LoadRunner 测试流程

测试部　　刘非

1. LoadRunner 概要介绍

LoadRunner® 是一种预测系统行为和性能的工业标准级负载测试工具。通过以模拟上
千万用户实施并发负载及实时性能监测的方式来确认和查找问题，LoadRunner 能够对整个
企业架构进行测试。通过使用 LoadRunner ，企业能最大限度地缩短测试时间，优化性能和
加速应用系统的发布周期。

1.1. 为什么应进行自动化性能测试？

自动性能测试是一项规范，它利用有关产品、人员和过程的信息来减少应用程
序、升级程序或修补程序部署中的风险。自动性能测试的核心原理是通过将生产
时的工作量应用于预部署系统来衡量系统性能和最终用户体验。构造严密的性能
测试可回答如下问题：

- 应用程序是否能够很快地响应用户的要求？
- 应用程序是否能处理预期的用户负载并具有盈余能力？
- 应用程序是否能处理业务所需的事务数量？
- 在预期和非预期的用户负载下，应用程序是否稳定？
- 是否能确保用户在真正使用软件时获得积极的体验？

通过回答以上问题，自动性能测试可以量化更改业务指标所产生的影响。进而可
以说明部署的风险。有效的自动性能测试过程将有助于您做出更明智的发行决
策，并防止系统出现故障和解决可用性问题。

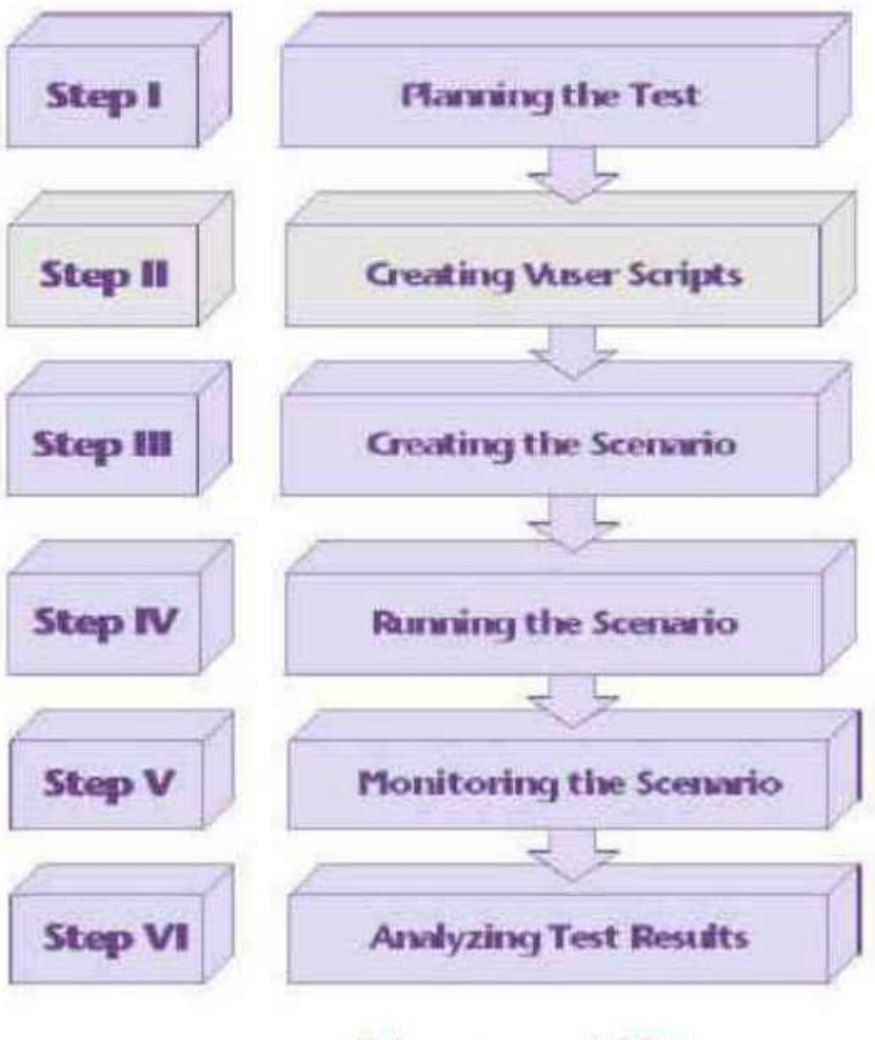
1.2. LoadRunner 主要组件有哪些？

LoadRunner 主要包含下列三种组件：

- Vugen 虚拟用户生成器用于捕获最终用户业务流程和创建自动性能测试脚本 （也称为虚
拟用户脚本）。
- Controller 用于组织、驱动、管理和监控负载测试。
- Analysis 有助于您查看、分析和比较性能结果。

1.3. 负载测试流程是什么？

测试分析->创建脚本->业务场景分析->创建场景->运行场景->监控场景->分析场景



2. LoadRunner 的安装

2.1. 需要的安装软件：LR8.1（虚拟光驱版）

里面包含的文件列表如下截图：

2.2. 安装步骤（安装前确保本机已装有虚拟光驱）

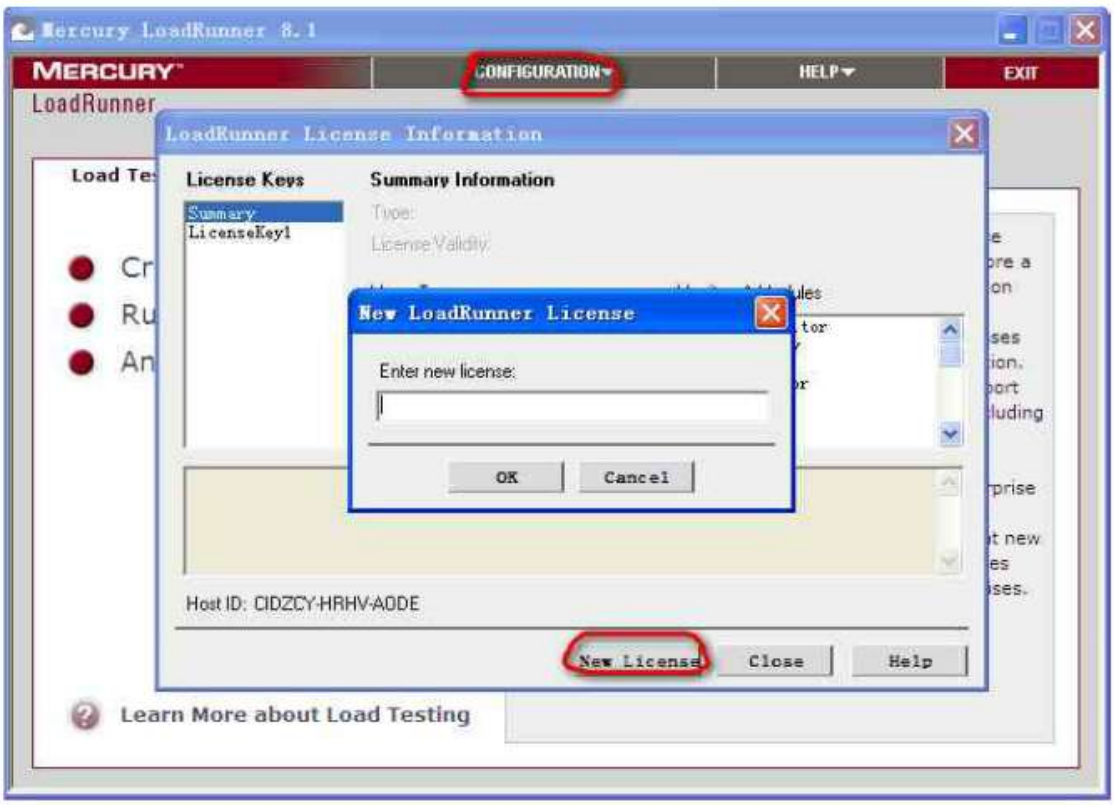
步骤一（非必须）：如果本机以前装过 LoadRunner，先运行 lr 删除注册表，确保原来 lr 在注册表中的信息能够全部删除

