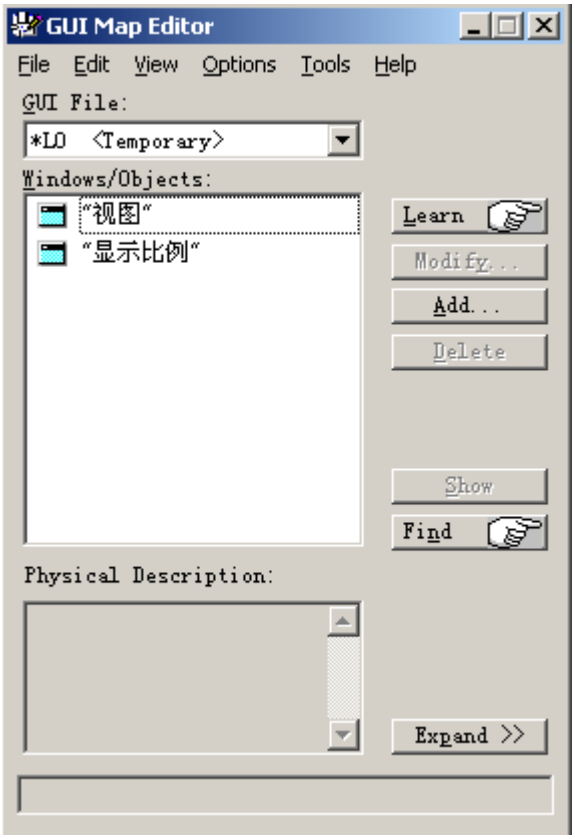
要避免在新的测试中把有价值的 GUI 信息覆盖掉，你必须把临时 GUI map 文件保存在一个永久的 GUI map 文件中。

方法是：

1). 选择 ***Tools>GUI Map Editor***，打开编辑器。

2). 选择 ***View>GUI Files***。

3). 确认<Temporary>文件显示在 GUI File 清单中。文件名前有一个星号（*）标识 GUI map 文件被改变了。当文件保存后，星号消失。



4). 在 GUI Map Editor 中，选择 ***File>Save*** 打开 Save GUI File 对话框。

5). 选择文件夹、文件名（或输入新文件名）。点击保存。被保存的 GUI map 文件被加载并显示在 GUI Map Editor 里。

你也可以把临时文件里的对象移动到一个已经存在的 GUI map 文件中。详细内容请参考**在文件之间复制和移动对象**。

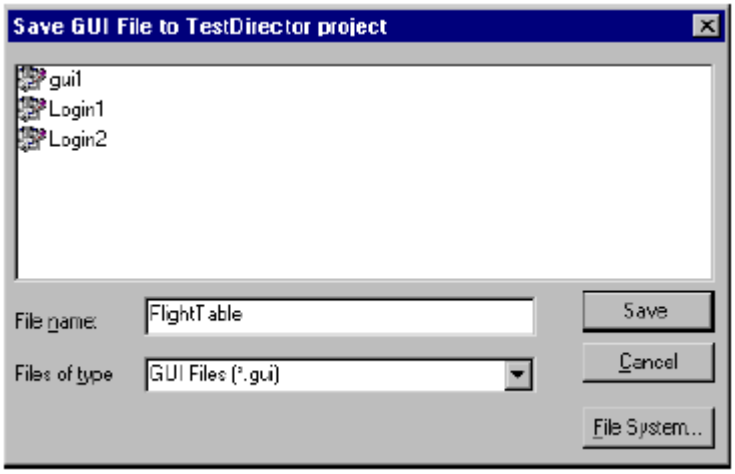
要把 GUI map 文件的内容保存到 TestDirector 的数据库中（你必须使用 TestDirector, 详细内容请参考**管理测试流程**）：

1). 选择 Tools>GUI Map Editor，打开编辑器。

2). 选择 ***View>GUI Files***。

3). 确认<Temporary>文件显示在 GUI File 清单中。文件名前有一个星号（*）标识 GUI map 文件被改变了。当文件保存后，星号消失。

4). GUI Map Editor 中，选择 ***File>Save*** 打开 Save GUI File to TestDirector 对话框。



5). 选择文件名（或输入新文件名）。点击保存。

5.5 加载 GUI Map 文件

WR 把对象的信息保存在一个 GUI map 文件中。当你需要用 GUI map 文件来寻找对象时，你必须把文件加载到 GUI map 中。你必须在测试开始前加载合适的 GUI map 文件。

有两种方法加载 GUI map 文件：

- 用 GUI_load 功能
- 从 GUI Map Editor 中加载

你可以在 GUI Map Editor 中查看被加载的 GUI map 文件。这类文件会在文件名前面出现一个“L”。你也可以打开一个 GUI map 文件来编辑但是不加载它。

注意：如果你使用 GUI Map File per Test 模式，你不能手工加载或卸除 GUI map 文件。

5.5.1 使用 GUI_load 功能加载 GUI map 文件

GUI_load 语句可以加载任何你需要的 GUI map 文件。虽然 GUI map 可能包含一个或多个 GUI map 文件，但你只能一次加载一个 GUI map 文件。想要加载多个 GUI map 文件，需要每个 GUI map 文件使用一个语句。你可以把 GUI_load 语句插到任何测试脚本的开头，但最好放到启动测试（第一个执行的测试）的开头。这样你启动 WR 的时候就自动加载这些 GUI map 文件。详细内容请参考**初始化特殊配置**。

使用 GUI_load 语句：

1). 选择 ***File>Open*** 打开你想要加载文件的测试。

2). 在测试脚本中输入，或在 Function Generator(功能生成器)中点选 GUI_load 功能，然后浏览或输入文件路径：

GUI_load(“file_name_full_path”);

例如：GUI_load(“c:\\qa\\flights.gui”);

有关 Function Generator 的使用请参考**生成功能**。

3). 执行这个测试就能加载文件了。

注意：如果你只想编辑 GUI map 文件，你可以用 GUI_open 功能打开 GUI map 文件来编辑，而不加载这个文件。用 GUI_close 功能关闭文件。用 GUI_unload 和 GUI_unload_all 功能卸除已经加载的 GUI map 文件。详细内容请参考**用编程的方法加强测试脚本**。

5.5.2 用 GUI Map Editor 加载 GUI map 文件

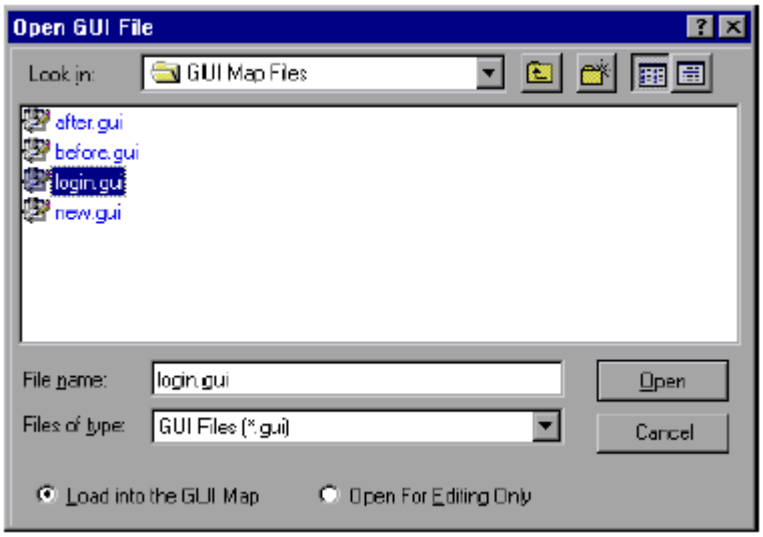
加载方法：

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 选择 ***File>Open***。

4). 在 Open GUI File 对话框中选择一个要打开的 GUI map 文件。



注意：这里的默认设置是打开 GUI map 文件并加载。如果你不想加载这个文件，可以点 Open For Editing Only，这样就只编辑而不会加载了。

5). 点击 **Open**。GUI map 文件被加载到 GUI 文件清单上。文件名前面会有一个“L”标明这个文件已经被加载了。

注意：从 TestDirector 数据库中加载 GUI map 文件的方法和上面的类似，但是前提是你必须使用 TestDirector 来管理这个测试。

5.6 Global GUI Map File 模式要点

- 为了提高性能，使用比较小的 GUI map 文件。把被测软件的 UI 按照窗口或其他规则划分成几个不同的 GUI map 文件。
- 有时对象的逻辑名不能被识别。如果你在录制前用 GUI Map Editor 学习对象，你可以修改 GUI map 中对象的逻辑名。然后在录制脚本时，新名字会出现在脚本里。如果你在修改逻辑名前录制了脚本，记住在运行脚本前在脚本里更新这个逻辑名。
- 不要把 WR 学到的 GUI 信息保存在临时 GUI map 文件里，因为当你离开 WR 时这些信息不会被自动保存。除非你创建临时测试，不然一定要在关闭测试前保存 GUI map。
- 如果使用录制的方法学习 GUI 对象，WR 只能学习那些录制过程中被操作的对象，而不是被测软件中所有的对象。如果你想要 GUI map 被复用，在录制前使用 RapidTest Script wizard 一次性学习所有的对象。
- 当被测软件的 GUI 改变时，最好有一位测试员作为“GUI Map 管理员”负责更新所有的 GUI map。

6. GUI Map File per Test 模式的使用

6.1 关于 GUI Map File per Test 模式

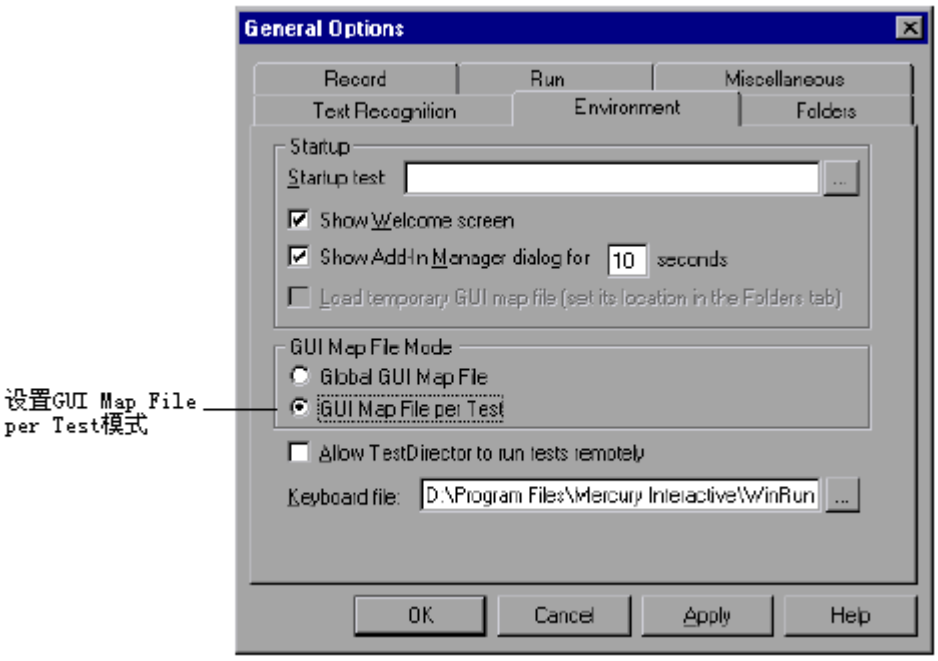
当你使用 GUI Map File per Test 模式，你不需要教 WR 去学习被测软件的 GUI，也不需要保存或加载 GUI map 文件。WR 会自动完成上述的一切。

在这种模式下，WR 在你创建新测试的时候自动创建一个新的 GUI map 文件；在你保存测试的时候自动保存 GUI map 文件；在你打开测试时自动加载 GUI map 文件。

注意：某些功能在这种模式下会被禁用：

- RapidTest Script wizard。详见在 **Global GUI Map File 模式的使用**。
- 在 WR 启动时重新加载前一次测试的临时 GUI map 文件（Load temporary GUI map file box）的功能。详见**设置 Global Testing 选项**。
- 当启动 WR 未加载 GUI map 文件，myinit 启动测试加载 GUI map 文件。有关启动测试详见**初始化特殊配置**。
- 编译过的模块没有加载 GUI map 文件。如果编译过的模块和 GUI 对象关联，那么这些 GUI 对象也一定和加载这些模块的测试相关联。详见**创建编译过的模块**。
- 如果一个被调用的测试（在 GUI Map File per Test 模式下创建的）和 GUI 对象关联，这个测试在 Global GUI Map File 模式下运行可能会有问题。

在 General Options 对话框中的 Environment 栏可以选择使用 GUI Map File per Test 模式。WR 重新启动后设置才会生效。



从 GUI Map File per Test 模式转换到 Global GUI Map File 模式，你必须把和每个测试对应的 GUI map 文件合并成被一组测试使用的共享文件。你可以使用 GUI Map File Merge Tool 合并。详见**合并 GUI Map 文件**。

6.2 GUI Map File per Test 模式下工作

在这种模式下，WR 通过录制的方式学习被测软件的 GUI。如果 GUI 发生变化，你可以用 GUI Map Editor 更新每个测试的 GUI map。你无需加载或保存 GUI map 文件。

注意：如果你改变了对象的逻辑名，你必须更新脚本。

6.3 GUI Map File per Test 模式要点

- 不要在 GUI Map Editor 里保存你对 GUI map 文件的修改。你保存测试时，这些变更会被自动保存。
- 不要手工加载或卸除 GUI map 文件。这些文件在你打开测试时会被自动加载。

7. 编辑 GUI Map

7.1 关于编辑 GUI Map

WR 使用 GUI map 来标识和查找 GUI 对象。一旦被测软件的 GUI 改变了，你就必须更新 GUI map 中对象的描述。

你有两种更新 GUI map 的方法：

- 在测试运行中使用 Run wizard。

  测试中如果 WR 找不到对象就会自动打开 Run wizard。它会指导你识别对象并把对象的描述更新到 GUI map 里。
- 用 GUI Map Editor 手工编辑 GUI map。

在你更新 GUI map 前，GUI map 必须先被加载。

注意：如果使用 GUI Map File per Test 模式，你不可以手工加载或卸除 GUI map 文件。

7.2 运行巫师（Run Wizard）

Run wizard 在测试运行中检测被测软件 GUI 的变化。当 WR 无法定位对象时，它就自动打开。它会提示你指向对象，确定该对象没有被发现的原因，然后提供解决方案。例如：它可能会建议你加载正确的 GUI map 文件。多数情况下，它会自动给 GUI map 添加新的描述或修改已有的描述。当这个过程结束，测试将继续。下次执行测试时，WR 就会找到这个对象了。

假设你在测试中在一个叫 **Open** 的窗体中点击 **Network** button。脚本会显示：

set_window(“Open”);

button_press(“Network”);

如果 **Network** button 不在 GUI map 中，Run wizard 就会打开并描述问题。

点击 **Hand** button 然后指向 **Network** button。Run wizard 就建议一个解决方案。

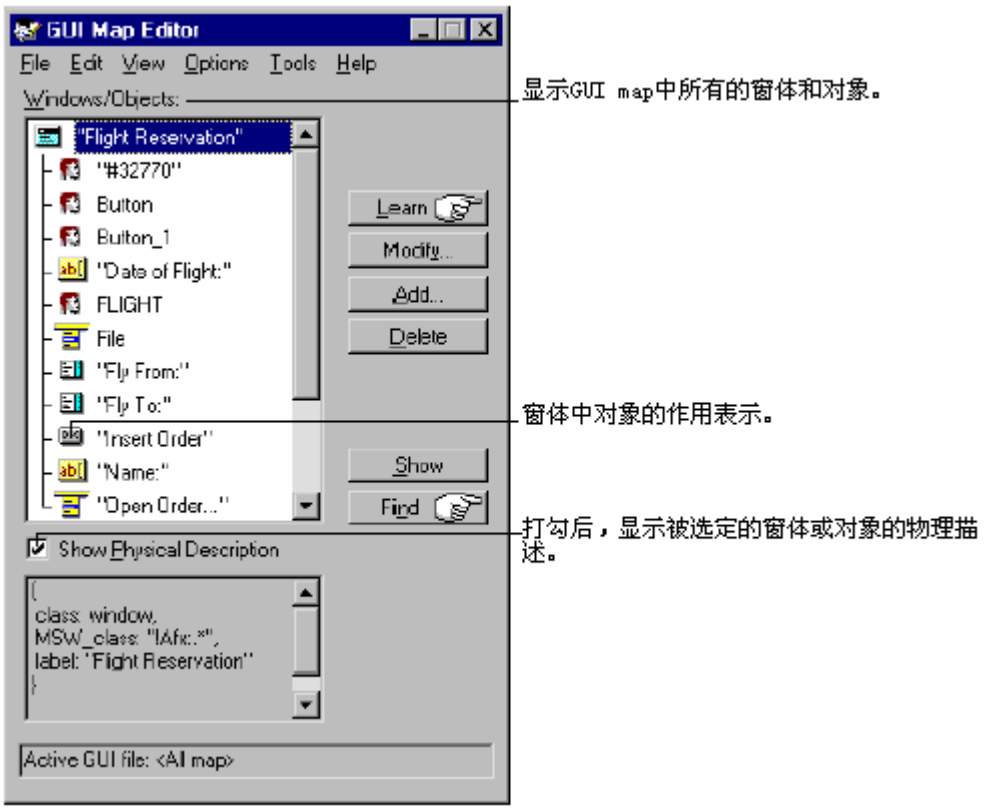
点击 **OK**，**Network** 这个对象的描述就自动添加到 GUI map 里，然后 WR 继续执行测试。

