*версия файла: “1.3”*

**“LR\_PARAM” – утилита для работы с LoadRunner скриптом.**

*author: vgonchar\_ext*

[***https://github.com/vaduxam1/LR\_PARAM***](https://github.com/vaduxam1/LR_PARAM)

***скачать последнюю версию:***

[***https://github.com/vaduxam1/LR\_PARAM/archive/master.zip***](https://github.com/vaduxam1/LR_PARAM/archive/master.zip)

Оглавление

[1) Общее описание 4](#_Toc3340358)

[*Для правильной работы утилиты* 4](#_Toc3340359)

[*Запуск утилиты* 4](#_Toc3340360)

[*При старте через lr\_start.cmd – автоматически установит python* 4](#_Toc3340361)

[*Если не работает* 4](#_Toc3340362)

[2) Задачи, решаемые утилитой. 5](#_Toc3340363)

[1) Поиск {param} имен в action.c, для дальнейшей их параметризации 5](#_Toc3340364)

[2) До создания wrsp есть возможность: редактирования и выбора всех вариантов LB/RB 5](#_Toc3340365)

[3) Удаление, web\_ объектов, по шаблону, из action.c 5](#_Toc3340366)

[4) Осуществлять навигацию по action.c, переходы между объектами, транзакциями и т.д 6](#_Toc3340367)

[6) Сделать имена {web\_reg\_save\_param} в action.c хоть как-то осмысленными 6](#_Toc3340368)

[7) Разное 6](#_Toc3340369)

[3.1) Теория LoadRunner параметризации, в контексте утилиты. 7](#_Toc3340370)

[1) Пользователь осуществляет запись теста в LoadRunner 7](#_Toc3340371)

[2) LoadRunner сохраняет сетевой “трафик” теста в подкаталог “data” 7](#_Toc3340372)

[*a)* “Request” запросы от пользователя к серверу 7](#_Toc3340373)

[*b)* “Response” ответы сервера, на запросы пользователя 7](#_Toc3340374)

[*c)* “.c” (action.c) LoadRunner файлы(копии) нагрузочного скрипта 7](#_Toc3340375)

[d) “.inf” файлы – связь Request/Response, с web\_ объектами внутри “.c” 7](#_Toc3340376)

[e) не представляющие ценности Request или Response файлы, бин. и т.д. 8](#_Toc3340377)

[f) другие неважные файлы, логи и т.д. 8](#_Toc3340378)

[inf файлы: 8](#_Toc3340379)

[Пример “t1.inf” 8](#_Toc3340380)

[Ошибочно считать, что из названия Request/Response-файла 8](#_Toc3340381)

[3) LoadRunner создает скрипт - “.c” файлы 9](#_Toc3340382)

[action.c LoadRunner тест 9](#_Toc3340383)

[Объекты некоторых типов, могут быть связаны с Request/Response файлами 9](#_Toc3340384)

[Обычно inf-файлов, больше, чем snapshot web\_ объектов 9](#_Toc3340385)

[Можно выстроить цепочку: 10](#_Toc3340386)

[4) Затем необходимо провести “параметризацию” 10](#_Toc3340387)

[a) В коде action.c, найти все id/имена “переменных” 10](#_Toc3340388)

[b) Выполнить для каждого найденного {param} 10](#_Toc3340389)

[*5)* LoadRunner cкрипт готов к запуску 10](#_Toc3340390)

[3.2) Создание web\_reg\_save\_param. 10](#_Toc3340391)

[LB/RB способ 11](#_Toc3340392)

[В данном примере Ord=2 11](#_Toc3340393)

[3.3) Вопросы 12](#_Toc3340394)

[4) Назначение основных окон. 12](#_Toc3340395)

[5) Главное окно. 13](#_Toc3340396)

[Термины 14](#_Toc3340397)

[Основные элементы управления 14](#_Toc3340398)

[Окно предназначено для 14](#_Toc3340399)

[Описание 14](#_Toc3340400)

[Работа без чекбокса auto 15](#_Toc3340401)

[6) Action.c окно. 16](#_Toc3340402)

[6.1.1) поиск и параметризация всех {param} в action.c. 16](#_Toc3340403)

[6.1.2) Запуск параметризации конкретным методом 17](#_Toc3340404)

[6.1.3) Найти все варианты, создания {web\_reg\_save\_param}, для выделенного param. 17](#_Toc3340405)

[6.1.4) Найти все “одноименные” {param} – имя которых начинается на выделенный текст. 17](#_Toc3340406)