步骤二（必须）：通过虚拟光驱装载镜像文件 LR_81_WIN32_CD1.iso（打开后如下图），运行 setup.exe 进行安装（默认选项即可），该镜像文件是 loadRunner 软件的安装程序，拥有了 LoadRunner 大部分功能。运行完后重启电脑。

步骤三（必须）：通过虚拟光驱装载镜像文件 LR_81_WIN32_CD1.iso（打开后如下图），该镜像文件是 LoadRunner 的插件，可以解决通过 webservice 方式录制脚本时接口参数不识别的问题。打开 LR_81_FP4 文件夹，运行 setup.exe 进行安装。

步骤四（非必须）：通过虚拟光驱装载镜像文件 LR81FP4CHS.iso，该镜像文件是中文插件，安装后 LoadRunner 操作界面变成中文。

步骤五（必须）：此时 LoadRunner 已安装完毕，但是还需要注册 License 来支持同时运行大量的虚拟用户。首先按照 LoadRunner 8.1 破解与无法安装 .txt 的说明，将 lm70.dll 和 mlr5lprg.dll 替换本地目录里的同名文件；然后运行 LoadRunner，点选下拉菜单 configuration->LoadRunner License，点击 New License，从 txt 中选择 License 注册：



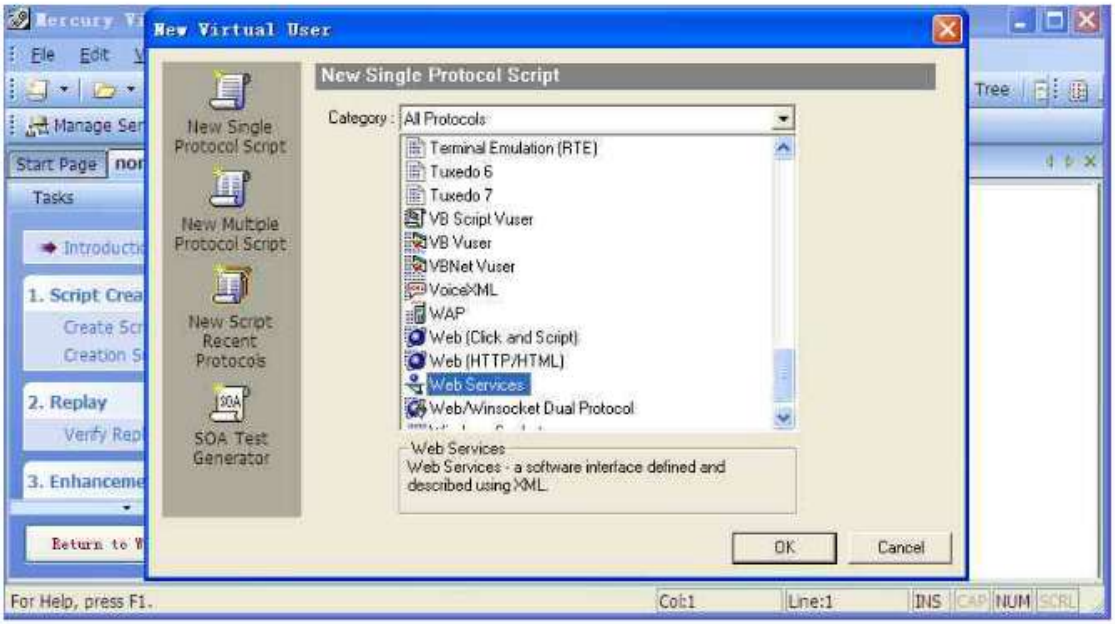
3. 根据负载测试流程进行性能测试(xxxx 组为例)

3.1 测试分析

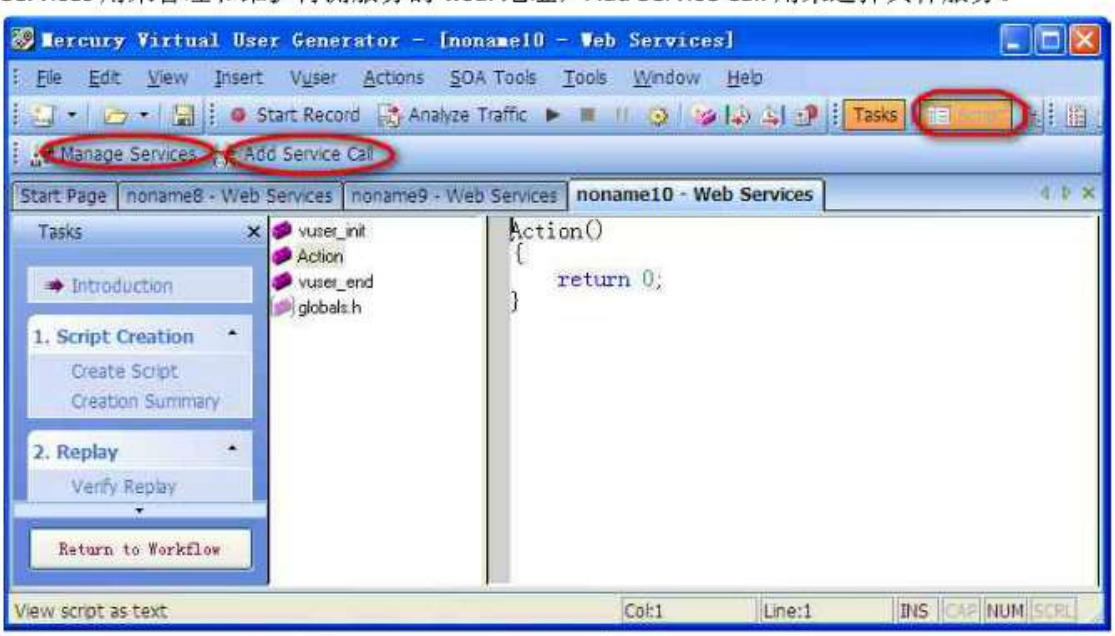
首先根据待测试应用的类型选择脚本录制的方式， 鉴于 xxxx 组应用最终都会发布成 webservice 提供给外界访问，我们选择 webservice 的方式录制脚本。这种方式直接访问服务，能够更准确的衡量服务的性能。

3.2 创建脚本

打开 Virtual User Generator,点选 File->New,弹出协议对话框，这里面不同的协议表示 lr 通过不同的方式和待测试应用交互，在这里我们选择 Web Services,点击 OK