有些情况下，Run wizard 会编辑测试脚本而不是 GUI map。例如，如果 WR 不能找到对象是由于相关窗口没有激活，那么 Run wizard 就在测试脚本中插入 set_window 语句。

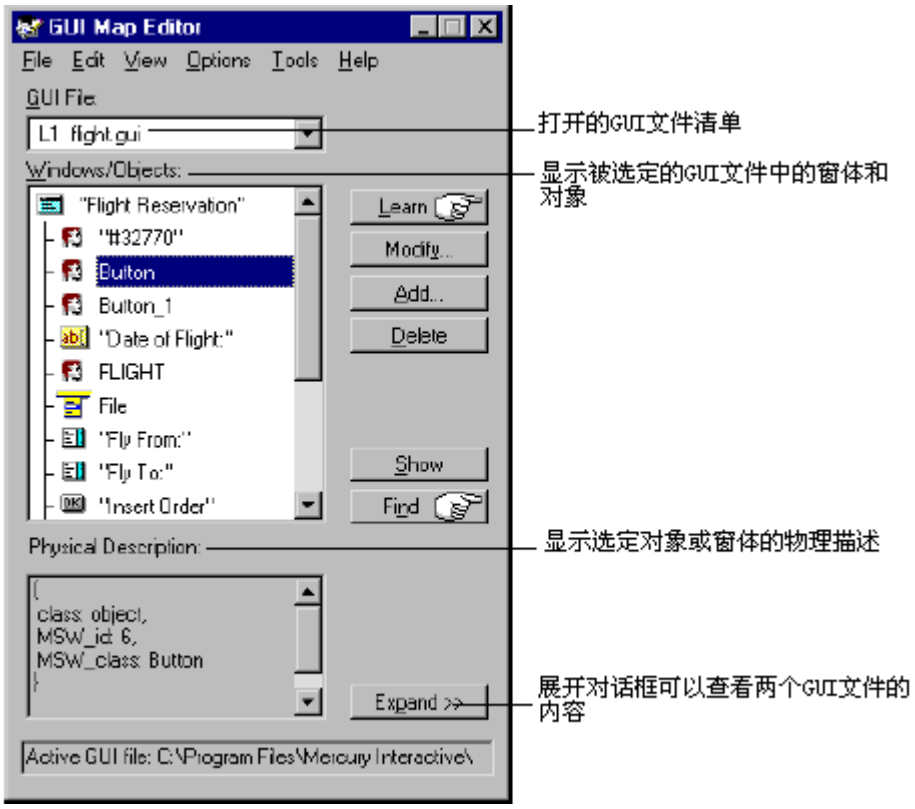
7.3 GUI Map 编辑器

选择 ***Tools>GUI Map Editor*** 打开。在编辑器里有两种查看模式，可以让你看到：

- 整个 GUI map
- 或单个 GUI map 文件



当你查看特定的 GUI map 文件内容时，你可以同时查看两个 GUI map 文件。这样你就可以方便地在文件之间复制或移动对象描述。需要查看单个 GUI map 文件的内容，选择 ***View>GUI Files***。



在编辑器中，对象在窗体图标下以树型结构显示。当你双击窗体名或图标，你可以查看它所包含的所有对象。想要查看树结构中所有的对象，选择 ***View>Expand Object Tree***。如果只想查看窗体，选择 ***View>Collapse Objects Tree***。

当你查看整个 GUI map 时，你可以选择 **Show Physical Description** check box 来显示在 **Windows/Objects** 清单中你选定的对象的描述。当你查看单个 GUI map 文件时，物理描述自动显示。

注意：如果你在 GUI map 里修改了对象的逻辑名，那你必须在脚本里也修改这个对象的逻辑名。

注意：如果属性值包含任何空格或特殊字符，这个值必须用引号关闭。

7.4 修改逻辑名和物理描述

在 GUI Map Editor 中你可以修改对象的逻辑名或物理描述。

当对象被赋予的逻辑名太长或含义不清晰时，就需要修改使脚本容易阅读。

当对象属性值变化时，就需要修改物理描述。例如，一个 button 的卷标从”Insert”改成”Add”，你就需要在物理描述里把 label 属性的值从

Insert button:{class:push_button,label:Insert}，改成

Insert button:{class:push_button,label:Add}

操作方法：

1). 选择 ***Tools>GUI Map Editor*** 打开编辑器。

2). 选择 ***View>GUI Files***。

3). 如果相关 GUI map 文件未加载，就选择 ***File>Open*** 打开文件。

4). 双击 Windows/Objects 栏里窗体的名字可以看窗体包含的对象。

5). 选定对象或窗体的名字，然后修改。

6). 点击 **Modify** 打开 Modify 对话框。

7). 编辑逻辑名或物理描述，完成后点 **OK**。修改后的内容立刻显示在 GUI map 文件中。

7.5 WinRunner 处理可变的窗体卷标

窗体经常使用可变的卷标。比如，Mircrosoft Word 在打开不同文件名的文件时，主窗体标题栏显示的名字都不同。

如果窗体被 WR 学习过后，由于窗体名称的变化使 WR 不能识别，那么 Run wizard 就会提示你识别这个窗体。一旦你识别完成，WR 就明白这个窗体有一个可变的卷标，它就会相应修改窗体的物理描述。

假设你录制了 Microsoft Word 的主窗体。WR 学到的物理描述为：

{

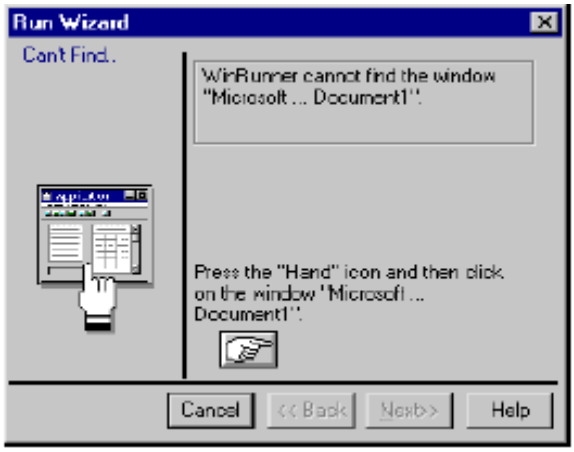
class: window,

label: “Microsoft Word – Document1”,

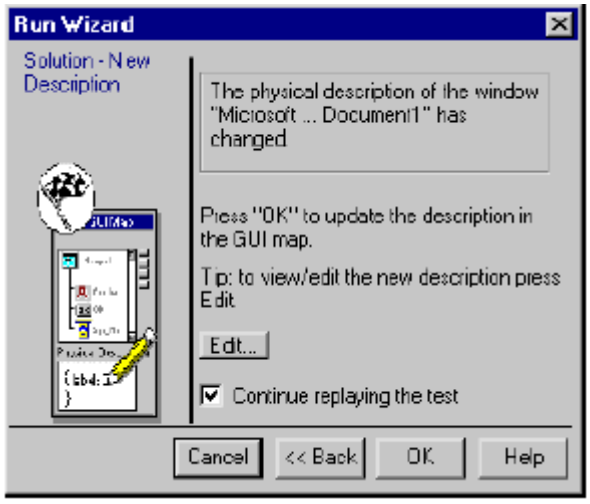
MSW_class: OpusApp,

}

然后当你在 Word 中打开 Document2 的时候执行测试。WR 找不到主窗体，就打开 Run wizard：



你点击 Hand button 然后再点 Word 窗口，WR 就会学习这个窗体。然后 Run wizard 会提示你在 GUI map 里更新这个窗体的描述。



如果你点击 Edit，你可以看到 WR 已经把窗体的物理描述改成常规表达式：

{

class: window,

label: “!Microsoft Word – Document.*”,

MSW_class: OpusApp

}

这个常规表达式使 WR 在识别 Microsoft Word 窗体时不管”Document”后面出现什么名字。

7.6 在物理描述中使用常规表达式

WR 为了在对象物理描述中使用常规表达式，就使用两个隐藏属性：regexp_label 和 regexp_MSW_class。

Regexp_label 属性只被用于窗体。它在后台操作：在窗体的卷标描述中插入一条常规表达式。注意：在 Windows 95 中使用 WR 时，这个属性不是必须的，所以 WR 不会记录或学习它。

Regexp_MSW_class 属性在对象的 MSW_class 中插入一条常规表达式。这个属性对所有类型的窗体和 object class（对象类）对象都是必须的。

7.6.1 添加常规表达式

你可以把 regexp_label 和 regexp_MSW_class 属性添加到 GUI 配置里。如果被测软件里存在对象的卷标或 MSW 类包含可以被安全忽略的字符，你就可以使用常规表达式。

7.6.2 禁用常规表达式

你可以禁用窗体物理描述中的常规表达式。比如用 label 属性在必须学习的属性清单上替换 regexp_label 属性。具体内容请参考**配置 GUI Map**。

有关常规表达式的更多内容请参考**使用常规表达式**。

7.7 在文件间复制和移动对象

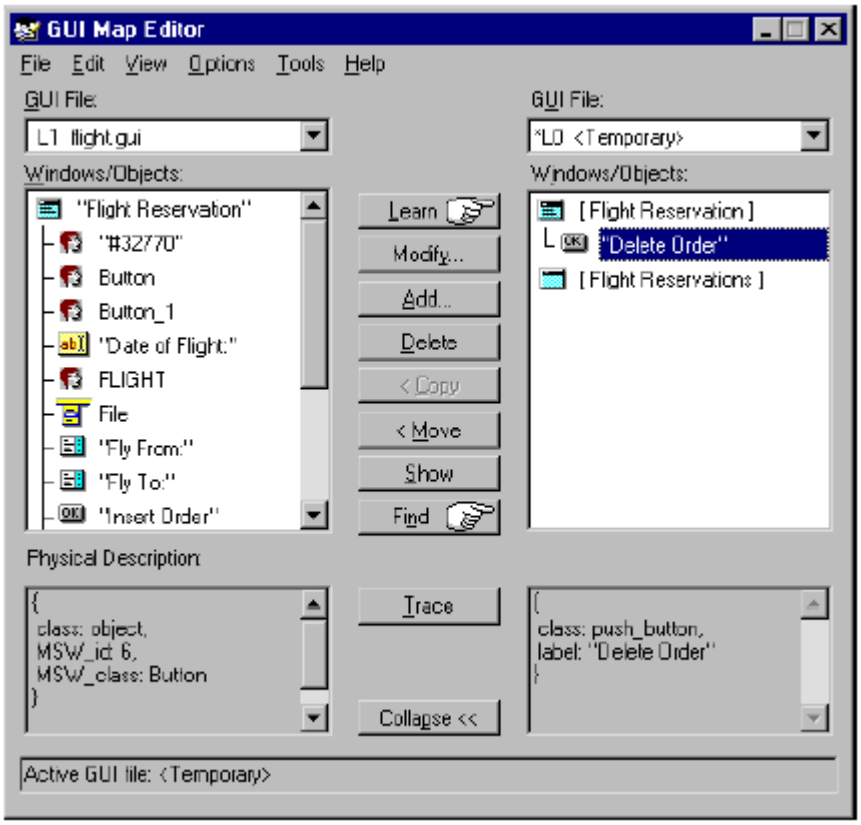
你可以用从一个 GUI map 文件复制或移动 GUI 对象到另一个 GUI map 文件的方式更新 GUI map 文件。注意：你只能从一个被打开但是未被加载的 GUI map 文件中复制对象。如果你使用 GUI Map File per Test 模式，就不可以手工打开或在文件之间复制或移动对象。

复制或移动对象的方法：

1). 选择 ***Tools>GUI Map Editor*** 打开编辑器。

2). 选择 ***View>GUI Files***。

3). 在编辑器中点击 **Expand**。对话框就会展开并同时显示两个 GUI map 文件。



4). 如果想在两边查看其他文件，可以在 **GUI File** 清单里选择。

5). 在一个文件中，选定你想复制或移动的对象。用 Shift 或 Ctrl 复选。用 ***Edit>Select All*** 选择一个 GUI map 文件中的全部对象。

6). 点击 **Copy** 或 **Move**。

7). 点击 **Collapse** 复原 GUI Map Editor。

注意：如果你从一个已经加载的 GUI map 文件中添加新窗体到临时 GUI map 文件，然后当你保存临时 GUI map 文件时，**New Windows** 对话框会打开。它会提示你添加新窗体到已经被加载的 GUI map 文件中或把它们保存到一个新的 GUI map 文件中。

7.8 在 GUI Map File 里找到对象

通过点击被测软件中的对象，你可以很容易地在 GUI map 文件里找到它的描述。

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 选择 ***File>Open*** 加载 GUI map 文件。

4). 点击 **Find**。鼠标会变成指向手(pointing hand)。

5). 点击被测软件中的对象。这个对象会在 GUI map 文件被加亮。

7.9 在多个 GUI Map File 里找到对象

如果一个对象在多个 GUI map 文件都被描述，你可以在 GUI Map Editor 里用 Trace(跟踪) button 快速找到所有的对象描述。如果你想让 WR 学习某个对象的新描述，而且在其他 GUI map 文件中找到并删除旧的描述，那么这个功能就非常有用。

操作方法：

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 选择 ***File>Open*** 打开（用 **Open for Editing Only** 模式）对象描述可能出现的 GUI map 文件。

4). 在 GUI File 栏里显示包含该对象最新描述的 GUI map 文件的内容。

5). 在 Windows/Objects 栏里选定这个对象。

6). 点击 **Expand** 展开对话框。

7). 点击 **Trace**。被发现包含这个对象的 GUI map 文件就会显示在对话框的另一边，且对象会被加亮。

7.10 在 GUI Map File 里手工添加对象

你可以通过复制另一个对象的描述的方式在 GUI map 文件中手工添加对象。如果有必要，你也可以编辑这些描述。

操作方法：

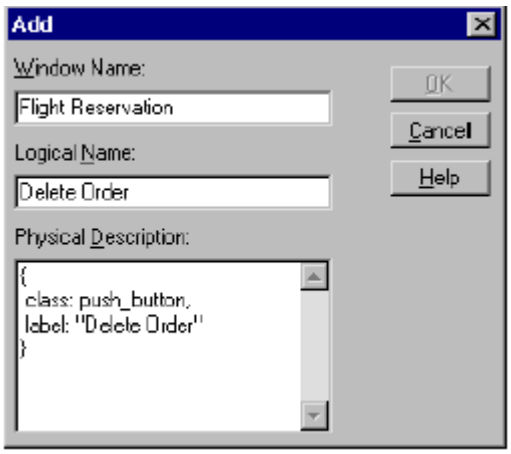
1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 选择 ***File>Open*** 打开相关的 GUI map 文件。

4). 选择一个作为编辑基础的对象（你将借用这个对象的描述）。

5). 点击 **Add** 打开对话框。



6). 编辑完相关栏目后点击 **OK**。这个新对象就会被添加到 GUI map 文件中。

7.11 从 GUI Map File 里删除对象

7.11.1 从 GUI map 文件中删除一个对象的方法操作：

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 选择 ***File>Open*** 打开相关的 GUI map 文件。

4). 选定想要删除的对象。可以用 Shift 或 Ctrl 选定多个对象。

5). 点击 **Delete**。

6). 选择 **File>Save** 保存。

7.11.2 从 GUI map 文件中删除全部对象的方法操作

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 选择 ***File>Open*** 打开相关的 GUI map 文件。

4). 选择 ***Edit>Clear All***。

7.12 清除 GUI Map File

你可以临时 GUI map 文件或其他 GUI map 文件的所有内容。操作方法：

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***View>GUI Files***。

3). 打开相关的 GUI map 文件。

4). 显示位于 GUI 文件清单最上方的文件。

5). 选择 ***Edit>Clear All***。

7.13 筛选显示对象

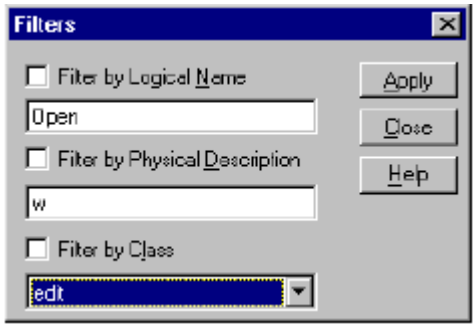
你可以使用以下筛选方式筛选 GUI Map Editor 里显示的对象：

- Logical name － 只显示含有特定逻辑名或逻辑名一部分（substring of logical name）的对象，如”Close”或“Cl”。
- Physical description – 只显示含有特定物理描述的对象。使用物理描述中的某个字符串，如使用“A”那么所有物理描述中包含“A”的对象都会显示。
- Class － 只显示属于某个特定的类的对象。