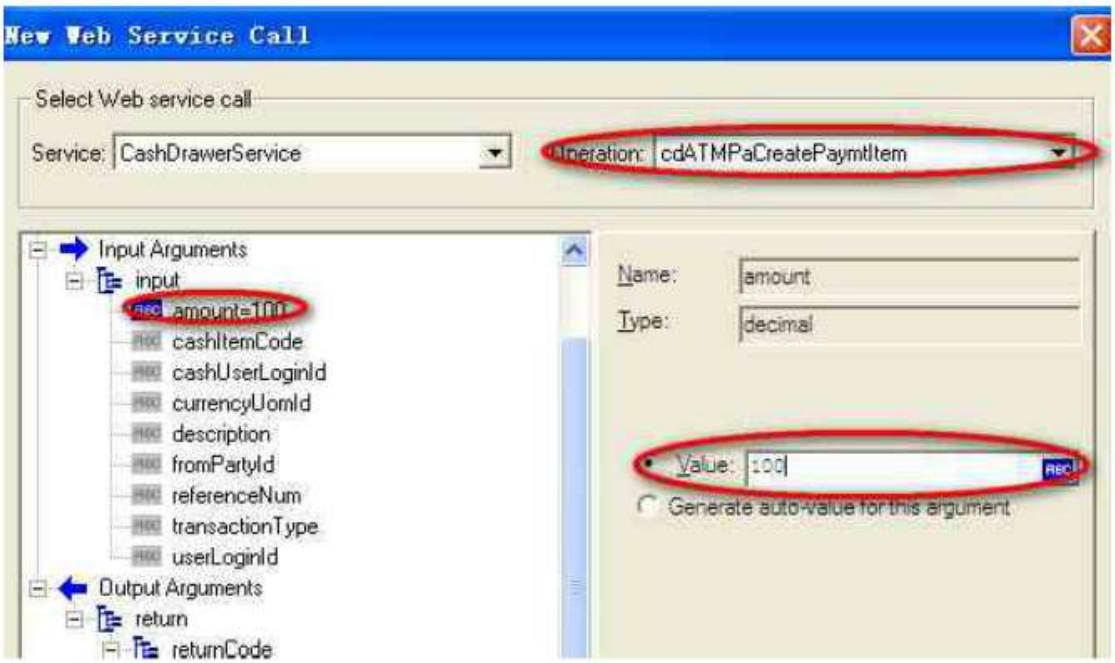


点击 OK 后如下图所示，如果显示和图示不一致，点击 Script 切换查看方式，其中 Manage Services 用来管理和维护待测服务的 wsdl 地址，Add Service Call 用来选择具体服务。

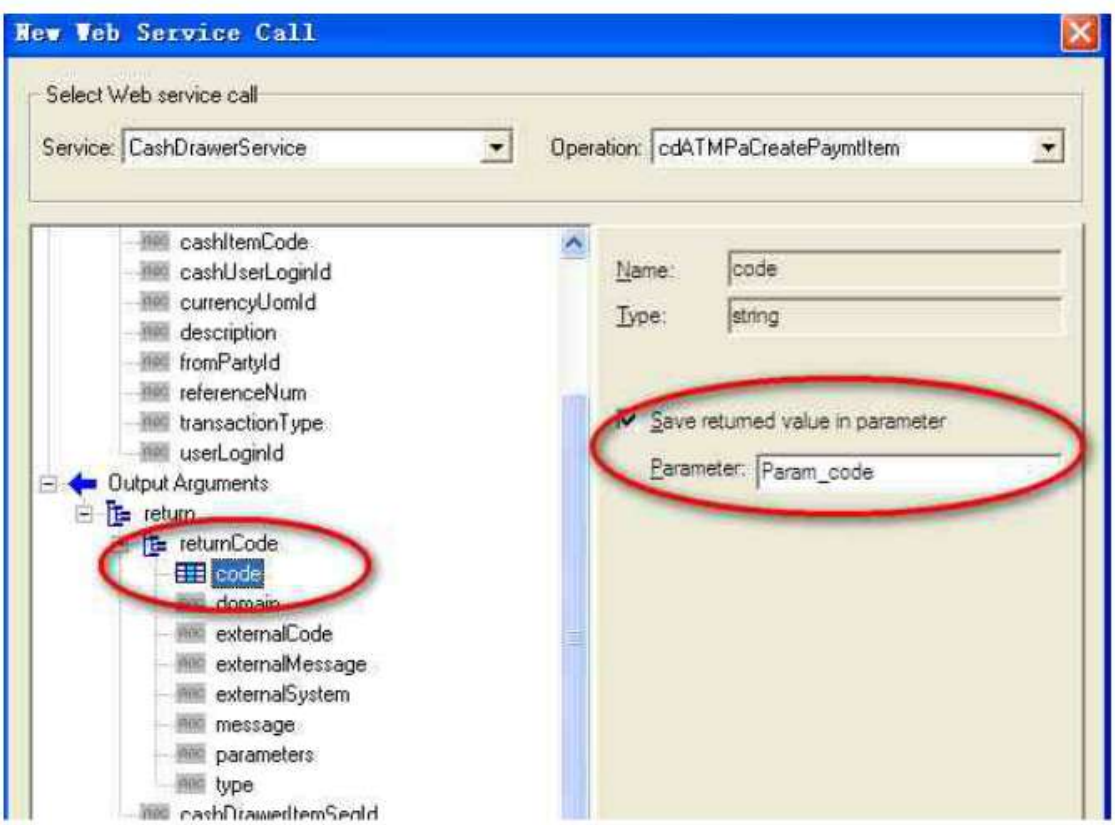


点击 Manage Services，点选 Import，通过 URL 方式选择 WSDL 地址，Analyze with toolkit 选择 Generic Mercury Solution

点击 Add Service Call，从 Operation 中选择想要测试的服务，选中后在对话框左边会显示出该服务所有的输入输出参数，将必选的输入参数赋予初始值。例图中 amount = 100



如果希望将服务返回的结果在脚本中返回，同样点击对话框左边输出参数列表中的返回参数，并将 Save returned value in parameter 勾选



基本设置完毕后点击 OK 回到编辑页面，如下图所示，该脚本基本由两部分组成，上半部分为输入信息，下半部分为输出信息，下面我们对该脚本处理和加工，使其成为真正意义上的测试脚本。

```
Action()
{

    web_service_call( "StepName=cdRecordATMCashDrawerDeposit_101",
        "SOAPMethod=CashDrawerService.CashDrawer.cdRecordATMCashDrawerDeposit",
        "ResponseParam=response",
        "Service=CashDrawerService",
        "Snapshot=t1283848436.inf",
        BEGIN_ARGUMENTS,
        "xml:input=<input><cashDrawerDepositInfos><amount>100</amount><cashItemCode>"
                "</cashItemCode><currencyUomId>USD</currencyUomId></cashDrawerDepositInfos>"
                "<comments></comments><partyIdFrom>0100</partyIdFrom><partyIdTo>0100</partyIdTo>"
                "<paymentMethodType>RECEIPT</paymentMethodType><referenceNum>"
                "123456</referenceNum><transactionType>123456</transactionType><userId></userId>"
                "<userLoginId>01000001</userLoginId></input>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        "return/returnCode/code=Param_code",
        "return/cashDrawerItemSeqIds/transactionId=Param_transactionId",
        END_RESULT,
        LAST);

    lr_think_time(3);

    return 0;
}
```

步骤一：设置协议

点击菜单 Vuser->Run-time Settings->Toolkit Options,在 Emulation type 中选择 MS SOAP(不选的话执行脚本的时候会报错)

步骤二：脚本的参数化

大多数脚本都是需要进行参数化的，参数化是为了保证测试数据的多样性，或者是出于实际中业务特点的考虑，如在多线程并发存款的时候，各线程之间存款的账户是不能相同的。

首先选中要参数化的币种常量，右键后在列表中选择 Replace with a parameter,出现对话框，在 Parameter name 栏中换成有业务意义的参数名称便于维护，其他保持默认点击 OK

这时发现脚本中原常量位置已由带括号的变量替换。

```
:input=<input><cashDrawerDepositInfos><amount>100</amount><cashItemCode>"
    "</cashItemCode><currencyUomId>${CurrencyUomId}</currencyUomId></cashDrawerDepositInfos>"
    "<comments></comments><partyIdFrom>0100</partyIdFrom><partyIdTo>0100</partyIdTo>"
    "<paymentMethodType>RECEIPT</paymentMethodType><referenceNum>"
    "123456</referenceNum><transactionType>020100</transactionType><userId></userId>"
    "<userLoginId>01000001</userLoginId></input>",
```

点击菜单 Parameter List 的快捷方式，打开对话框如图，可以看到 CurrencyUomId（币种）下只有 USD 一个值，点击 Edit with Notepad

打开一个文本文件，在第一个值 USD 下列出其他数据

这样，币种字段参数化就完成了，在参数化后，币种字段将由三个备选值组成，即 USD，RMB，JPY 具体在脚本运行中将执行到哪个值，根据参数设置决定，默认为按顺序执行。

（参数化的详细设置说明参见附录）

步骤三：定义事务

定义事务是为了在性能测试时更准确的统计服务的响应时间，通常将一个服务定义成一个事务。这样在性能测试后事务响应时间就可以看做是服务的响应时间了。如下图所示：

```
Action()
{
    lr_start_transaction("存款");
    web_service_call("StepName=cdRecordATMCashDrawerDeposit_101",
        "SOAPMethod=CashDrawerService.CashDrawer.cdRecordATMCashDrawerDeposit",
        "ResponseParam=response",
        "Service=CashDrawerService",
        "Snapshot=t1283926912.inf",
        BEGIN_ARGUMENTS,
        "xml:input=<input><cashDrawerDepositInfos><amount>100</amount><cashItemCode>"
                "</cashItemCode><currencyUomId>{CurrencyUomId}</currencyUomId></cashDrawerDepositInfos>"
                "<comments></comments><partyIdFrom>0100</partyIdFrom><partyIdTo>0100</partyIdTo>"
                "<paymentMethodType>RECEIPT</paymentMethodType><referenceNum>"
                "123456</referenceNum><transactionType>020100</transactionType><userId></userId>"
                "<userLoginId>01000001</userLoginId></input>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        "return/returnCode/code=Param_code",
        "return/cashDrawerItemSeqIds/transactionId=Param_transactionId",
        END_RESULT,
        LAST);

    lr_end_transaction("存款",LR_PASS);
    lr_think_time(3);

    return 0;
}
```

步骤四：根据服务实际结果和预期结果是否相同来决定事务的成功或失败

性能测试时首先要保证在服务功能执行成功的前提下统计服务的响应时间，否则统计出现逻辑错误的服务的响应时间没有任何意义。下图中开始事务没有变化，只是将结束事务放在 if-else 分支中。该代码块的意思是如果服务实际返回码 Param_code 和预期结果 success 不一致，则事务按失败处理并在日志中显示错误码，否则事务按成功处理。

```
Action()
{
    lr_start_transaction("存款");
    web_service_call("StepName=cdRecordATMCashDrawerDeposit_101",
        "SOAPMethod=CashDrawerService.CashDrawer.cdRecordATMCashDrawerDeposit",
        "ResponseParam=response",
        "Service=CashDrawerService",
        "Snapshot=t1283926912.inf",
        BEGIN_ARGUMENTS,
        "xml:input=<input><cashDrawerDepositInfos><amount>100</amount><cashItemCode>"
                "</cashItemCode><currencyUomId>{CurrencyUomId}</currencyUomId></cashDrawerDepositInfos>"
                "<comments></comments><partyIdFrom>0100</partyIdFrom><partyIdTo>0100</partyIdTo>"
                "<paymentMethodType>RECEIPT</paymentMethodType><referenceNum>"
                "123456</referenceNum><transactionType>020100</transactionType><userId></userId>"
                "<userLoginId>01000001</userLoginId></input>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        "return/returnCode/code=Param_code",
        "return/cashDrawerItemSeqIds/transactionId=Param_transactionId",
        END_RESULT,
        LAST);
    if((strstr(lr_eval_string("{Param_code}"),"success"))==NULL)
    {
        lr_end_transaction("存款",LR_FAIL);
        lr_output_message("----%s",lr_eval_string("{Param_code}"));


    }
    else
    {
        lr_end_transaction("ATM存款",LR_PASS);

    }
    lr_think_time(3);

    return 0;
}
```

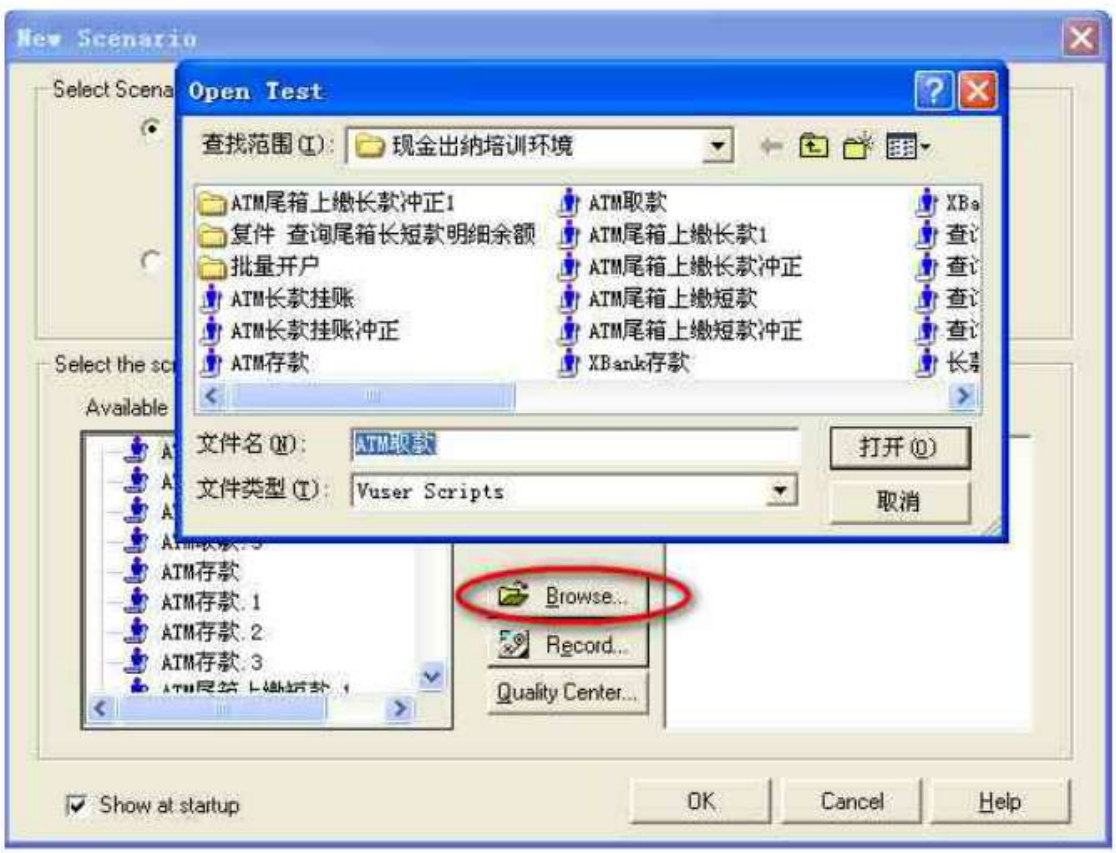
经过以上四个步骤，一个测试脚本就做好了，在测试前，首先要确保单脚本中的事务成功再进行并发测试。一般单脚本事务失败是由于数据问题或者是服务逻辑错误导致的。

3.3. 业务场景分析

业务场景分析是为了在测试中尽可能的模拟实际业务发生时的真实负载，并制定服务性能指标。假设我们设置成 100 个 Vuser 并发访问存款服务，如果存款服务一天的业务量据统计有 100 万，每天员工工作时间为八小时，我们可根据二八原则，估算出服务的每秒通过数指标。（1000000*80%）/（100*8*3600*20%）=1.39 即每秒做 1.39 个服务就能满足实际要求，而服务的响应时间指标就是 1/1.39=0.72，即服务响应时间是 0.72 秒就能满足要求了。合理的业务场景分析能够帮助我们更准确把握实际情况，避免测试负载不足或者测试负载过重。