操作方法：

1). 选择 ***Tools>GUI Map Editor***。

2). 选择 ***Options>Filters*** 打开筛选对话框。



3). 选择筛选方式。

4). 点击 **Apply**。GUI Map Editor 会显示筛选结果。

7.14 保存 GUI Map 的变更

在 GUI map 中编辑对象的逻辑名和物理描述或在 GUI map 文件中修改对象或窗体后，你必须在离开 WR 前保存这些变更。

你有两种方法保存变更：

- 在 GUI Map Editor 中选择 ***File>Save*** 把变更保存到相关 GUI map 文件中。
- 选择 ***File>Save As*** 保存到一个新的 GUI map 文件中。

注意：如果你从一个已经加载的 GUI map 文件中添加新窗体到临时 GUI map 文件，然后当你保存临时 GUI map 文件时，**New Windows** 对话框会打开。它会提示你添加新窗体到已经被加载的 GUI map 文件中或把它们保存到一个新的 GUI map 文件中。

8. 合并 GUI Map File

8.1 关于合并 GUI Map File

GUI Map File Merge Tool（GUI map 文件合并工具）允许你把多个 GUI map 文件合并成一个文件。在合并前，你必须指定至少两个源文件和至少一个目标文件。目标 GUI map 文件可以是已经存在的文件或新文件。

你可以在自动模式或手工模式下使用这个工具。

- 在自动模式下，合并工具自动合并文件。如果被合并的文件之间有冲突，这些冲突会被加亮而且你会被提示去解决它们。
- 在手工模式下，你必须手工把 GUI 对象添加到目标文件中去。合并工具不会加亮文件之间的冲突。

在两种模式下，合并工具都会在合并文件时防止你制造冲突。

合并完成后，你必须改变 GUI map 文件的模式（使用 Global GUI Map File mode），而且修改你的脚本去加载正确的 GUI map 文件。

8.2 合并 GUI Map File 的准备

开始合并前，你必须决定合并类型、源文件和目标文件。

操作方法：

1). 选择 ***Choose>Merge GUI Map Files***。

WR 会弹出一个消息框提示你：所有打开的 GUI map 文件将被关闭而且所有未保存的变更都会被放弃。

- 点击 **OK** 继续。
- 如果你想要保存变更到 GUI map 文件，点击 **Cancel** 并使用 GUI Map Editor。当你保存结束后，重新开始步骤 1。

GUI Map File 合并工具打开后，你就可以选择合并类型、源文件和目标文件。

2). 在合并类型（Merge Type）框选择自动合并（Auto Merge）或手工合并(Manual Merge)。

3). 指定目标文件时，点击浏览按键。

- 选择一个已经存在的 GUI map 文件，点击 **OK** 后将覆盖这个文件。
- 输入文件名创建一个新的 GUI map 文件。

4). 指定源文件。

- 想把一个文件夹中所有文件添加到源文件清单上，你可以点击 Browse Folder 选择文件夹。
- 添加单个文件到源文件清单上，点击 **Add File**。
- 想从源文件清单上删除文件，先在 **Source** 框中选定想删除的文件，然后点击 **Delete**。

5). 点击 **OK** 关闭对话框。

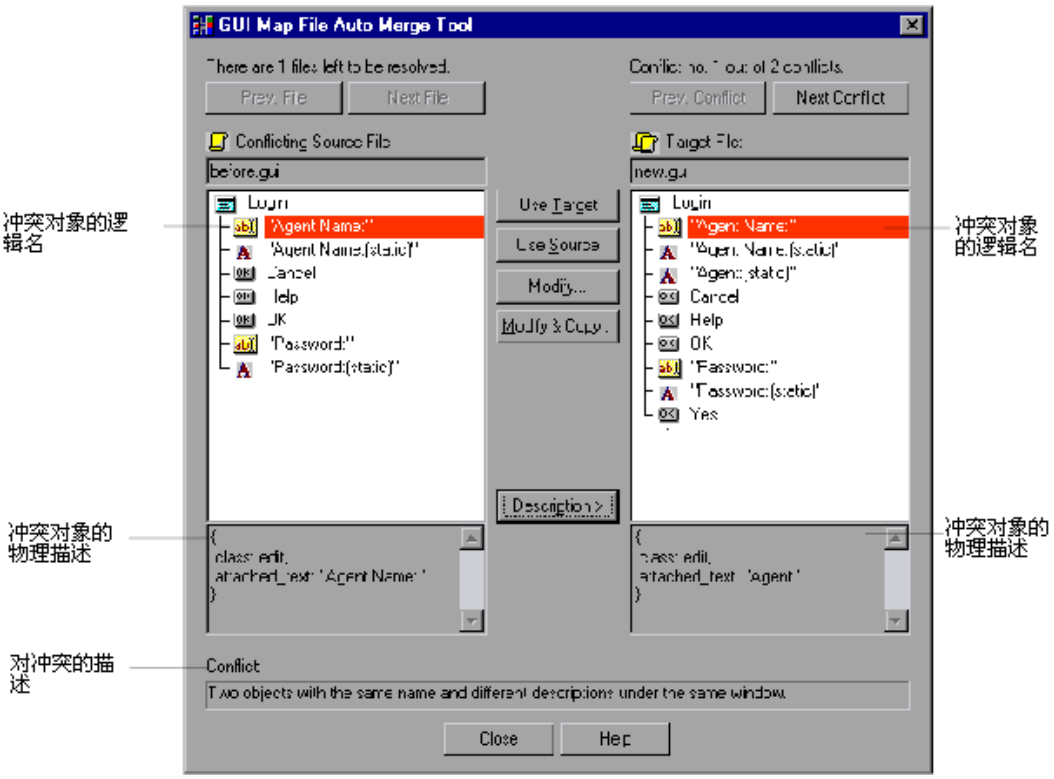
- 如果你选择自动合并，且合并成功，WR 会发消息确认。
- 如果你选择自动合并，但是源文件有冲突，WR 会弹出消息框警告。当你点 OK 关闭消息框时，GUI Map Auto Merge Tool 会打开。详见**解决自动合并 GUI Map File 的冲突**。
- 如果你选择手工合并，GUI Map File Manual Merge Tool 会打开。详见**手工合并 GUI Map File**。

8.3 解决自动合并 GUI Map 文件的冲突

自动合并 GUI map 文件时，会在下面的几种情况下发生冲突：

- 两个窗体有不同的物理描述，但是有相同的名称。
- 同一窗体中的两个对象有不同的物理描述，但是有相同的名称。

下面是自动合并时，冲突发生的例子。



发生冲突的对象被加亮，且冲突描述显示在对话框底部。冲突原因是同一窗体中的两个对象有不同的物理描述，但是有相同的名称。

点击 **Description** 查看冲突对象或窗体的物理描述。

用以下按钮可以解决被加亮的冲突。注意：只有当冲突对象或窗体在两侧都被加亮时，这些按钮才能使用：

冲突解决选项	描述
Use Target（使用目标文件)	使用目标 GUI map 文件里的对象或窗体的逻辑名和物理描述来解决冲突。
Use Source（使用源文件）	使用源 GUI map 文件里的对象或窗体的逻辑名和物理描述来解决冲突。

Modify（修改）	如果可能，使用可以同时适合目标和源文件中对象或窗体物理描述的常规表达式来解决冲突。你可以修改这个描述。
Modify&Copy（修改和复制）	编辑源文件中对象或窗体的物理描述来解决冲突。 注意：你所做的修改不会被保存在源文件中。

注意：

- 如果在多个源文件中都存在冲突，你可以点击 **Prev. File** 或 **Next File** 来切换。
- 如果一个源文件中存在多个冲突，你可以点击 **Prev.Conflict** 或 **Next Conflict** 来切换。

一旦当前目标文件和源文件之间的冲突都被解决后，源文件会被自动关闭而且下一个有冲突的源文件会被打开。当所有冲突都被解决后，剩余的源文件和目标文件（不存在冲突的文件）都会被关闭，且 GUI Map File Auto Merge Tool 也会关闭。

注意：

- 有时，你在当前源文件里解决冲突的方法也会被用于其他源文件（但是这些解决方法你不想用来解决其他源文件里的冲突）。这时，**Remove File** 按钮会出现。点击后，当前源文件就从源文件清单里消失了。
- 你对目标文件做的修改会被自动保存。

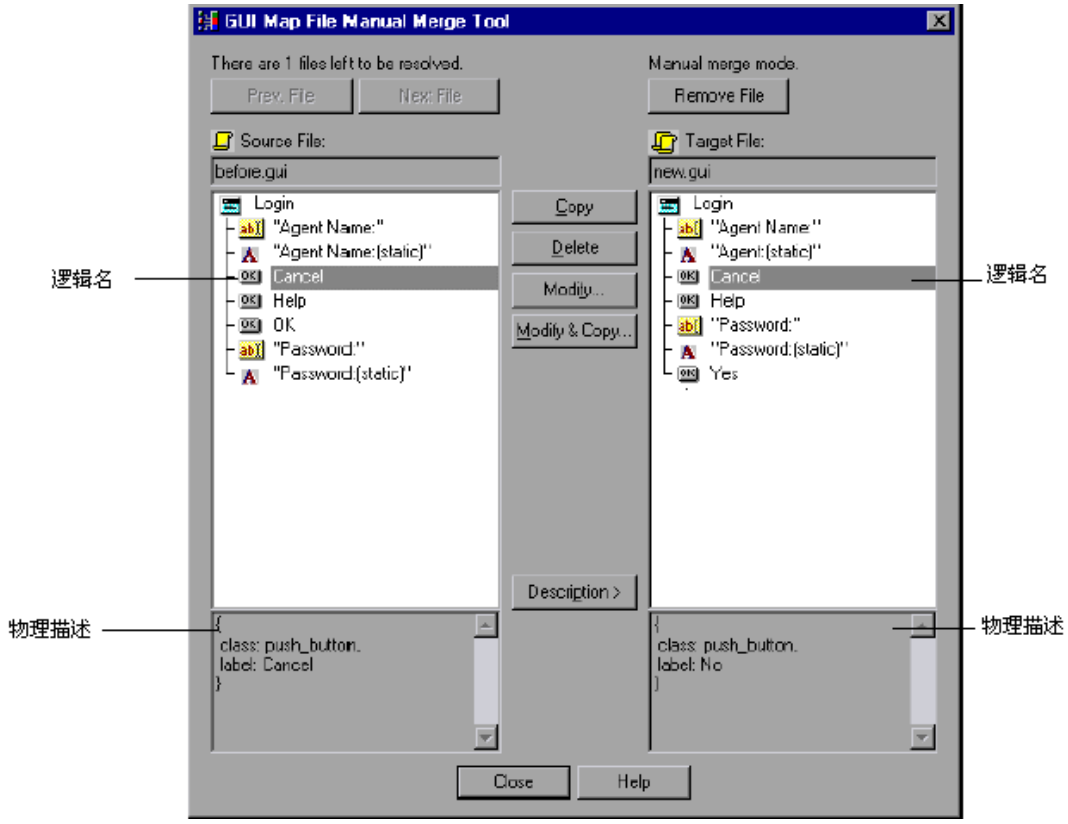
8.4 手工合并 GUI Map 文件

如果你手工合并 GUI map 文件，每个目标文件你都要用源文件来合并。合并工具会防止你在合并时制造冲突。

手工合并时，目标 GUI map 文件不能包含以下内容：

- 两个窗体有相同名称，但是物理描述不同。
- 两个窗体有相同名称和相同物理描述。
- 一个窗体中的两个对象有相同名称，但是物理描述不同。
- 一个窗体中的两个对象有相同名称和相同物理描述。

下面是一个手工合并时发生冲突的例子：



手工合并操作方法：

1). 与**合并 GUI Map File 的准备**相似，选择合并类型：**Manual Merge**。选定源文件和目标文件并点击 **OK** 后，GUI Map File Manual Merge Tool 就会打开。源文件和目标文件的内容都会显示出来。

2). 选择合并对象或窗体。

- 双击窗体可以看到窗体包含的对象。
- 如果有多个源文件，可以点击 **Prev. File** 或 **Next File** 来切换。
- 点击 **Description** 查看被加亮的对象或窗体的物理描述。

3). 使用以下合并选项：

合并选项	描述
Copy（当对象或窗体在当前源文件中被加亮时才可用）	从源文件中把加亮的对象或窗体复制到目标文件的窗体或父窗体中。 注意：复制窗体时会同时复制窗体中所有的对象。
Delete（当对象或窗体在目标文件中被加亮时才可用）	从目标文件中删除被加亮的对象或窗体。 注意：删除窗体时会同时删除窗体中所有的对象。
Modify（当对象或窗体在目标文件中被加亮时才可用）	打开 **Modify** 对话框，修改目标文件中被加亮的对象或窗体逻辑名或物理描述。
Modify&Copy（当对象或窗体在当前源文件中被加亮时才可用）	打开 **Modify** 对话框，修改源文件中被加亮的对象或窗体逻辑名或物理描述，并复制到目标文件的窗体或父窗体中。 注意：你在源文件中做的修改不会被保存在源文件中。

注意：合并过的源文件可以用 **Remove File** 从文件清单上清除掉。你对目标文件做的改动会被自动保存。

8.5 改变到 Global GUI Map File 模式

从 GUI Map File per Test 模式改变到 Global GUI Map File 模式，最复杂的准备工作就是合并 GUI map 文件。另外，你还必须做以下的相应改变：

- 修改启动测试，你必须加载正确的 GUI map 文件。
- 你必须在 General Option 里选择 Global GUI Map File 模式，保存变更并重新启动 WR。
- 从此以后，你对 GUI map 文件做的任何修改都必须保存。

9. 配置 GUI Map

9.1 关于配置 GUI Map

被测软件中每个 GUI 对象都有多个属性，如 class（类）、label（卷标）、MSW_class（微软窗体类）、MSW_id（微软窗体标识符）、x（X 坐标）、y（Y 坐标）、width（宽度）和 height（高度）。WR 在 Context Sensitive 测试时使用这些属性来标识对象。

当 WR 学习对象时，它并不学习对象所有的属性，而是学习可以标识唯一对象的最少属性。对每个对象的类（如 push_button, list, window,或 menu），WR 学习默认的一组属性：这就是 GUI map 配置。

例如：一个标准的 push button 有 26 个属性，如 MSW_class, label, text, nchildren, x, y, height, class, enabled。在多数情况下，WR 只需要 class 和 label 就可以创建一个唯一标识来识别这个 push button。

许多软件包含自定义的 GUI 对象。自定义对象指任何不包括在 WR 使用的标准对象类中的对象。因此，这些对象被归到一般对象类中。当 WR 在录制中遇到一个自定义对象，就会在测试脚本中生成 obj_mouse_语句。

如果一个自定义对象和一个标准对象相似，你可以把它映射到一个标准对象类中。你也可以配置 WR 用来识别自定义对象的属性。无论是映射还是配置，都只能用于当前的测试进程中。如果想要永久保留映射或配置，你必须把配置语句添加到启动脚本中。每次你启动 WR，启动测试都会激活这个配置。

注意：如果你的软件包含自己画的 button，你可以把它们都映射到一个标准 button 类上，不需要每个都分开映射一次。你可以在 **General Option** 对话框 **Record** 栏里 **Advanced Record Options** 的 **Record Owner-Drawn Buttons** 里选一个标准 button 类。你也可以在脚本中设置 rec_owner_drawn 测试选项和 setvar 功能。有关 General Options 对话框详见**设置 Global Testing 选项**。关于 setvar 功能，详见**从测试脚本中设置测试选项**.。

对象属性根据它们的可移动程度变化。有些是不可移动的（只能在唯一平台上），如 MSW_class 或 MSW_id。有些是半移动（支持多平台，但是有个值可能会变化），如 handle,或 Toolkit_class。其他都是完全可移动，如 label, attached_text, enabled, focused 或 parent。

9.2 理解 GUI Map 的默认配置

对于每个类来说，WR 都学习一组默认属性。每个默认属性都被分成“必须（obligarory）”或“可选（optional）”。

- obligarory 属性只要存在 WR 就一定会学习。
- optional 属性只有当 obligarory 属性不足以提供唯一标识的时候才使用。这些可选属性储存在一个清单里。WR 从清单中挑选足以唯一标识对象的最小数量的属性。它从清单第一个属性开始，一个接一个把属性添加到描述里，直到足以唯一标识对象。