3.4. 创建并运行场景

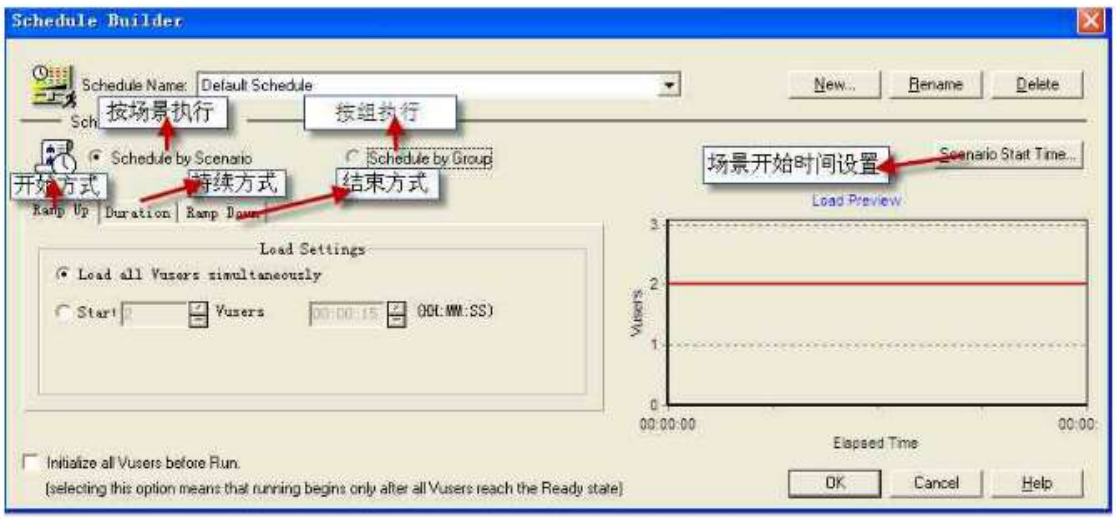
创建场景用到了 LoadRunner 组件 Controller。
打开 Controller,点击 Browse,从对话框中选择待测试的脚本



设置 Vuser 并发数：

	Group Name	Script Path	Quantity	Load Generators
☑	atm取款:2	C:\...\Administrator\桌面\现金出纳\现金出纳培训环境\ATM取款	100	localhost

运行场景设置如下图所示：



➢按场景执行和按组执行是场景执行脚本的两种方式，一般选按场景执行；按组执行是场景中有不止一个脚本的时候，可以设置脚本执行的先后顺序。

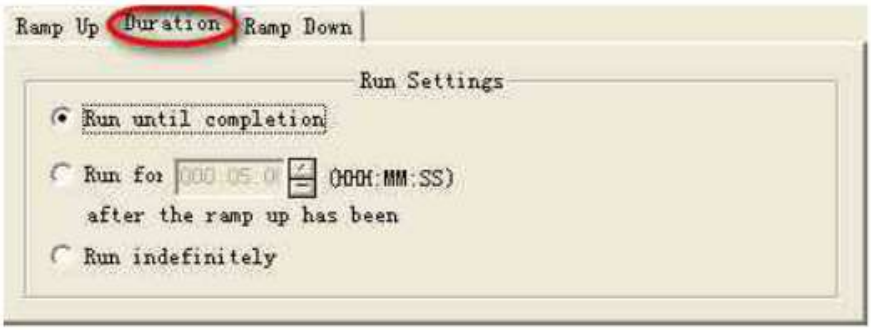
➢场景开始时间设置可以设置场景开始运行时间，达到定时运行的目的。（如下图）

选项一：无延迟运行
选项二：延时一定时间运行
选项三：在某一时间点开始运行

➢开始方式，持续方式，结束方式制定 Vuser 运行方式
开始方式：Vuser 的启动方式

选项一：同时启动所有 Vuser
选项二：每隔若干秒启动若干 Vuser，达到逐渐加压的目的

持续方式：Vuser 的持续方式



选项一：运行直到全部完成
选项二：运行若干时间
选项三：运行一直运行
（选项一在希望精确指定迭代次数的情况下使用，如何指定迭代次数见 Run-Time_Settings）

结束方式：Vuser 的结束方式

选项一：同时结束所有 Vuser
选项二：每隔若干秒结束若干 Vuser，达到逐渐减压的目的

设置好场景后点击 Run-Time Settings 进行运行时设置（如下图）

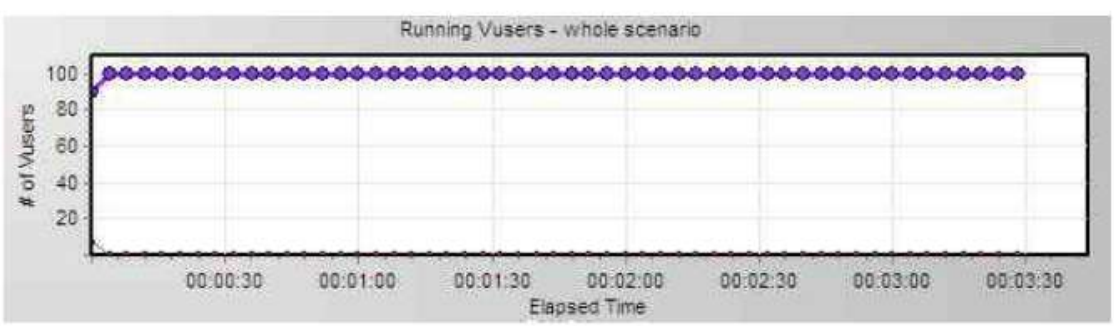
主要参数选项设置如下：

➢Run Logic：设置 Vuser 迭代次数，该迭代次数在持续方式选择选项一时才有效，如果选择选项二，则以持续时间为准。
➢Log：输出日志级别。日志输入给我们提供了有用信息，但是也会影响响应时间。建议在调试场景时输出详细日志，在真正运行场景时设置无日志输出
➢Think Time:如果无业务需求，设置成忽略思考时间

3.5. 监控场景

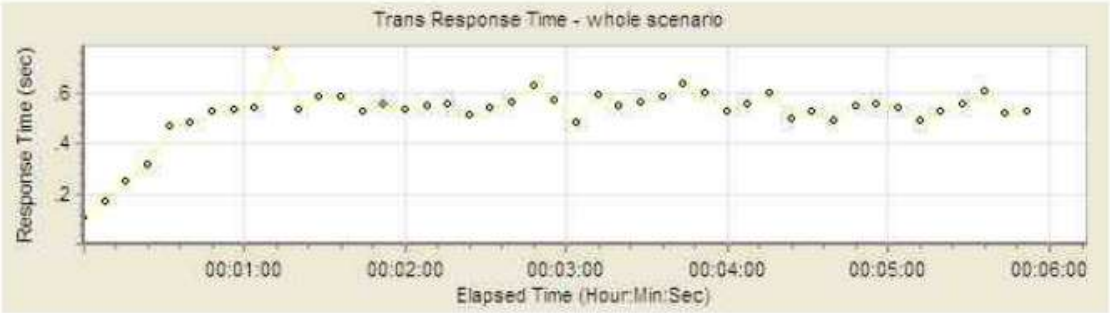
监控场景我们一般关注四个指标：虚拟用户数，响应时间，每秒点击数和每秒事务数。

虚拟用户数：即同时并发的用户数，直接体现了对待测应用的负载强度。

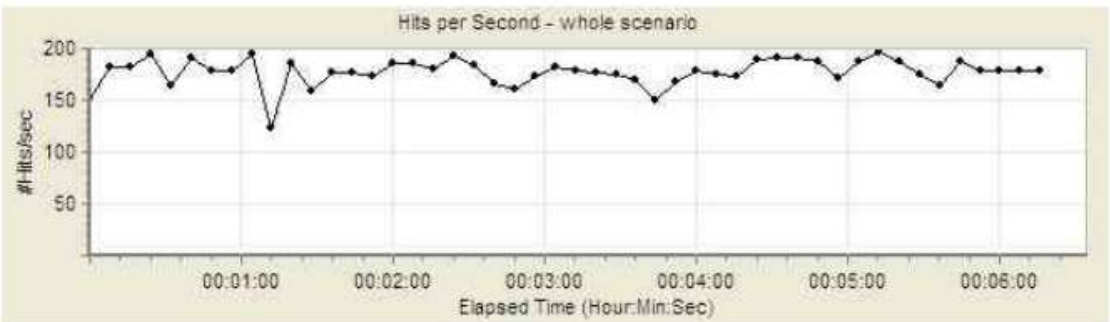


响应时间：响应时间是可以判断一个被测应用系统是否存在性能瓶颈的最直观的要素。响应时间一般包括最大响应时间和平均响应时间，响应时间包括网络上的传输时间，WEB 服务器上处理时间、应用服务器上的处理时间、数据库服务器上的处理时间，响应时间不包括