如果你用 GUI Spy 查看一个 **Cancel** button 的默认属性，你会看到 WR 学习了 class 和 label 属性。这个 button 的物理描述是：{class: push_button, label:”Cancel”}。

在必须和可选属性都不能唯一标识对象时，WR 使用一个选择符(selector)。例如：如果一个窗体中有两个 **OK** button 都有同样的 MSW_id，WR 会使用选择符来区分它们。选择符有两种：

- 位置选择符（location selector）用对象的空间位置（上下左右）。
- 索引选择符（index selector）用一个唯一数字来标识一个窗体中的对象。如果对象的位置可能发生变化，就要用这种选择符。

9.3 把自定义对象映射到标准的类

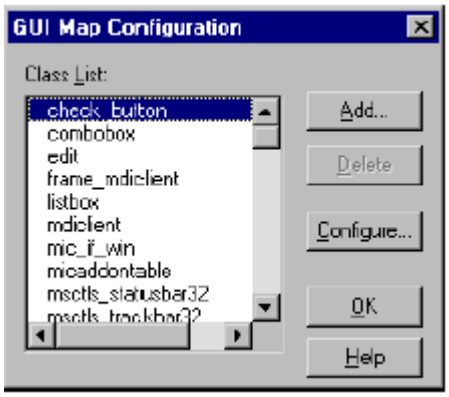
不属于 WR 使用的任何标准的类的对象就是自定义对象。当录制时遇到这种对象，WR 就使用 obj_mouse_ 语句。

通过 **GUI Map Configuration** 对话框，你可以教 WR 自定义对象并把它映射到一个标准类。例如：被测软件中有一个 WR 不能识别的自定义 button，WR 把点击这个的 button 的操作录成 **obj_mouse_click**。你可以教 WR 并映射到标准 push_button 类。以后当你点击这个 button 的时候，WR 就会录成 **button_press**。

注意：自定义对象只能被映射到和它行为类似的标准类中去。比如你不能把自定义 push button 映射到 edit 类。

自定义对象映射到标准类的操作方法：

1). 选择 ***Tools>GUI Map Configuration***。Class 清单显示所有标准类和 WR 识别出的自定义类。

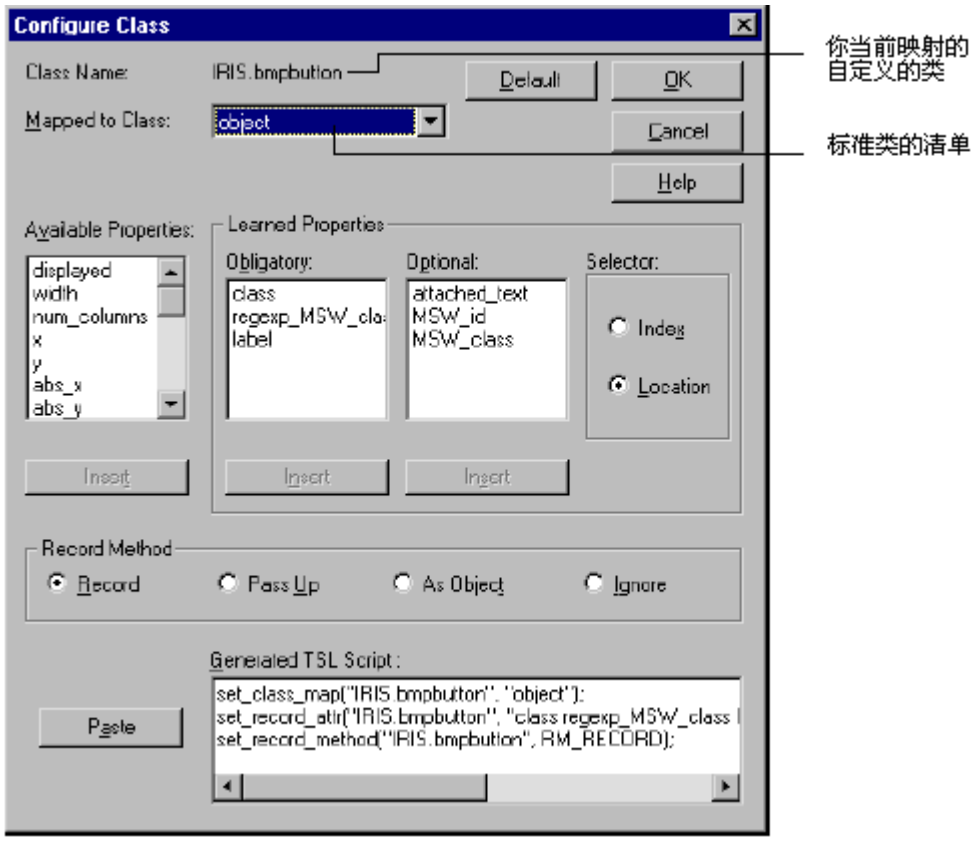


2). 点击 Add 打开添加类对话框。

3). 点击手型图标然后点击想添加的类的对象。自定义对象的名称会出现在 Class Name 框里。这个名称是对象的 MSW_class 属性的值。

4). 点击 **OK** 关闭对话框。新添加的类会在 Class 清单的底部并被加亮，而且前面会有个“U”（user-defined 用户定义）。

5). 点击 **Configure** 打开配置类对话框。



6). 在 **Mapped to Class** 清单中，选择你想要映射的标准类。选择后，对话框会显示这个类的 GUI map 配置。你也可以修改自定义类的 GUI map 配置（包括要学习的属性、选择符和录制方法）。

7). 点击 **OK** 完成配置。

注意：配置只适用于当前的测试。想要永久保留这个配置，你需要把 TSL 语句粘贴到启动脚本中。

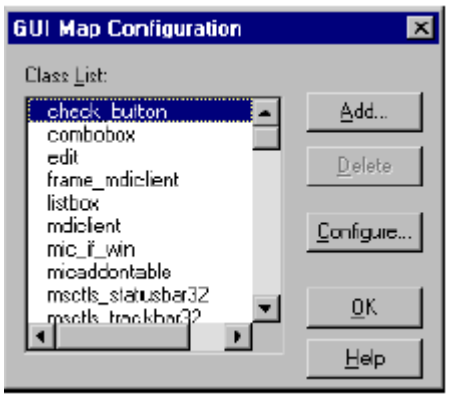
9.4 配置标准或自定义的类

对于任何标准或自定义得类，你可以修改以下部分：

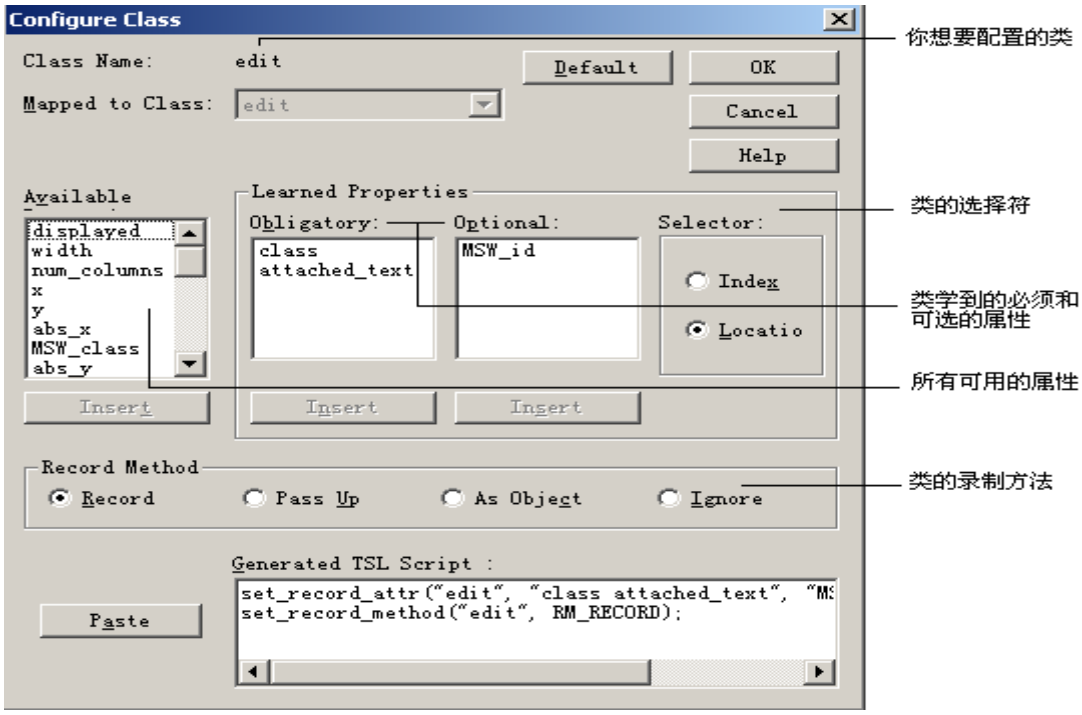
- 学到的属性
- 选择符
- 录制方法

操作方法：

2). 选择 ***Tools>GUI Map Configuration*** 打开 GUI map 配置对话框。类清单（Class List）包含所有的标准类以及你添加的自定义类。



3). 点选你想要配置的类，并点击 **Configure**。配置类对话框就会打开。



4). 你可以修改学到的属性，选择符或录制方法。

5). 修改完成后点击 **OK**。

6). 在 GUI Map Confguration 对话框中点击 **OK**。

9.4.1 配置学到的属性

对话框中 **Learned Properties** 这一栏允许你配置类的被录制和学习的属性。你在一个清单中选择一个属性，在你想要插入该属性的清单下点击 **Insert**。每个属性只能出现在一个清单上。

- Obligatory 清单上包含必须学习的属性。
- Optional 清单上的属性只有当 Obligatory 清单上的属性不足以提供唯一标识的时候才被使用。WR 从清单第一个属性开始，选择可以唯一标识对象的最小数量的属性（如果前三个属性足以唯一标识对象，就不用看第四个了）。
- Available 清单上的是不在前两个清单的上的其他的可用属性。

操作方法：

1). 先点选想要移动的属性，然后在目标清单下点击 **Insert**。例如：

- 想把 label 属性从 Obligatory 清单移动到 Optional 清单，在 Obligatory 清单点这个属性，然后在 Optional 清单下点击 **Insert**。
- 如果想把一个属性从清单上删除以使它不被学习，可以在 Obligatory 清单或 Optional 清单上点这个属性，然后在 Available 清单下点击 **Insert**。

2). 想要修改清单中属性的排列顺序（尤其在 Optional 清单中），点一个或多个属性，并点击清单下的 **Insert**。被点的属性就会排列到清单的底部。

3). 点击 **OK** 保存所做的修改。

注意：不是所有的属性都适用于全部的类。下面是属性和类的搭配清单：

属性(Property)	**类(Class)**
abs_x	所有类
abs_y	所有类
active	所有类
attached_text	combobox, edit, listbox, scrollbar
class	所有类
displayed	所有类

enabled	所有类
focused	所有类
handle	所有类
height	所有类
label	check_button, push_button, radio_button, static_text, window
maximizable	calendar, window
minimizable	calendar, window
MSW_class	所有类
MSW_id	所有类(除了窗体)
nchildren	所有类
obj_col_name	edit
owner	mdiclient, window
pb_name	check_button, combobox, edit, list, push_button, radio_button, scroll, window(object)
regexp_label	所有含义卷标的类
regexp_MSWclass	所有类
text	所有类
value	calendar, check_button, combobox, edit, listbox, radio_button, scrollbar, static_text
vb_name	所有类
virtual	list, push_button, radio_button, table, object (virtual objects only)
width	所有类
x	所有类
y	所有类

9.4.2 配置选择符

在必须和可选属性都不能唯一标识对象的情况下，WR 使用选择符：location 或 index。location 选择符根据对象在窗体中的位置（从上到下，从左到右）来选择。index 选择符根据开发人员分配给对象的唯一数字来选择。WR 默认使用 location 选择符。

9.4.3 配置录制方法

录制方法决定WR在录制属于某一类的对象时采用的操作方式。一共有四种可用方式：

- **Record** 指 WR 录制对某个 GUI 对象的所有操作。这是 WR 默认的对所有类的录制方式。（唯一的例外是 static 类，这个类的默认方式是 Pass Up。）
- **Pass Up** 指 WR 录制对包含某个对象的元素的一个操作。通常这个元素是个窗体，操作被录成 win_mouse_click。
- **As Object** 指 WR 录制对某个 GUI 对象的所有操作，并认为这个对象的类是“object”类（这个类的对象是那种不属于已经存在的任何一个类的对象）。
- **Ignore** 指 WR 忽略对某个类的所有操作。

9.5 创建永久的 GUI Map 配置

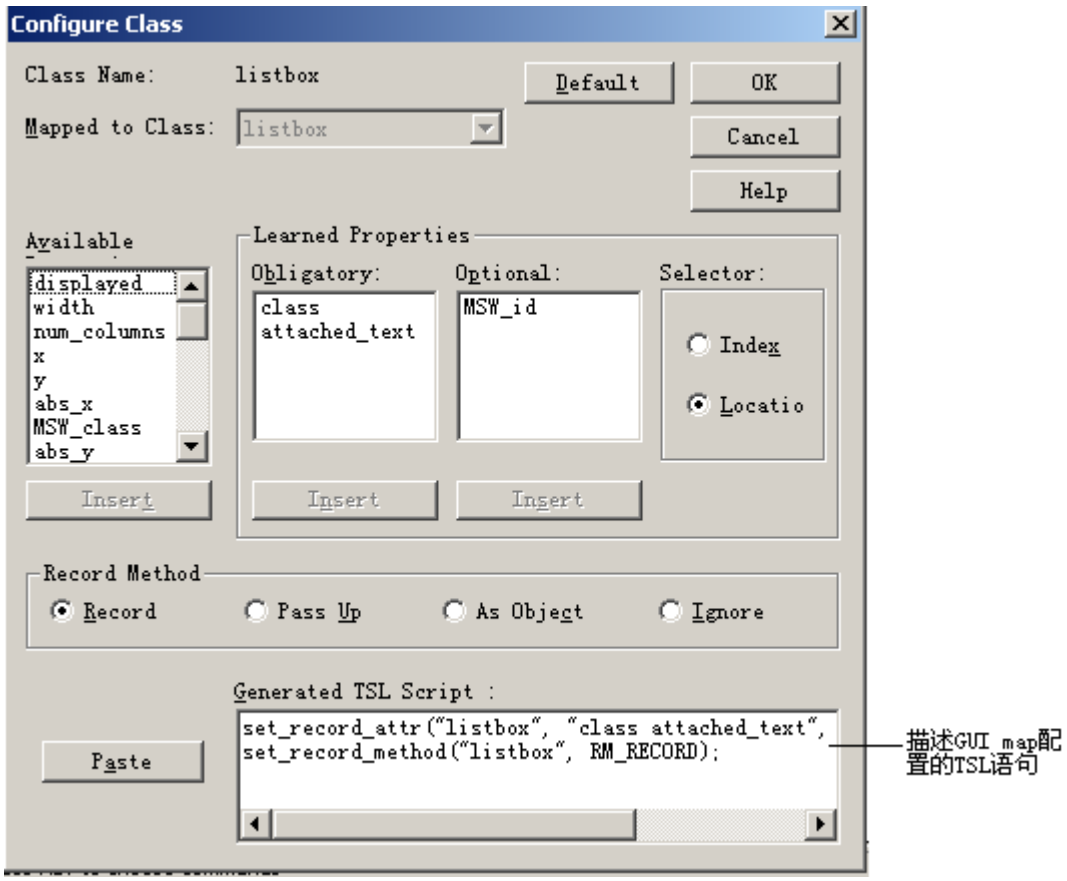
生成描述你所作配置的 TSL 语句，并把它们插入启动脚本中。这样一来，WR 就总能使用正确的 GUI map 配置。

为一个类创建永久 GUI map 配置的方法：

1). 选择 ***Tools>GUI Map Configuration*** 打开对话框。

2). 选择一个类，点击 **Configure**。

3). 配置完这个类后，你可以发现在对话框底部 WR 自动生成了描述这个配置的 TSL 语句。



4). 点击 **Paste** 可以把语句粘贴到启动脚本中。

9.6 删除自定义的类

你只能删除自定义的类。WR 使用的标准类不能删除。

操作方法：

1). 选择 ***Tools>GUI Map Configuration*** 打开对话框。

2). 在清单中点选你想删除的类。

3). 点击 **Delete**。

9.7 类属性

类属性是 WR 用来识别一类 GUI 对象的主要属性。WR 把 GUI 对象归到以下几类：

类（Class）	描述
calendar(日历)	一个标准的 calender 对象属于 CdateTimeCtrl 或 CmonthCalCtrl MSW_class
check_button	check box
edit	edit field(编辑区)
frame_mdiclient	使 WR 能把窗体当作 mdiclient 对象
list	list box, 它可以是常规 list 或 combo box
menu_item	菜单项目
mdiclient	mdiclient 对象
mic_if_win	使 WR 对这个窗体中的任何对象的录制和操作都遵从 mic_if library。这个和 WR 自定义操作有关。
object	包含任何不属于这个列表中某个类的对象，或称为“其他”
push_button	push button（按键）
radio_button	radio button（可选建）
scroll	scroll bar 或 slider（滚动条）
spin	旋转对象
static_text	只用来显示的文本，不是任何对象的一部分
status bar	窗体的状态栏
tab	tab 项目
toolbar	工具条对象
window	窗口，对话框或表格，包括 MDI windows

注意：如果不明白某个类对应的对象是什么，可以在脚本中通过语句查找软件中的对象。

9.8 所有属性

下面的表格包括所有 WR 在 Context Sensitive 测试中使用的属性。表格分成可移动属性、半移动属性和不可移动属性。

注意：XRunner 中创建的脚本不能在 WR 中使用。

可移动属性

属性	描述
abs_x	对象左上角对应屏幕左上角的 x 轴相对坐标
abs_y	对象左上角对应屏幕左上角的 y 轴相对坐标
attached_text	靠近对象的静态文本
class	请参考**类属性**
class_index	用来标识对象的序号，和窗体中其他同类对象的位置有关（只在 JAVA 插件加载的情况下）
count	菜单中包含的菜单项目的数量
displayed	一个布尔值，用来指出对象是否被显示：1 表示在屏幕上可见，0 表示不可见
enabled	一个布尔值，用来指出对象是否可用：1 表示可用，0 不可用
focused	一个布尔值，用来指出键盘输入是否直接到达对象：1 表示对象有键盘聚焦，0 没有
height	对象的高度（以象素为单位）
html_url	一个 URL（只在 WebTest 情况下）
label	对象上显示的文本
maximizable	一个布尔值，用来指出对象是否可最大化：1 表示可以，0 不可以
minimizable	一个布尔值，用来指出对象是否可最小化：1 表示可以，0 不可以
module_name	一个可执行文件的名称，这个文件创建了特定的窗体
nchildren	对象拥有的所有子元素的数量
NSTBTitle	浏览器中一个工具栏的标题（只在 WebTest 中）
NSTitle	浏览器的标题（只在 WebTest 中）
num_columns	表格对象（只在 Terminal Emulator 软件中）
num_rows	表格对象（只在 Terminal Emulator 软件中）
parent	对象的父对象的逻辑名

part_value	一组中的 radion button 或 check box 的名称（只在 WebTest 中）
position	菜单项目在菜单中从上到下的位置（第一个项目在位置 0）
submenu	布尔值，用来指出菜单项目是否有子菜单：1 表示有，0 没有
value	在不同类中含义不同： Radion 和 check button：1 表示已经被选，0 表示未被选 菜单项目：1 表示菜单已经被选，0 表示未被选 List 对象：指出被选择项目的文本字符串 Edit/Static 对象：指出文本区域的内容 滚动对象：指出滚动位置 所有其他对象：value 属性是一个空字符串(null string)
width	对象的宽度（以象素为单位）
x	对象左上角对应窗口原点的 x 轴坐标
y	对象左上角对应窗口原点的 y 轴坐标

半移动属性

属性	描述
handle	一个指向对象的运行指针：HWND handle
TOOLKIT_class	特定 toolkit 类的值。在视窗系统中这个属性的值和 MSW_class 的值相同。

不可移动的 Microsoft Windows 属性

属性	描述
active	布尔值，指出窗口是否被激活
MSW_class	Microsoft Windows 类
MSW_id	Microsoft Windows ID
obj_col_name	DataWindow 和列名称的串联。PowerBuilder 插件下，对于 edit field 对象而言是指明列的名称。
owner	窗体所属的软件的名称
pb_name	开发人员给 PowerBuilder 对象分配的字符串，只在 PowerBuilder 插件下有效。

regexp_label	使 WR 用可变卷标识别对象的字符串和常规表达式
regexp_MSW class	Microsoft Windows class 结合一个常规表达式，使 WR 能识别包含可变 MSW_class 的对象
sysmenu	布尔值，指出一个菜单项目是否是系统菜单的一部分
text	对象或窗体的可见文本
vb_name	开发人员给 Visual Basic 对象分配的字符串，只在 Visual Basic 插件下有效。

9.9 默认学习属性

下面是各个类默认学习的属性。

类	必须属性	可选属性	选择符
All buttons	class, label	MSW_id	location
list, edit, scroll, combobox	class, attached_text	MSW_id	location
frame_mdiclient	class, regexp_MSWclass, regexp_label	label, MSW_class	location
menu_item	class, label, sysmenu	position	location
object	class, regexp_MSWclass, label	attached_text, MSW_id, MSW_class	location
mdiclient	class, label	regexp_MSWclass, MSW_class	
static_text	class, MSW_id	label	location
window	class, regexp_MSWclass, label	attached_text, MSW_id, MSW_class	location

9.10 Visual Basic 对象的属性

所有 Visual Basic 对象的 label 和 vb_name 属性是必须属性。

9.11 PowerBuilder 对象的属性

下面是 PowerBuilder 对象的标准对象类和学习属性：

类	必须属性	可选属性	选择符
All buttons	class, pb_name	label, MSW_id	location
list, scroll, combobox	class, pb_name	attached_text, MSW_id	location
edit	class, pb_name, obj_col_name	attached_text, MSW_id	location
object	class, pb_name	attached_text, MSW_id, MSW_class	location
window	class, pb_name	label, MSW_id	location

10. 学习虚拟对象

10.1 关于学习虚拟对象

被测软件可能包含位图，而且看上去象是 GUI 对象。WR 录制对这些位图的操作时使用 win_mouse_click 语句。通过把位图定义成虚拟对象(virtual object)，你可以让 WR 像对待 GUI 对象那样处理它们。这样一来，测试脚本就比较容易阅读和理解。

例如：当你录制 Windows 自带的计算器软件时，由于 WR 不能把 calculator button 识别成 GUI 对象，它就把点击 button 录制成类似以下脚本：

```
set_window("Calculator");
win_mouse_click("Calculator",87,175);
win_mouse_click("Calculator",204,200);
win_mouse_click("Calculator",121,163);
win_mouse_click("Calculator",242,201);
```

这样的脚本很难理解。但是如果你把 calculator button 定义成虚拟对象并归如 button 类，WR 就会把脚本录制成类似以下：

```
set_window("Calculator");
button_press("seven");
button_press("plus");
button_press("four");
button_press("equal");
```

你可以创建虚拟 push buttons, radio buttons, check buttons, lists 或 tables，这些都依据位图在被测软件中的行为表现。如果这些类型都不适合，你可以创建一个属于综合类的虚拟对象。

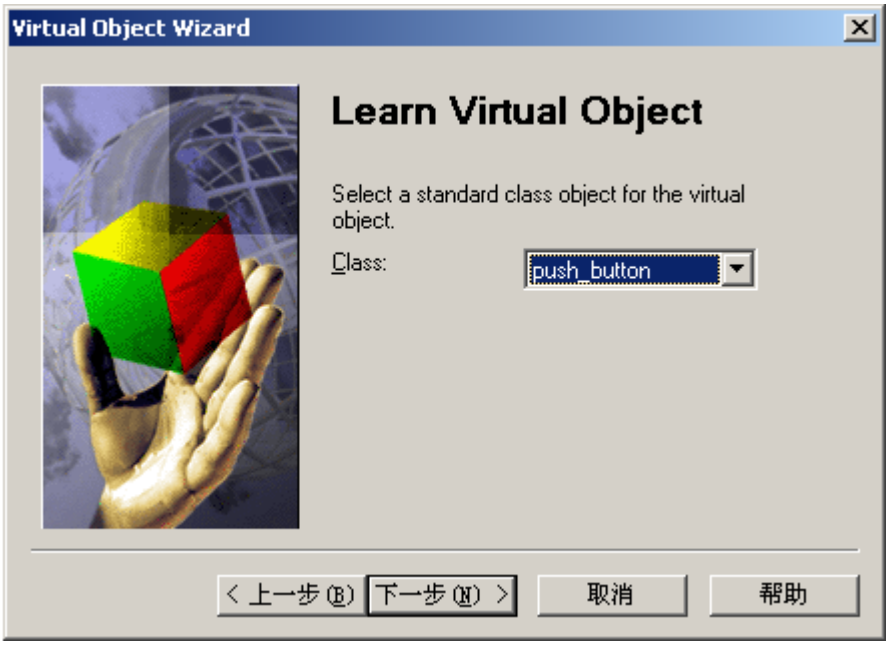
你使用 Virtual Object wizard（虚拟对象巫师）定义虚拟对象。巫师会提示你选择一个和对象相关的标准类。然后使用一个十字光标圈定对象的范围。最后你给对象选择一个逻辑名。WR 就把对象的逻辑名和物理描述添加到 GUI map 中。

10.2 定义一个虚拟对象

使用虚拟对象巫师的操作方法：

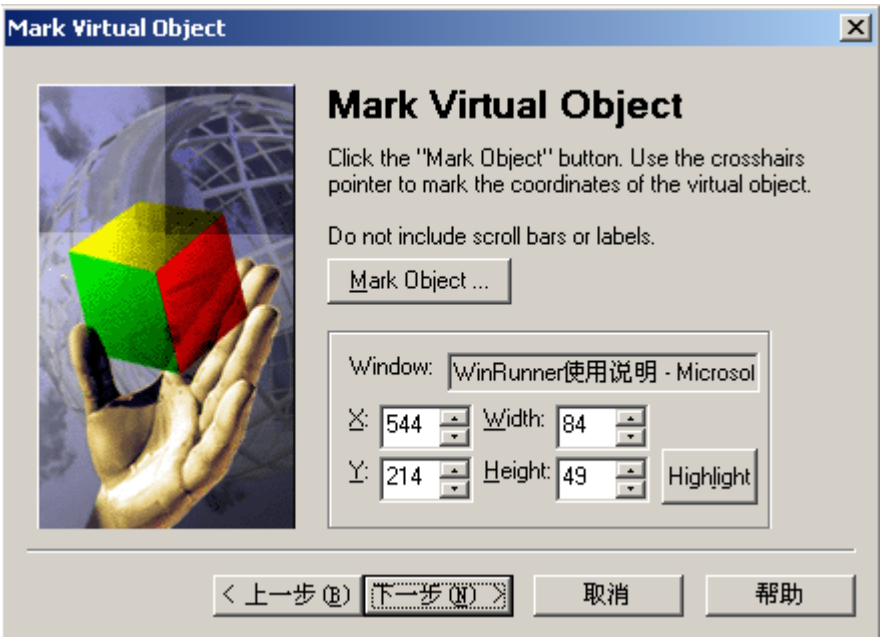
1). 选择 ***Tools>Virtual Object Wizard***。打开对话框之后点击 **Next**。

2). 在类清单中，选择一个适合新对象的类。如果你选择 list 类，就要选择行数。如果你选择 table 类，就要选择可见的行数和列数。然后点击 **Next**。



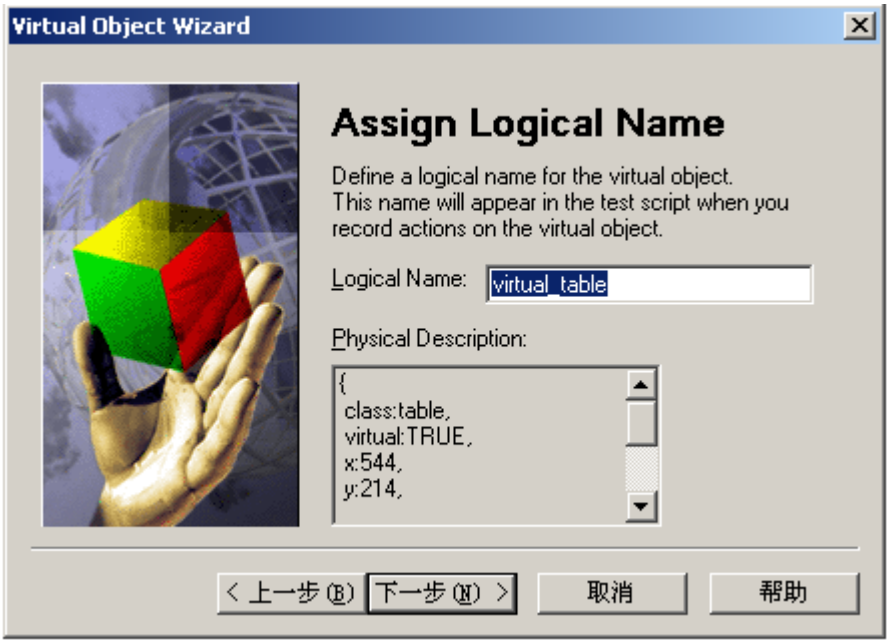
3). 点击 **Mark Object**。使用十字光标圈定虚拟对象的区域。你可以使用方向键修正区域的大小。圈定完成后按回车或点鼠标右键就会显示虚拟对象的坐标。

注意：虚拟对象的区域不要和 GUI 对象重叠。如果重叠的话，WR 可能会在录制或执行 GUI 对象时发生问题。

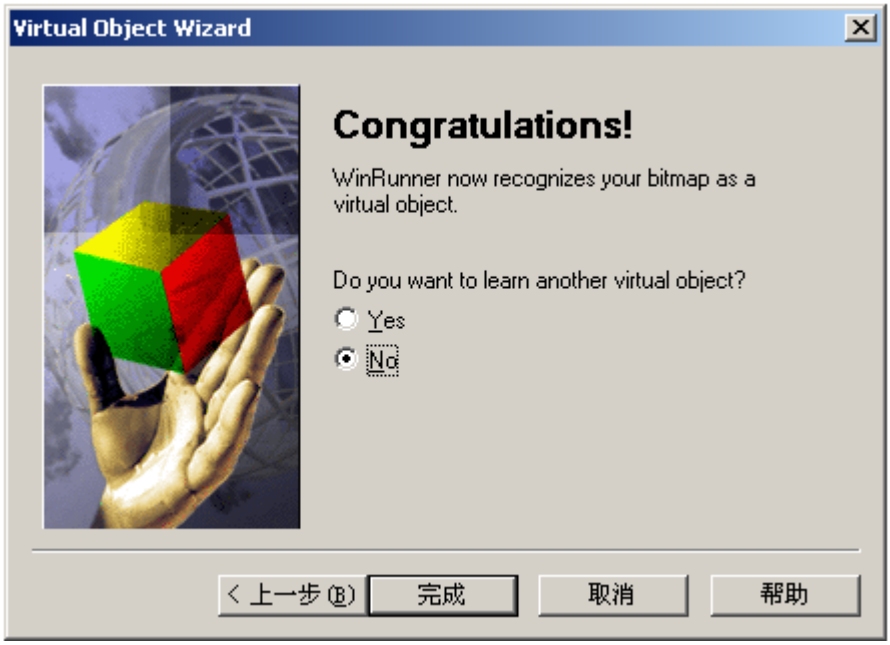


如果对象在屏幕上可见，你点击 **Highlight** 就可以查看它。点击 **Next**。

4). 给虚拟对象分配一个逻辑名。这个名字会出现在测试脚本中。如果对象包含 WR 可以读取的文本，巫师会建议使用这些文本作为逻辑名。否则，巫师会建议使用 virtual_table, virtual_list 之类。你也可以自己决定逻辑名。WR 会检查是否有相同的逻辑名被使用。完成后点击 **Next**。



5). 完成后，如果你想学习其他的虚拟对象就选择 **Yes**，点击 **Next**。要关闭巫师就点击 **Finish**。



10.3 理解虚拟对象的物理描述

当你创建虚拟对象后，WR 把它的物理描述添加到 GUI map 中。虚拟对象的物理描述中没有 label 属性。它有一个特殊的属性 virtual。这个属性用来识别虚拟对象，它的值总是 TRUE。

由于 WR 识别虚拟对象是根据它的尺寸和在窗体中的位置，因此 x, y 轴坐标、宽度和高度在虚拟对象物理描述中经常出现。例如一个 virtual_push_button 的物理描述包括：

{

class: push_button,

virtual: TRUE,

x: 77,

y: 111,

width: 50,

height: 30,

}

如果这些属性被修改或删除，WR 就不能识别这个虚拟对象了。如果对象被移动或改变尺寸，你必须创建一个新的虚拟对象。

11. 创建测试

11.1 关于创建测试

你可以用录制或编程的方式创建测试。通常从录制一个基础测试脚本开始。你对被测软件进行操作，TSL 语言记录下这些操作。你可以在脚本中修改、添加功能等。

有两种可用的录制模式：

- **Context Sensitive** 模式录制你对 GUI 对象的操作。
- **Analog** 模式用 X 轴和 Y 轴定位跟踪鼠标运行轨迹。

你可以在脚本中添加 GUI、位图、文本、数据库检查点和同步点。检查点让你比较软件当前版本和以前版本的区别。同步点用来处理测试运行中可能发生的时间控制和窗体位置等问题。