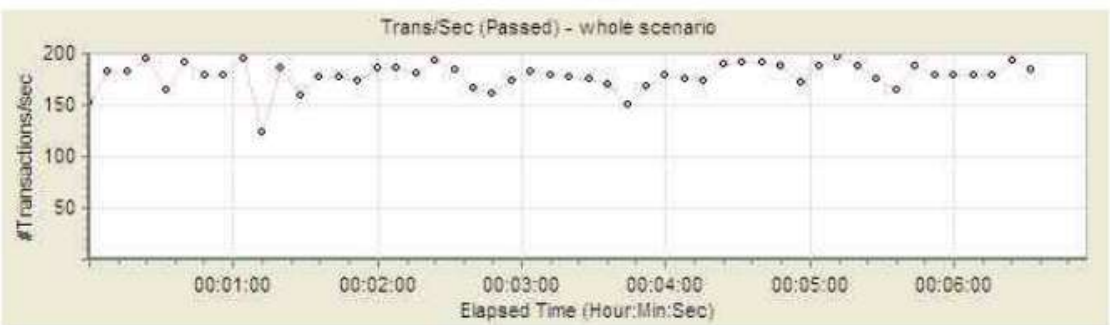
浏览器上的内容显示时间。



每秒点击数：每秒点击数（HPS）是用来衡量很多用户使用客户端进行操作，向服务器发送请求的效率。我们认为 HPS 表现的是最终用户的整体行为，是衡量在线负载程度的一个指标。



每秒事务数：TPS (Transactions Per Second)是估算应用系统性能的重要依据。其意义是应用系统每秒钟处理完成的交易数量，尤其是交易类系统。一般的，评价系统性能均以每秒钟完成的技术交易的数量来衡量。

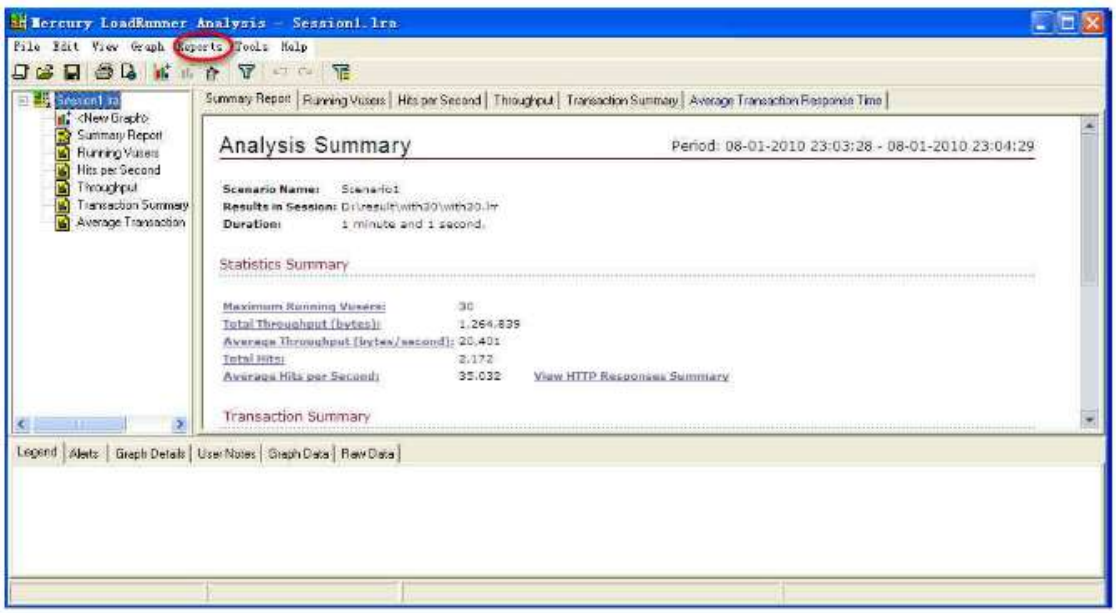


3.6. 分析场景

分析场景是为了更深入的了解服务运行情况，发现服务的瓶颈，并解决服务性能问题。通常，我们可以通过各种指标之间的关系来判断服务运行是否正常。例如随着并发用户数的增加每

秒事务数也应呈递增趋势，如果每秒事务数并没有随用户数增加，则可能在某些方面遇到瓶颈，网络传输能力或者数据库连接池的限制都可能造成这种情况。

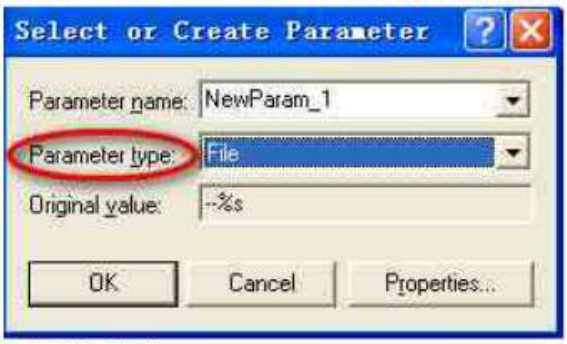
通过 LoadRunner Analysis 打开后缀为.lra 的 结果文件，里面有测试服务的测试结果和各种监测指标的图示。点击 Reports 可以生成 LoadRunner 的性能测试报告。