你也可以创建数据驱动测试。测试将使用内部数据库来驱动。

创建测试脚本的主要步骤：

1). 决定你想要测试的功能点。确定你想在脚本中使用的检查点和同步点。
2). 在测试属性对话框里把测试相关的信息文档化。
3). 选择录制模式（Context Sensitive 或 Analog）并录制。
4). 给测试分配一个名字并保存。

11.2 解决常见的环境感应录制问题

1). WR 不能录制正确的 TSL 语句

当 WR 不能正确识别对象时，就会录制成 obj_mouse 语句。可能的发生的原因和解决方法有以下几种：

可能的原因	可能的解决方法
支持对象的插件没有加载。	你必须安装并加载支持目标对象的插件。比如要识别 HTML 对象，就需要加载 WebTest 插件。
对象属于一个自定义类的对象。	如果自定义对象和某个标准对象相似，你可以把这个自定义的类映射到那个标准类。
	你可以添加一个自定义 GUI 对象类。
	你可以创建一个自定义录制和执行功能。如果你的对象改变了，你只需要修改这个功能就可以了。

2). WR 不能从 HTML 页面读取文本

可能的原因	可能的解决方法
WebTest 插件没有加载。	你必须安装并加载支持 Web 对象的插件。
WR 不能识别 HTML 框架或表格中的文本。	使用 ***Create>Get Text>From Selection(Web only)***命令从 HTML 页面中重新得到文本。对于 frame，WR 插入 web_frame_get_text 语句。对于其他对象，WR 插入 web_obj_get_text 语句。
	使用 ***Create>Get Text>WebText Checkpoint*** 命令检查是否有特定的字符串存在于 HTML 页面中。对于 frame，WR 插入 web_frame_text_exists 语句。对于其他对象，WR 插入 web_obj_text_exists 语句。

11.3 模拟录制

Analog 模式录制键盘输入、鼠标点击和鼠标运行轨迹。例如你从软件菜单选择 Open 命令，WR 录制的脚本类似于：

#mouse track

move_locator_track(1);

#left mouse button press

mtype("<T110><kLeft>-");

#mouse track

move_locator_track(2);

#left mouse button release

mtype("<T110><kLeft>+");

在 Context Sensitive 和 Analog 模式可以互相切换。

11.4 检查点

在脚本中可以插入以下四类检查点：

- GUI 检查点检验 GUI 对象信息。比如：你可以查看一个 button 是否可用或一个清单中哪个项目被选定了。详见**检查 GUI 对象**。
- 位图检查点做一个窗体或区域的截图，并把这张图片和以前版本进行比较。详见**检查位图**。
- 文本检查点读取 GUI 对象和位图中的文本，使你可以检验文本内容。详见**检查文本**。
- 数据库检查点检查一定数量的行和列组成的集合（这个集合由你在数据库中创建）的内容。详见**检查数据库**。

11.5 数据驱动测试

如果你想用多组数据测试相同的操作步骤，你可以创建数据驱动测试。测试会循环执行指定的次数，每次执行都由不同的数据驱动。为了使 WR 可以使用这些数据，你必须在测试脚本中建立和数据的联系。这就叫**测试参数化**。数据会被储存在一个数据表中。具体操作时你可以手工也可以用 DataDriver Wizard 来参数化测试以及把数据储存到表格中。详见**创建数据驱动测试**。

11.6 同步点

同步点让你解决测试和被测软件之间的时间占用问题。例如：如果你创建一个打开数据库软件的测试，你可以插入一个同步点让测试等待直到在数据库中的记录在屏幕上被加载。

对于模拟测试，你也可以使用同步点让 WR 把一个窗体移动到特定位置。

11.7 计划一个测试

在开始录制或编写代码之前仔细地计划测试。计划时有以下几个要点：

- 确认你将要测试的功能点。最好设计简短、测试单一功能点的测试。尽量不要设计繁长、一次涉及多个功能点的测试。
- 决定你要在测试中使用的检查点和同步点。
- 如果你计划使用录制的方式，就要决定哪个部分使用 Analog 模式，哪个部分使用 Context Sensitive 模式。
- 决定在脚本中添加哪些代码，如循环、数组或是自定义功能。

11.8 测试信息文档化

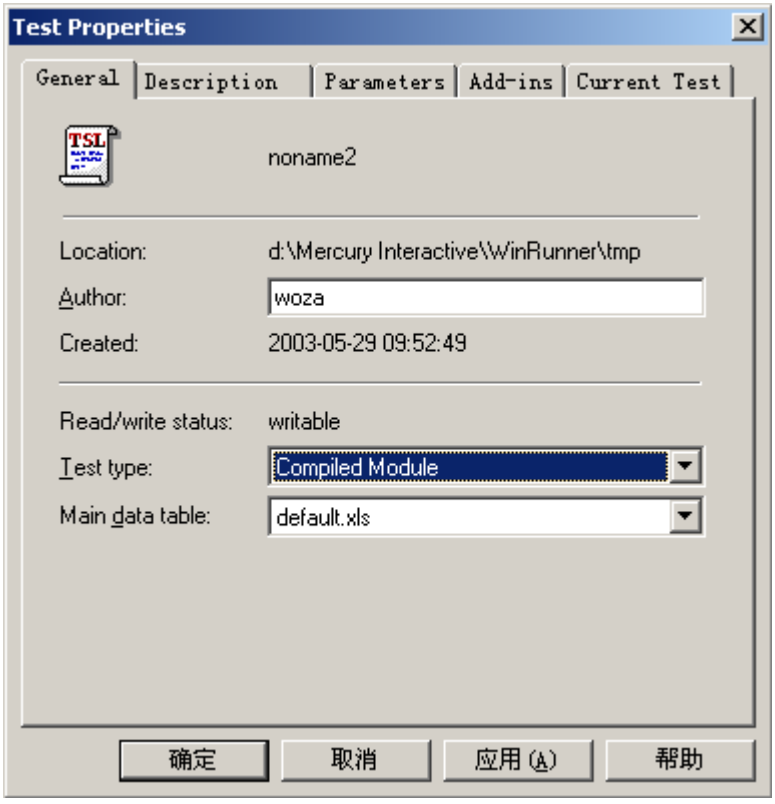
创建测试前，你可以把测试相关信息写在测试属性(Test Properties)对话框的综合和描述栏里。你可以输入测试创建者的名字，测试类型，对于测试的详细描述和功能描述文档。

你也可以用测试属性对话框来确定测试中使用的插件，给测试分配一个数据表，定义测试变量，把测试作为一个编译过的模块，复审测试的当前信息。

文档化测试信息的操作方法：

1). 选择 ***File>Test Properties*** 打开测试属性对话框。

2). 点击 **General** 栏。



栏目中显示如下信息：

选项	描述
TSL	显示测试的名称
Location	如果测试保存在 TestDirector 中，就显示在其中的位置。否则就显示在文件系统中的位置。

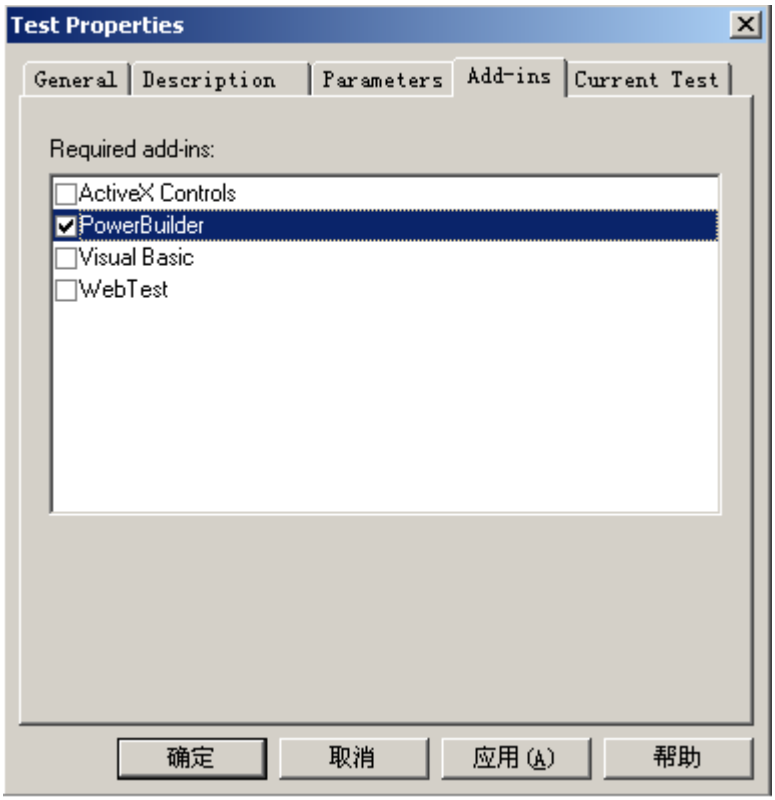
Author	测试作者的名字。
Created	显示测试创建的日期和时间。
Read/write status	指明测试是否只读。如果测试是只读的，那么测试属性中所有可编辑的区域都不可用。
Test type	指明测试类型是 Main Test 还是 Compiled Module。详见**创建一个编译过的模块**。
Main data table	显示测试主要数据表。

3). 点 **Description** 栏。并在里面添加必要的信息。

4). 点击 **OK** 保存这些信息并关闭对话框。

11.9 测试相关插件

在测试属性对话框中可以选择测试所需的插件。



你可以从 TestDirector 中加载所选插件。如果 WR 已经打开，但是未加载所需插件，TestDirector 会关闭 WR，加载插件并重新启动 WR。如插件未安装，TestDirector 就会显示错误信息“Cannot open test.”。

11.10 录制测试

录制时有以下要点：

- 开始录制前，关闭所有无关的程序。
- 使用 invoke_application 语句打开被测软件。
- 录制窗体中的对象前，点击窗体的标题栏来录一个 win_activate 语句。这是用来激活窗体的。
- 创建可以自我复原的测试。当测试结束时，测试环境必须恢复到测试前的状态。
- 录制时，你可以把 WR 最小化而使用 User 工具条（需要先把工具条和菜单分开）。这样你就可以全屏录制程序了。所有主要命令都在工具条上，当然你也可以自定义工具条的内容。
- 录制过程中，在一个窗体中移动时尽量使用鼠标，不要用 Tab 键。
- 在 Analog 模式下录制时，尽量使用热键插入检查点，不要用 WR 菜单或工具条。
- 在 Analog 模式下录制时，避免超前输入。如果你需要打开一个窗口，就等窗口完全打开后再继续操作。避免按住一个鼠标键不放，尤其是当这个当作会造成重复操作时（如点在滚动条上来移动屏幕显示的内容）。因为这样会造成一个无法被回放的时间感应（time-sensitive）操作。如果真的需要对鼠标键连续操作，可以多点击几下。
- WR 支持 RTL-style（从右到左）的窗体属性。包括菜单从右到左排列和输入、在左侧的滚动条、附加文本在 GUI 对象的右上角。WR 还支持输入时同时按下 CTRL 和 SHIFT（或 ALT 和 SHIFT）切换语言。
- WR 支持下拉和类菜单的工具条（比如 IE）。虽然类菜单的工具条看上去和菜单一模一样，但两者属于不同的类，而且 WR 录制起来也不同。工具条中被选定的项目会被录成 toolbar_select_item 语句，菜单中的则是 menu_select_item 语句。
- 如果一个测试文件夹或测试脚本文件在系统中被标记成只读，WR 就不能对它进行任何修改操作。

录制测试的操作方法：

1). 选择 ***Create>Record-Context Sensitive*** 或 ***Create>Record-Analog*** 或点击 **Record-Context Sensitive** 按钮。

2). 使用鼠标和键盘根据计划执行操作。使用工具条或从菜单中插入检查点和同步点。

3). 选择 ***Create>Stop Recording*** 或点击 **Stop** 停止录制。

11.11 用热键激活测试创建命令

以下是热键清单。你也可以自定义配置热键。

命令	默认热键	功能
RECORD	F2	开始录制。在录制过程中也用来在 Context Sensitive 和 Analog 模式之间切换。
CHECK GUI FOR SINGLE PROPERTY	右 Alt + F12	检查一个 GUI 对象的单个属性。
CHECK GUI FOR OBJECT/WINDOW	右 Ctrl + F12	为对象或窗体创建一个 GUI 检查点。
CHECK GUI FOR MULTIPLE OBJECTS	F12	打开创建 GUI 检查点对话框
CHECK BITMAP OF OBJECT/WINDOW	左 Ctrl + F12	捕捉一个对象或窗体的位图。
CHECK BITMAP OF SCREEN AREA	左 Alt + F12	捕捉一个区域的位图。
CHECK DATABASE(默认)	右 Ctrl + F9	创建对整个数据库的检查。
CHECK DATABASE(自定义)	右 Alt + F9	检查数据库中一定数量的行和列，以及特定内容。
SYNCHRONIZE OBJECT/WINDOW PROPERTY	右 Ctrl + F10	指示 WR 等待对象或窗体的一个属性出现一个期望值。
SYNCHRONIZE BITMAP OF OBJECT/WINDOW	左 Ctrl + F11	指示 WR 等待一个特定的对象或窗体位图出现。
SYNCHRONIZE BITMAP OF SCREEN AREA	左 Alt + F11	指示 WR 等待一个特定的区域位图出现。
GET TEXT FROM OBJECT/WINDOW	F11	捕捉对象或窗体上的文本。
GET TEXT FROM WINDOW AREA	右 Alt + F11	从一个特定区域捕捉文本，并在脚本中插入一条 obj_get_text 语句。

GET TEXT FROM SCREEN AREA	右 Ctrl + F11	从一个特定区域捕捉文本，并在脚本中插入一条 get_text 语句。
INSERT FUNCTION FOR OBJECT/WINDOW	F8	为 GUI 对象插入一个 TSL 功能。
INSERT FUNCTION FROM FUNCTION GENERATOR	F7	打开功能生成器对话框
STOP	左 Ctrl + F3	停止录制。
MOVE LOCATOR	左 Alt + F6	录制一条说明光标当前位置的 move_locator_abs 语句。

11.12 测试编程

你可以用编写代码的方法创建一个完整的脚本或是增强已录制的脚本。WR 自带一个虚拟编程工具－功能生成器（Function Generator），可以让你快速无错地在脚本中添加 TSL 功能。添加时，只需点一个对象或从功能清单上选择。详见**生成功能**。

你也可以像普通编程一样使用变量、控制语句、数组和自定义功能。你只需要把这些直接输入到脚本中就可以。

11.13 编辑测试

在修改测试脚本时，可以使用编辑菜单或工具条上相应的命令。下面是可用的命令：

编辑命令	描述
Undo	取消前一次的操作。
Redo	把前一次 Undo 掉的内容恢复。
Cut	剪切。
Copy	复制。
Paste	粘贴。
Delete	删除。
Selete All	全选。
Comment	把被选定的某行代码变成注释。
Uncomment	把注释变成代码。
Increase Indent	把被选定的文本向右边缩进一个 tab 的空间。
Decrease Indent	把被选定的文本向左边移动一个 tab 的空间。
Find	查找特定内容。
Find Next	查找下一个。
Find Previous	查找前一个。
Replace	替换。
Go To	把插入光标移动到脚本中特定一行。

12. 检查 GUI 对象

12.1 关于检查 GUI 对象

你可以使用 GUI 检查点来检验被测软件中的 GUI 对象。比如你可以检查一个对话框在何时打开，一个 button 是否可用等。你所要做得就是指向这个对象，选择你想要 WR 检查的属性。你可以检查 WR 建议的属性或自己指定属性。GUI 对象和被选定的属性保存在一个检查清单上。然后 WR 捕捉对象的当前属性值并保存起来作为期望值。这时一个 GUI 检查点就自动被插到脚本中。在脚本中这个检查点显示为 obj_check_gui 或 win_check_gui 语句。

执行测试时，检查点就把实际值和期望值比较。如果不符合就说明检查失败。检查结果可以在测试结果窗口看到。详见**分析测试结果**。

你检查的对象如果不在 GUI map 中就会被自动添加到临时 GUI map 文件里。

你可以在一个 edit 对象或 static 文本对象上用常规表达式创建 GUI 检查点。详见**使用常规表达式**。

你也可以用检查点检查表的属性和内容。详见**检查表的内容**。

12.2 检查单个属性的值

要检查对象的某个属性，使用 Check Property（检查属性）对话框添加下面的某个功能到脚本中：

button_check_info

edit_check_info

list_check_info

obj_check_info

scroll_check_info

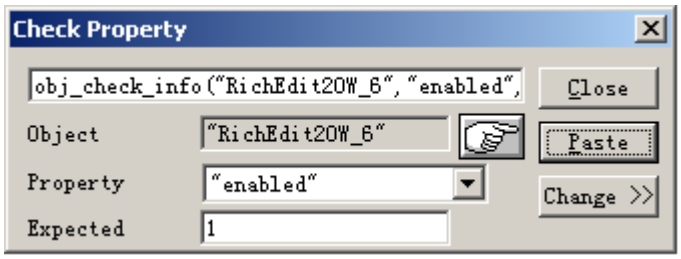
static_check_info

win_check_info

具体操作方法：

1). 选择 ***Create>GUI Checkpoint>For Single Property***。WR 窗口会最小化，鼠标指针也会变成手型。如果你使用 Analog 模式，使用右 Alt + F12 来避免鼠标移动。

2). 点击一个对象。然后 Check Property 对话框会打开并显示可以在对象上使用的默认功能。WR 自动给功能变量赋值。



3). 你可以修改想要检查的属性。

- 在 **Property** 清单里选择你想要检查的属性。期望值会在 **Expected** 框中被更新。
- 点击 **Change** 会打开功能生成对话框并显示功能清单。
- 想要选择不同的对象，点击手型图标，然后在新的对象上点击。
- 如果对象和所选的功能不一致，会提示当前功能不能用在这个对象上。点击 **OK** 关闭消息框，再点击 **Close** 关闭对话框。然后重复步骤 1 和 2。

4). 点击 **Paste** 把语句粘贴到脚本中。

12.3 检查单个对象

你可以创建 GUI 检查点检查被测软件中的单个对象。你可以检查对象的默认属性，也可以指定几个属性。

每个标准对象类都有一组默认检查属性。使用 gui_ver_set_default_checks 功能设置默认检查内容。

12.3.1 使用默认检查内容创建 GUI 检查点

你可以创建一个检查点来检查 WR 建议的几个属性。操作方法：

2). 选择 ***Create>GUI Checkpoint>For Object/Window***，或在工具条上点击 **GUI Checkpoint for Object/Window**。WR 窗口会最小化，鼠标指针会变成手型，屏幕上会出现帮助窗口。

3). 点一个对象。

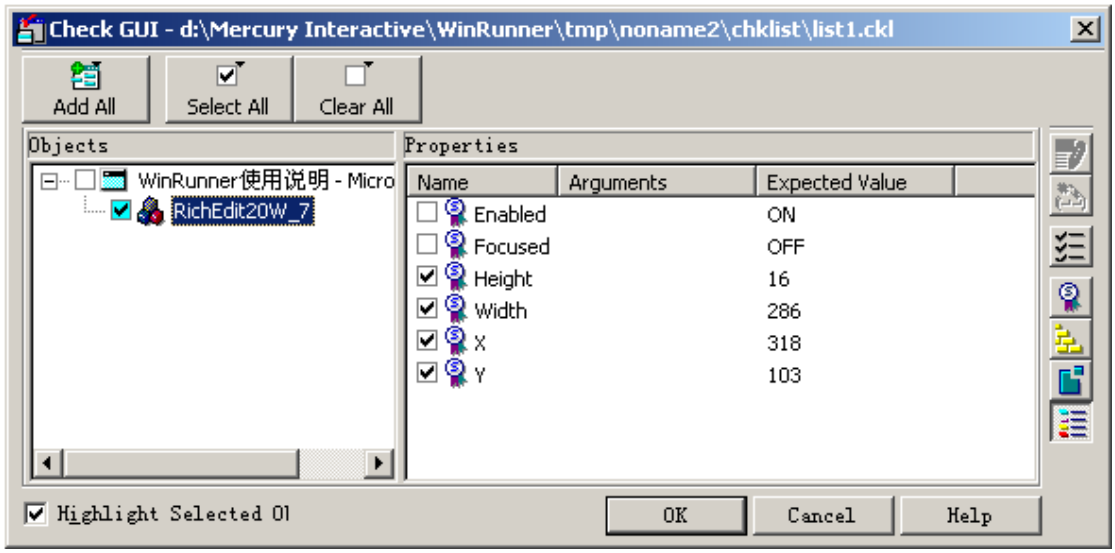
4). WR 把这个对象的当前属性值捕捉下来，并保存在期望结果文件夹里。然后 WR 窗口恢复，并在脚本中插入 obj_check_gui 语句。

12.3.2 创建检查点检查指定属性

你可以给检查点指定你想要检查的属性。操作方法：

1). 选择 ***Create>GUI Checkpoint>For Object/Window***，或在工具条上点击 **GUI Checkpoint for Object/Window**。WR 窗口会最小化，鼠标指针会变成手型，屏幕上会出现帮助窗口。

2). 双击对象或窗体。Check GUI 对话框会打开。



3). 在 **Objects** 框中点一个对象的名称。**Properties** 框中就会出现这个对象的所有属性。

4). 选择你想要检查的属性。

- 想要编辑一个属性的期望值，先选中这个属性，然后可以点 **Edit Expected Value** 按钮或是双击 **Expected Value** 列中的那个期望值。
- 想在属性中指定变量，先选中属性，然后点 **Specify Arguments** 按钮或双击 **Arguments** 列。注意：如果在 Arguments 栏中出现 3 个点，那你就必须给这个属性指定变量。如果已经有默认变量就不需要指定。
- 使用 Show Properties 按钮可以改变查看方式。

5). 点击 **OK** 关闭对话框。

12.4 检查一个窗体中的多个对象

你可以使用 GUI 检查点检查窗体中两个或更多对象。操作方法：

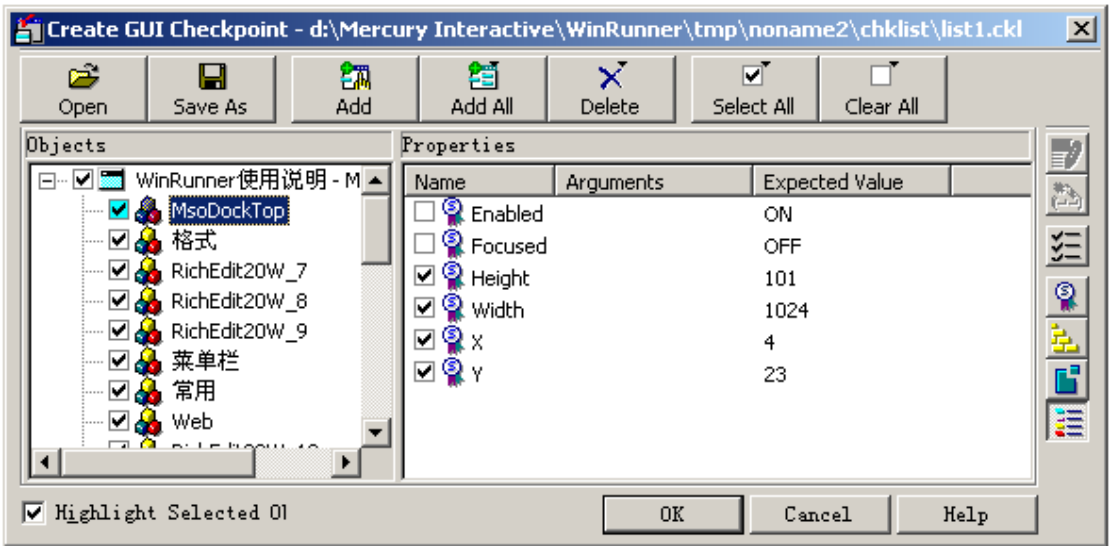
1). 选择 ***Create>GUI Checkpoint>For Multiple Objects*** 或在工具条中点 **GUI Checkpoint for Multiple Objects** 按钮。创建 GUI 检查点对话框就会打开。

2). 点击 **Add** 按钮。

3). 想添加一个对象，就点一下。如果你点在窗体标题栏或菜单栏上，帮助窗口会提示你检查窗体中所有的对象。

4). 完成一个对象后还可以继续选择下一个对象。但是你不可以把不同窗体中的对象插入到同一个检查点中。

5). 点击鼠标右键停止。创建 GUI 检查点对话框会打开。



6). **Objects** 栏中是检查点中的窗体和对象的名称。想要哪个对象被检查，就把名称前的勾打上。**Properties** 栏里是对象的全部属性。

- 想要编辑一个属性的期望值，先选中这个属性，然后可以点 **Edit Expected Value** 按钮或是双击 **Expected Value** 列中的那个期望值。
- 想在属性中指定变量，先选中属性，然后点 **Specify Arguments** 按钮或双击 **Arguments** 列。注意：如果在 Arguments 栏中出现 3 个点，那你就必须给这个属性指定变量。如果已经有默认变量就不需要指定。
- 使用 Show Properties 按钮可以改变查看方式。

7). 点击 **OK** 关闭并保存。WR 会在脚本中插入 win_check_gui 语句。

12.5 检查一个窗体中的所有对象

你可以用检查点查看窗体中的所有 GUI 对象的默认检查内容或指定检查内容。

12.5.1 用默认检查方法检查窗体中所有对象

1). 选择 ***Create>GUI Checkpoint>For Object/Window***，或在工具条上点击 **GUI Checkpoint for Object/Window**。WR 窗口会最小化，鼠标指针会变成手型，屏幕上会出现帮助窗口。

2). 点你想检查的窗口的标题栏或菜单栏。Add All 对话框会打开。

3). 选择对象或菜单或两者都选。你选择的内容就会被包括在检查清单中。

4). 点 **OK** 关闭对话框。

12.5.2 指定检查内容

1). 选择 ***Create>GUI Checkpoint>For Object/Window***，或在工具条上点击 **GUI Checkpoint for Object/Window**。WR 窗口会最小化，鼠标指针会变成手型，屏幕上会出现帮助窗口。

2). 双击你想检查的窗口的标题栏或菜单栏。WR 会生成一个包含窗体中所有对象的检查清单。等几分钟后，Check GUI 对话框会打开。

3). 指定需要的检查内容，点 **OK** 关闭对话框。

12.6 理解 GUI 检查点语句

只检查单个 GUI 对象的检查点语句是 obj_check_gui。检查多个对象的检查点语句是 win_check_gui。两个语句都和检查清单以及储存预期结果的文件关联。

- 检查清单列出需要被检查的对象和属性。当你创建 GUI 检查点时，你可以创建一个新的检查清单，也可以使用现存的。
- 预期结果文件包含了检查清单上每个对象的预期属性值。这些值在创建检查点时被捕捉下来，你可以手工更新这些值，也可以在更新模式下执行测试时更新。

obj_check_gui 语句句法如下：

obj_check_gui(object, checklist, expected results file, time);

object 是 GUI 对象的逻辑名。*checklist* 是检查清单的名字。*expected results file* 是储存期望结果文件的名字。*Time* 表明前一次输入和捕捉当前属性值之间的最大延迟（以秒计算）。这个时间间隔在测试执行时被添加到 timeout_msec 测试选项中。详见**从脚本中设置测试选项**。

例如：在航班预订程序中，如果你在登陆窗口点击 **OK**，语句就是

obj_check_gui(“OK”,”list1.ck1”,”gui1’,1);

win_check_gui 语句句法如下：

win_check_gui(window, checklist, expected results file, time);

window 是 GUI 窗体的逻辑名。*checklist* 是检查清单的名字。*expected results file* 是储存期望结果文件的名字。*Time* 表明前一次输入和捕捉当前属性值之间的最大延迟（以秒计算）。这个时间间隔在测试执行时被添加到 timeout_msec 测试选项中。详见**从脚本中设置测试选项**。

例如：在航班预订程序中，如果你点击登陆窗口的标题栏，语句就是

obj_check_gui(“Login”,”list1.ckl”,”gui1’,1);

注意：WR 把测试中的第一个检查清单命名为 list1.ckl，第一个预期结果文件为 gui1。

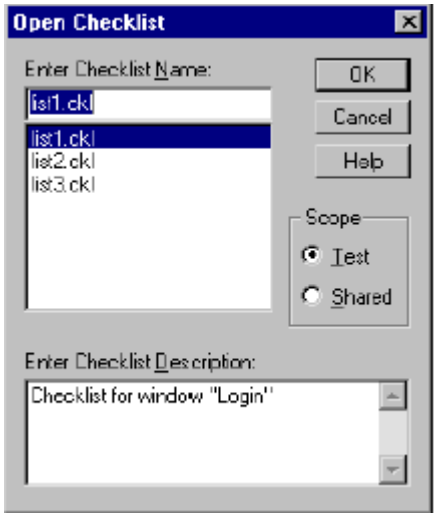
12.7 在 GUI 检查点中使用已存在的 GUI 检查清单

你可以在新创建的 GUI 检查点中使用已经存在的 GUI 检查清单。复用检查清单可以提供效率。如果你想在多个测试中共享检查清单，你必须把清单保存在共享文件夹中。使用已经存在的检查清单的操作方法：

1). 选择 ***Create>GUI Checkpoint>For Multiple Objects*** 或在工具条上点击 **GUI Checkpoint for Multiple Objects** 按钮。

2). 点击 **Open**。

3). 想查看共享文件夹中的检查清单，点 **Shared**。



4). 选择一个检查清单，点 OK。对话框会关闭，被选的清单会出现在 Create GUI Checkpoint 对话框中。

5). 打开被测软件中包含检查清单上对象的窗体。

6). 点击 **OK**。WR 就会捕捉当前属性值，并在脚本中插入 win_check_gui 语句。

12.8 修改 GUI 检查清单

你可以修改检查清单的内容。检查清单只包括需要检查的对象和属性。它并不包括属性的期望值。不过检查点的期望值也可以修改，详见**修改 GUI 检查点的期望结果**。

修改 GUI 清单时，你可以

- 把清单保存在共享文件夹中
- 编辑清单

12.8.1 把清单保存在共享文件夹中

默认情况下，检查清单被保存在当前测试的文件夹中。WR 保存共享清单的默认文件夹是 WR 安装文件夹/chklist。你可以用 General Options 对话框中的 Folders 栏中的 Shared Checklists 框改变保存文件夹。

保存到共享文件夹的操作方法：

1). 选择 ***Create>Edit GUI Checklist***。注意 GUI 检查清单有.ckl 后缀，数据库检查清单有.cdl 后缀。

2). 选一个 GUI 检查清单，然后点 **OK**。

3). 点 **Save As** 保存检查清单。

4). 在 **Scope** 框中点 **Shared**。输入文件名，点 **OK**。

5). 点 **OK** 关闭 Edit GUI Checklist 对话框。

12.8.2 编辑 GUI 检查清单

通过编辑，你可以

- 改变窗体中需要检查的对象
- 改变对象的检查属性
- 改变属性中的检查变量
- 给一个新的属性指定变量

注意，开始编辑前必须先加载检查清单。

选择 ***Create>Edit GUI Checklist***。其他操作和前几个章节的相关操作完全一样。编辑完成后

- 点击 **OK**，检查清单就会被保存，WR 会提示你是否覆盖原来的文件。
- 点 **Save As** 用其他名字保存。

注意：使用 Verify 模式运行前，先用 Update 模式更新期望结果来吻合你所做的修改。

12.9 理解 GUI 检查点对话框

创建和维护 GUI 检查点一共涉及三个对话框：Check GUI 对话框，Create GUI Checkpoint 对话框和 Edit GUI Checklist 对话框。

12.9.1 对话框中可能出现的信息

信息	含义	对话框	位置
Complex Value	属性的期望值或实际值太复杂以至于无法在列中显示。这个信息经常出现在有内容检查的表中。	Check GUI, Create GUI Checkpoint, GUI checkpoint Results	Properties pane, Expected Value column or Actual Value column
N/A	属性的期望值未被捕捉到：需要先指定变量，或需要把属性添加到检查点中才能捕捉。	Check GUI, Create GUI Checkpoint, GUI checkpoint Results	Properties pane, Expected Value column
Cannot Capture	属性的期望值或实际值不能被捕捉。	Check GUI, Create GUI Checkpoint, GUI checkpoint Results	Properties pane, Expected Value column or Actual Value column
No properties are available for this object	对象没有任何属性。	Check GUI, Create GUI Checkpoint, Edit GUI Checklist	Properties pane
No properties were captured for this object	当检查点创建时，这个对象未被指定需要检查的属性。	GUI Checkpoint Results	Properties pane

注意：关于 GUI Checkpoing Results 对话框详见**分析测试结果**。

12.10 属性检查和默认检查

以下是各个类可以被检查的属性：

1). Calendar Class

- **Enabled**：检查日历是否能被选择。
- **Focused**：检查键盘是否可以直接对日历进行操作。
- **Height**：检查日历的高度（象素为单位）。
- **Selection**：日历中被选定的日期（默认检查）。
- **Width**：检查日历的宽度（象素为单位）。
- **X**：检查日历左上角相对于窗口的 x 轴坐标。
- **Y**：检查日历左上角相对于窗口的 y 轴坐标。