附录：

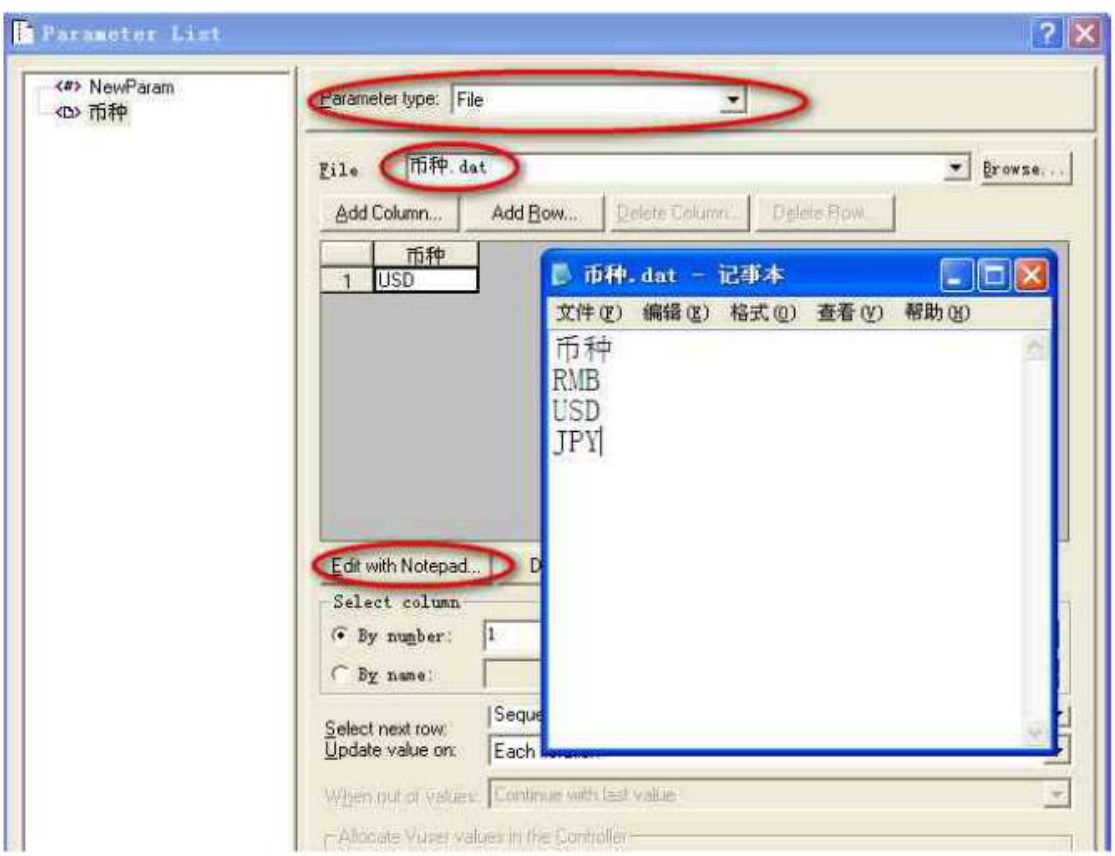
参数化的详细设置说明

参数的类型：在参数化一个常量的时候有很多参数类型可供选择（如下图）：



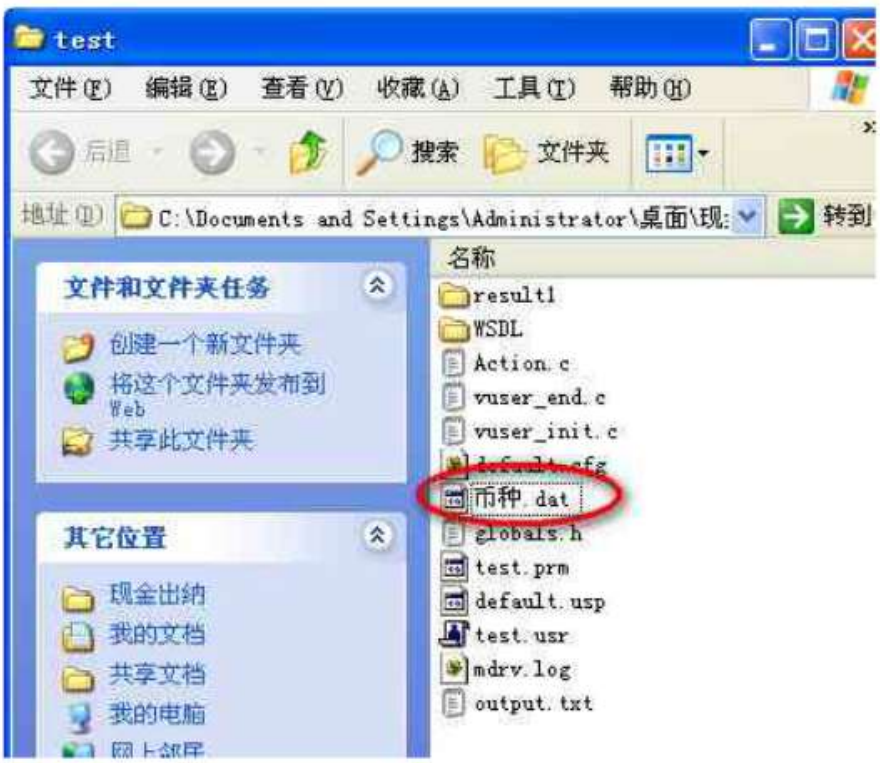
一般常用的有 File,Unique Number ,Date/Time

File: 通过文件的形式参数化常量,这种方式可以将参数化数据存储在一个.dat 文件中，是一种最常用的方式。



编辑该 dat 文件有两种方式：

1. 通过点击上图 Edit with Notepad 打开.dat 文件编辑
2. 找到该文件在硬盘里的绝对位置进行编辑
   方法：将光标放在脚本任意位置，右键点选 Open script directory 可以打开该脚本所在目录，找到该 dat 文件双击打开编辑即可



参数在场景中的运行方式：（如下图）通过 select next row 和 update value on 的不同组合来决定参数在场景中不同的取值方式。

select next row:不同 Vuser 之间的取值方式

Sequential: 每个 Vuser 均是顺序取值

Random: 每个 Vuser 均是随机取值

Unique: 每个 Vuser 取值不能相同

Update value on: 取值更新方式

Each occurrence:每次出现更新

Each iteration:每次迭代更新

Once：在所有迭代中只取一个值保持不变。

如果在一个脚本中该参数只出现一次，则 Each occurrence 和 Each iteration 是相同的。

Allocate Vuser values in the Controller:该选项为 select next row 选择 Unique 时所特有的。

选项 1：LoadRunner 为每个 vuser 自动分配参数值。如果采用这种方式，假设在场景运行时设置迭代次数为 2，则第一个 Vuser 使用参数中的前 1，2 个，第二个 Vuser 使用参数中的前 3，4 个，第三个 Vuser 使用参数中的前 5，6 个...以此类推。

选项 2 ：手工设定分配参数值。假设分配 n 个值给每个 Vuser，则第一个 Vuser 使用参数中的前 1-n 个，第二个 Vuser 使用参数中的 n+1-2n 个，第三个 Vuser 使用参数中的前 2n+1-3n 个...以此类推。

select next row 和 Update value on 这两个选项是两个描述不同维度的设置，select next row 是描述不同 Vuser 第一次取值时彼此之间的作用关系。Update value on 是描述在一个 Vuser 里不同迭代里面的取值关系。这两个选项的不同的搭配可以组合出多种参数取值方式，用来满足不同的业务需求。

假设某参数有 1.2.3.4.5.6.7.8.9.10 这 10 个备选值，场景设置为运行 3 个 Vuser，迭代次数为 2，现对不同搭配做比较：

Sequential+ Each iteration：这种方式最常用，表示每个 Vuser 都是顺序取值，每次迭代更新。运行后取值如下：

Vuser1: 1，2

Vuser2: 1，2

Vuser3: 1，2

Sequential+ Once：这种方式表示表示每个 Vuser 都是顺序取值，每次迭代取值不变，运行后取值如下：

Vuser1: 1，1

Vuser2: 1，1

Vuser3: 1，1

Unique+Once：表示每个 Vuser 彼此之间各不相同，但是每个 Vuser 在每次迭代中值保持不变。在测试存款性能的时候，由于在实际业务中不存在两个人同时向同一个账户存钱的可能，所以对存款账号的参数化就可以采用这种方式，保证不同的用户操作不同的账号，不会在一个账号上有并发产生，运行后取值如下：