2). Check_button Class 和 Radio_button Class

- **Enabled**：检查 button 是否可用。
- **Focused**：检查键盘是否可以直接对 button 进行操作。
- **Height**：检查 button 的高度（象素为单位）。
- **Label**：检查 button 的卷标。
- **State**：检查 button 的状态（开或关），（默认检查）。
- **Width**：检查 button 的宽度（象素为单位）。
- **X**：检查 button 左上角相对于窗口的 x 轴坐标。
- **Y**：检查 button 左上角相对于窗口的 y 轴坐标。

3). Edit Class 和 Static Text Class

检查以下五个属性时（Compare, DateFormat, Range, RegularExpression, TimeFormat），你必须指定变量。

- **Compare**：检查对象的内容（默认检查）。你可以指定以下变量：

a). 把内容当作文本做情况相关（case-sensitive）检查（默认设置）。

b). 把内容当作文本做情况无关（case-insensitive）检查。

c). 在内容当作数字检查。

- **DateFormat**：检查对象使用特定日期格式的内容。你必须指定变量（日期格式）。WR 支持很多种日期格式。
- **Enabled**：检查对象是否能被选择。
- **Focused**：检查键盘是否可以直接对对象进行操作。
- **Height**：检查对象的高度（象素为单位）。
- **Range**：检查对象在特定范围内的内容。你必须指定变量（范围上限和下限）。

- **RegularExpression：**检查对象中符合常规表达式的字符串。你必须指定变量（字符串）。你不需要在表达式前加感叹号。详见**使用常规表达式**。
- **TimeFormat：**检查对象使用特定时间格式的内容。你必须指定变量（时间格式，如 hh.mm.ss 或 hh:mm:ss 或 hh:mm:ssAM/PM）。
- **Width：**检查对象的宽度（象素为单位）。
- **X：**检查对象左上角相对于窗口的 x 轴坐标。
- **Y：**检查对象左上角相对于窗口的 y 轴坐标。

4). List Class

- **Content：**检查整个 list 的内容。
- **Enabled：**检查 list 是否能被选择。
- **Focused：**检查键盘是否可以直接对 list 进行操作。
- **Height：**检查 list 的高度（象素为单位）。
- **ItemsCount:** 检查 list 中项目的数量。
- **Selection：**检查当前 list 中被选定的项目（默认检查）。
- **Width：**检查 list 的宽度（象素为单位）。
- **X：**检查 list 左上角相对于窗口的 x 轴坐标。
- **Y：**检查 list 左上角相对于窗口的 y 轴坐标。

5). Menu_item Class

- **HasSubMenu：**检查一个菜单项目是否有子菜单。
- **ItemEnabled：**检查菜单是否可用（默认检查）。
- **ItemPosition：**检查菜单中各个项目的位置。
- **SubMenusCount：**计算子菜单中项目的数量。

6). Object Class

- **Enabled：**检查对象是否能被选择。
- **Focused：**检查键盘是否可以直接对对象进行操作。
- **Height：**检查对象的高度（象素为单位）(默认检查)。
- **Width：**检查对象的宽度（象素为单位）（默认检查）。
- **X：**检查对象左上角相对于窗口的 x 轴坐标（默认检查）。
- **Y：**检查对象左上角相对于窗口的 y 轴坐标（默认检查）。

7). Push_button Class

- **Enabled：**检查 button 是否能被选择（默认检查）。
- **Focused：**检查键盘是否可以直接对 button 进行操作。

- **Height**：检查 button 的高度（象素为单位）。
- **Label**：检查 button 的卷标。
- **Width**：检查 button 的宽度（象素为单位）。
- **X**：检查 button 左上角相对于窗口的 x 轴坐标。
- **Y**：检查 button 左上角相对于窗口的 y 轴坐标。

8). Scroll Class

- **Enabled**：检查滚动条是否能被选择。
- **Focused**：检查键盘是否可以直接对滚动条进行操作。
- **Height**：检查滚动条的高度（象素为单位）。
- **Position**：检查滚动条上滑块的当前位置（默认检查）。
- **Width**：检查滚动条的宽度（象素为单位）。
- **X**：检查滚动条左上角相对于窗口的 x 轴坐标。
- **Y**：检查滚动条左上角相对于窗口的 y 轴坐标。

9). Window Class

- **CountObjects**：计算窗体中 GUI 对象的数量（默认检查）。
- **Enabled**：检查窗体是否能被选择。
- **Focused**：检查键盘是否可以直接对窗体进行操作。
- **Height**：检查窗体的高度（象素为单位）。
- **Label**：检查窗体的卷标。
- **Maximizable**：检查窗体是否可以被最大化。
- **Maximized**：检查窗体是否已经被最大化。
- **Minimizable**：检查窗体是否可以被最小化。
- **Minimized**：检查窗体是否已经被最小化。
- **Width**：检查窗体的宽度（象素为单位）。
- **X**：检查窗体左上角 x 轴坐标。
- **Y**：检查窗体左上角 y 轴坐标。

12.11 为属性检查指定变量

为 edit class 或 static_text class 指定变量的方法：

1). 确认 GUI Checkpoint 对话框包含你想要指定变量的对象属性。

2). 在 **Objects** 栏中选择对象。

3). 在 **Properties** 栏选择属性。

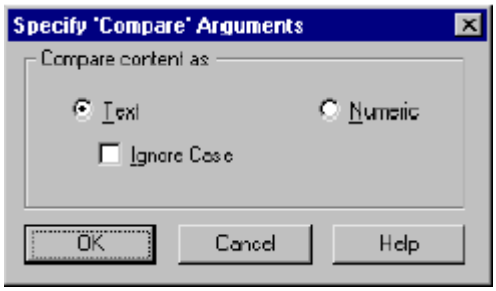
4). 点击 **Specify Aguments** 或双击默认变量或点右键选择 **Specify Aguments**。

5). 在对话框中指定变量。

6). 点 **OK** 关闭对话框。

12.11.1 比对属性（Compare Property）检查

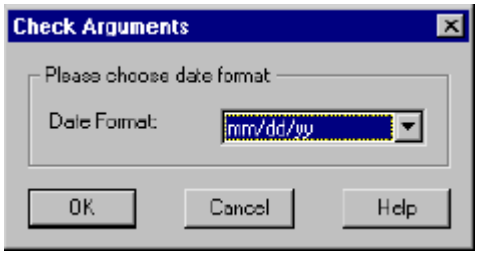
打开 **Specify 'Compare' Arguments** 对话框。



- 点击 **Text** 把内容作为文本来检查（默认设置）。
- 要在检查时忽略情况，选择 **Ignore Case**。
- 点击 **Numeric** 把内容当作一个数字来检查。

12.11.2 日期格式属性检查

想要指定日期格式，在 Check Arguments 对话框中的下拉菜单中选择。



可选的日期格式有以下几种：

mm/dd/yy (01/22/99)

dd/mm/yy (22/01/99)

dd/mm/yyyy (22/01/1999)

yy/dd/mm

dd.mm.yy

dd.mm.yyyy

dd-mm-yy

dd-mm-yyyy

Day, Month dd, yyyy　Friday(or Fri), September(or Sept) 24, 1999

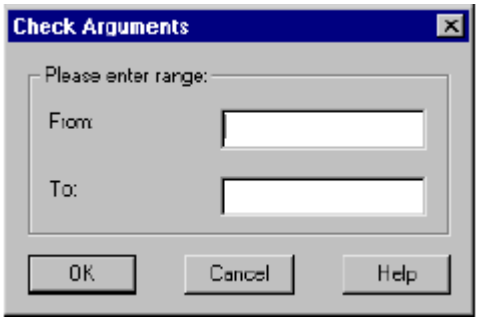
dd Month yyyy　24 September 1999

Day dd Month yyyy　Friday(or Fri) 24 September(or Sept) 1999

注意：日子或月份中出现 0（如 9 月就是 09），这个 0 在格式检查时不是必要的。

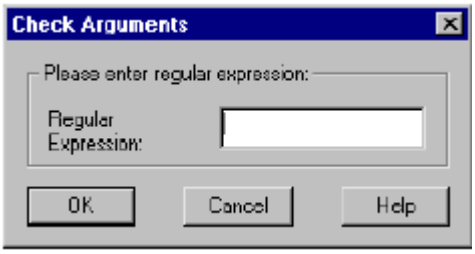
12.11.3　范围属性检查

注意：所有数字前的货币符号在检查时都被去掉。

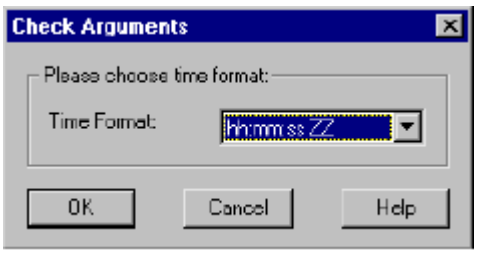


12.11.4　常规表达式属性检查

在对话框中输入表达式时不需要在前面加感叹号。双斜杠“\\”被认为是单斜杠“\”。



12.11.5　时间格式属性检查



WR 支持以下几种时间格式：

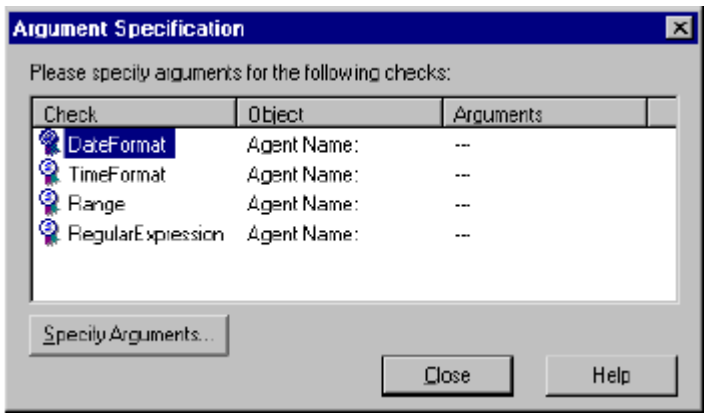
hh.mm.ss　　11.22.33

hh:mm:ss　　11:22:33

hh:mm:ss ZZ　　11:22:33 AM

12.11.6　关闭 GUI 检查点对话框

如果你选择了需要指定变量的检查属性而未给它们指定变量，当你点 **OK** 关闭对话框时，会提示你指定变量。



12.12 编辑属性期望值

当你创建一个 GUI 检查点时，WR 把对象属性的当前值捕捉下来作为期望值。在你执行测试时，WR 再次捕捉这些属性的当前值，并和期望值做比较。如果你在 GUI 检查点创建完成后想改变某个属性的值，你只需要在 Check GUI 对话框或 Create GUI Checkpoint 对话框中编辑这个属性的期望值。

注意：你不能在 Edit GUI Checklist 对话框中编辑期望值，因为你使用这个对话框时，WR 并不捕捉当前值，所以这个对话框不显示可被编辑的期望值。如果你想修改的期望值已经是 GUI 检查点的一部分，那么你必须修改 GUI 检查点的期望结果。详见**修改 GUI 检查点的期望结果**。

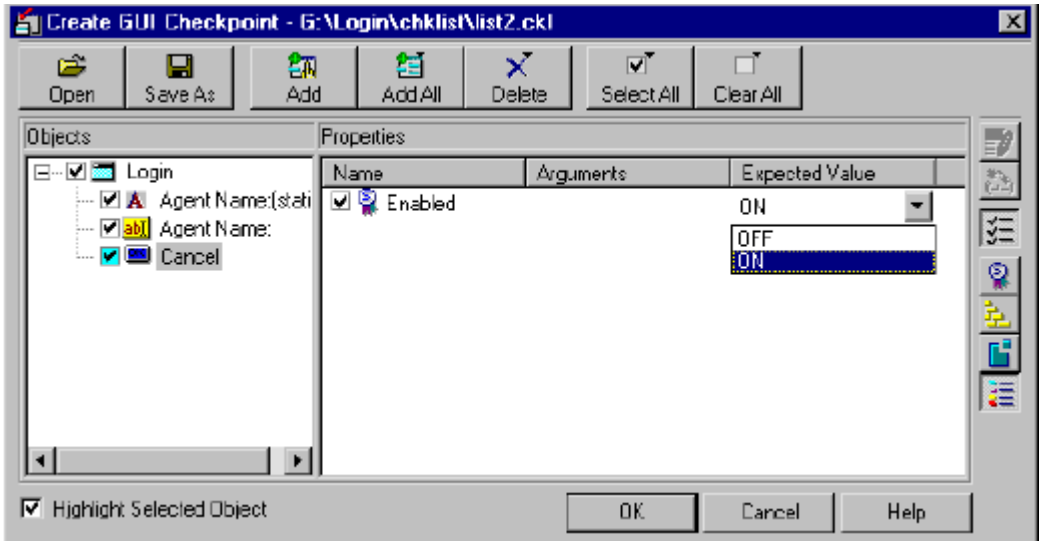
编辑对象期望值的操作方法：

1). 选择 ***Create>GUI Checkpoint>For Multiple Objects*** 打开对话框。点 **Open** 打开检查清单。注意：Check GUI 对话框只在你创建新 GUI 检查点时才打开。

2). 选定对象。

3). 选定需要编辑的属性。

4). 编辑属性



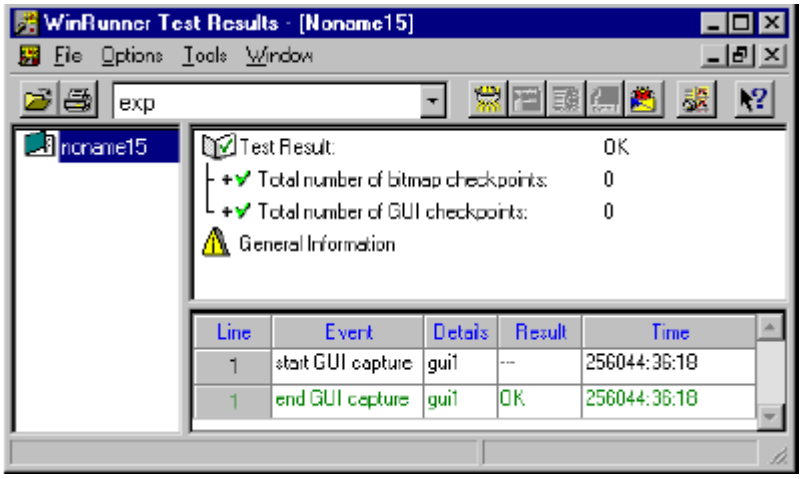
5). 点击 **OK** 改变对话框。

12.13 修改 GUI 检查点的期望结果

你可以通过改变检查点中属性期望值的来修改检查点的期望结果。可以在运行脚本前后做修改。

操作方法：

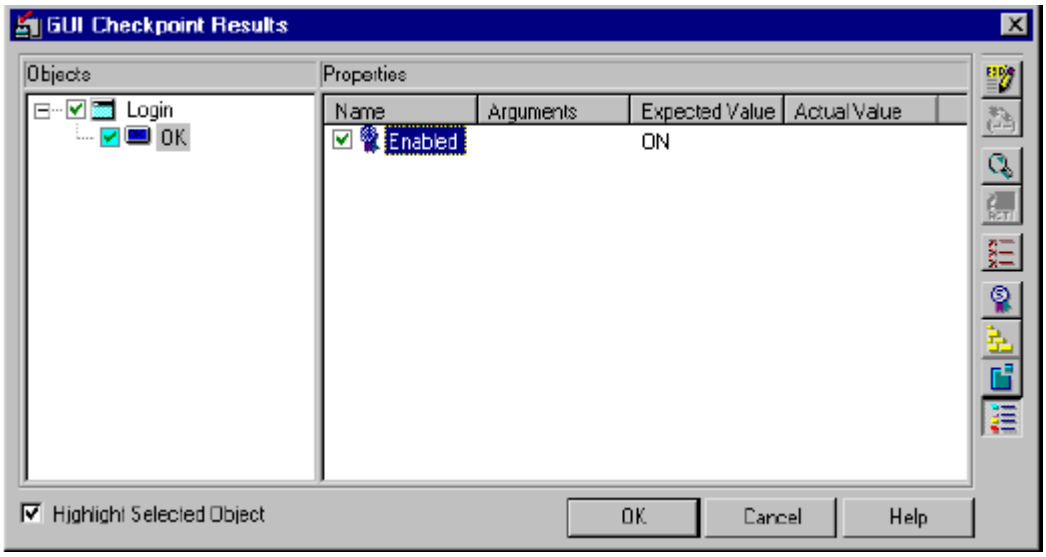
1). 选择 ***Tools>Test Results*** 或点击 **Test Results**。WR 测试结果窗口打开。



2). 选择你的期望结果文件夹（默认是 exp）。

3). 在 test log 中的 Event 列里，找到“end GUI capture”。Line 列里是测试脚本中的行数。

4). 双击“end GUI capture”,或点这个记录再点 **Display**。GUI 检查点结果对话框打开。



5). 选择需要修改的属性。改完后点 **OK**。注意：你可以在测试结束后用实际值替换期望值来作为下一次测试的期望值。