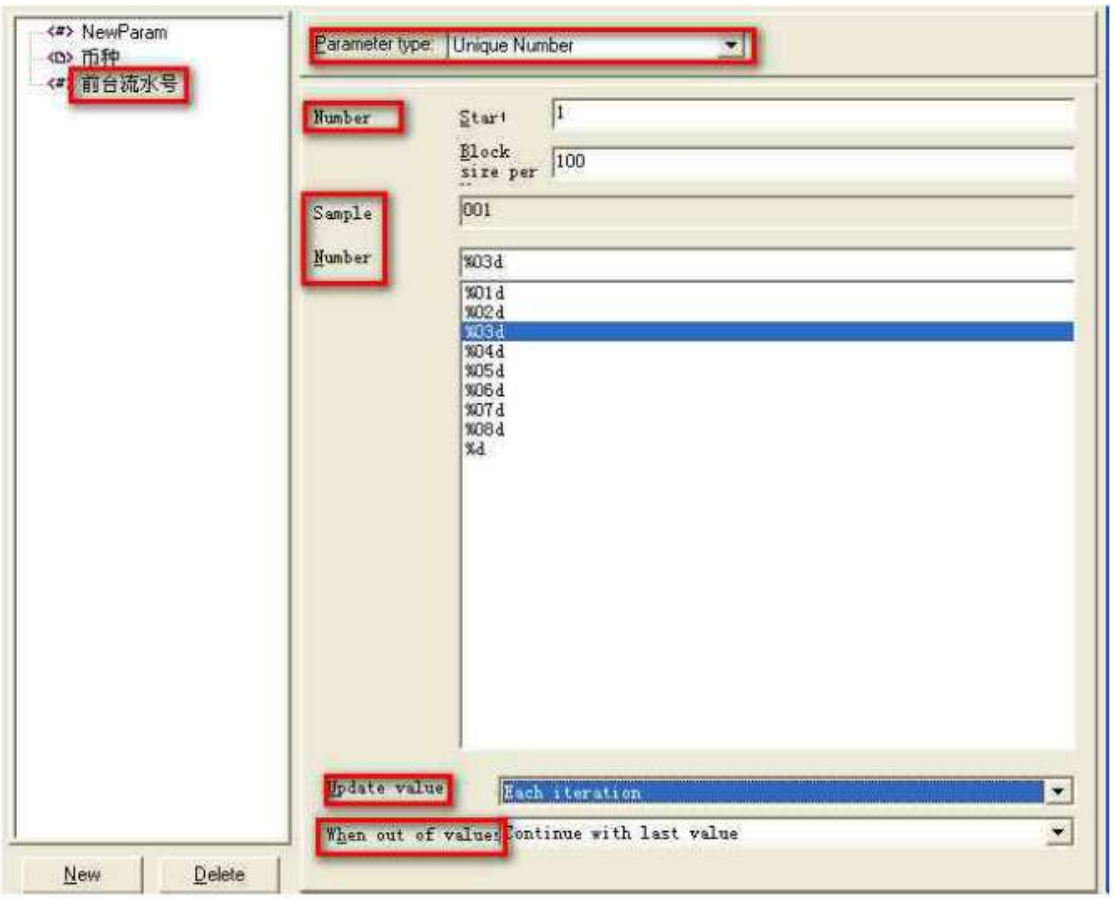
Vuser1: 1，1

Vuser2: 2，2

Vuser3: 3，3

Unique Number：取得一个唯一值，一般对有特殊业务要求的值进行参数化时选择。

比如前台流水号必须唯一，便可以选择这种方式参数化。



如上图，将前台流水号字段以唯一值方式参数化，各选项说明如下：

Number：代表一个 Vuser 里可选值的取值范围，如图所示为 1-100。如果该脚本创建好后，在执行场景的时候希望有五个 Vuser 并发，则这五个 Vuser 里前台流水号参数的取值依次是:1-100,101-200,201-300,301-400,401-500.

Sample 和 Number：这两个字段用来规定取值的输出样式，如上图选择%03d,则 1 的输出样式为 001，如果选择%04d,则 1 的输出样式为 0001.

Update value:取值更新方式

Each occurrence:每次出现更新

Each iteration:每次迭代更新

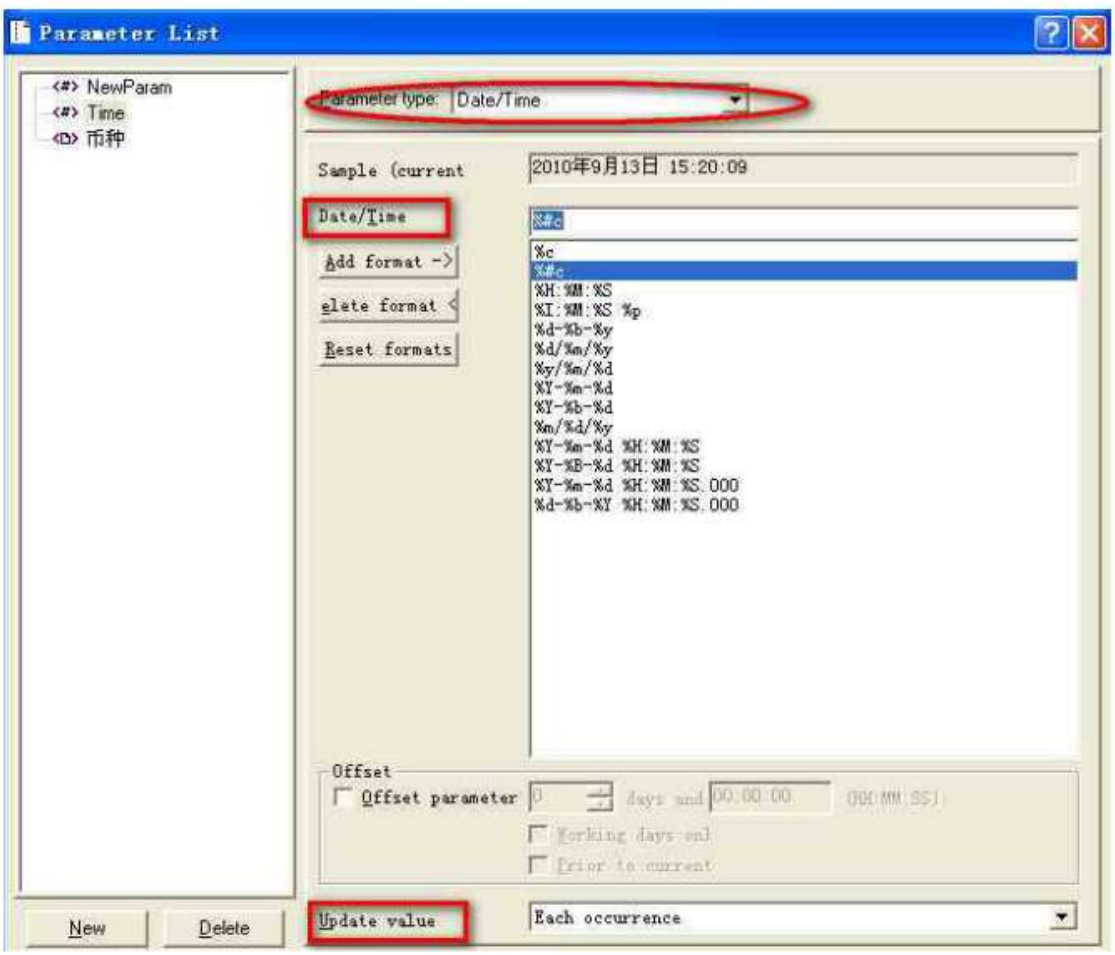
Once：在每次迭代中只取一个值保持不变。

如果在一个脚本中该参数只出现一次，则 Each occurrence 和 Each iteration 是相同的。

When out of values:当范围内可选值都已取过后的取值方式。比如 Number 中设置为 1-100，则在运行脚本的迭代次数达到 100 次时，则取值范围内的值都已取过一遍，此时如果迭代还没结束则有三种取值方式.

Abort Vuser:如果超出范围则 LoadRunner 报错。报错信息样式如下：
Parameter '前台流水号': All values in unique block already used. Block size is 5. The Vuser is aborted according to "When Out Of Values" policy
Continue in a cyclic manner: 继续循环取值
Continue with last value:继续取最后一个值。

Date/Time:取得当前日期或时间



Date/Time : 选择时间显示方式
Update Value: 时间更新方式（参见其他参数方式该字段设置）