[6.1.5) Найти все {param}, имеющие выделенный LB=. 18](#_Toc3340407)

[6.1.6) создать {web\_reg\_save\_param}, для выделенного param. 18](#_Toc3340408)

[6.2) Редактирование LB/RB в уже сформированном web\_reg\_save\_param и пересчет Ord=. 18](#_Toc3340409)

[6.3.1) Удаление из action.c, уже сформированного web\_reg\_save\_param. 19](#_Toc3340410)

[6.3.2) Удаление из action.c, всех сформированных web\_reg\_save\_param. 19](#_Toc3340411)

[6.3.3) удалить web\_ объекты 19](#_Toc3340412)

[6.3.4) удалить lr\_think\_time объекты 20](#_Toc3340413)

[6.4.1) переименовать транзакции 20](#_Toc3340414)

[6.4.2) переименовать (+) web\_reg\_save\_param 20](#_Toc3340415)

[6.4.3) переименовать (-) web\_reg\_save\_param 21](#_Toc3340416)

[6.5) Декодирование выделенного фрагмента. 21](#_Toc3340417)

[6.6) Настройка информационных комментариев и имени web\_reg\_save\_param 21](#_Toc3340418)

[6.7) настройка общих переменных 22](#_Toc3340419)

[6.8.1) Показ дополнительных управляющих виджетов 22](#_Toc3340420)

[6.8.2) Показ дополнительных навигационных виджетов 22](#_Toc3340421)

[6.9) Показ сводной информации 23](#_Toc3340422)

[6.10) открытие/сохранение action.c 23](#_Toc3340423)

[6.11) Просмотр Response файлов ответов, при записи и воспроизведении 24](#_Toc3340424)

[6.12) Отображение схемы распределения web\_reg\_save\_param 24](#_Toc3340425)

[Приложение: Файлы утилиты 25](#_Toc3340426)

**1) Общее описание**.

*Утилита* ***не призвана заменить LoadRunner****. Это “инструмент”, который помогает решить* ***задачи*** *“****action.c****” файла, которые LoadRunner решает недостаточно хорошо, либо не решает их вовсе. Предоставляет более гибкие/настраиваемые – “полезные” инструменты, и некоторые “бесполезные” – как альтернативу.*

*При параметризации, в полностью автоматическом режиме, вероятность получения с первого раза, 100% “готового” состояния action.c скрипта – есть, но небольшая. Т.к. предполагается, что после запуска скрипта в LoadRunner, выявятся проблемные места, и далее они* ***решаются при помощи утилиты, и анализа причин****, почему проблемные web\_reg\_save\_param’s должны создаваться как-то по-особенному. Иногда некоторые {param} не могут быть созданы LB/RB-способом и необходим regexp-способ, либо может потребоваться изменение каких-то настроек, то есть необходим полуавтоматический/итерационный/ручной подход.*

***Для правильной работы*** *утилиты,* ***все*** *файлы* ***(vuser\_init.c, action.c, vuser\_end.c)****, необходимо* ***объединить в один*** *файл, например* ***action****.c. Саму утилиту (каталог “lr\_lib” и файлы ”lr\_start.\*”) необходимо скопировать внутрь каталога LoadRunner-скрипта, рядом с подкаталогом “data”.*

***Запуск*** *утилиты, файлом “****lr\_start.py****” или* ***“lr\_start.cmd”****.*

*Утилита использует сторонние библиотеки (для некритичного функционала), поэтому они включены в ее состав (lr\_lib\whl\install\\*.whl), и при необходимости, установятся автоматически.*

***При старте*** *через**lr\_start****.cmd –*** *автоматически* ***установит python*** *3.4.4 x32, если python не установлен в OS.*

***Если не работает*** *(****буфер обмена****/меню правой кнопки/что-то похожее) – поменять раскладку клавиатуры на ENG.*

***Если не запускается*** *– возможно python уже установлен, а offline-библиотеки-py-whl не подходят к нему.  
Тогда можно поступить любым из путей:  
 1)* ***Установить*** *python из "\lr\_lib\whl\python-3.4.4.msi".   
 Установить все whl из "\lr\_lib\whl\install\", путем выполнения в cmd:   
 "pip install имя\_Файла\_модуля.whl", например:*

*" c:\Python34\Scripts\pip.exe install c:\LR\_SCR\lr\_lib\whl\install\typing-3.6.6-py3-none-any.whl"*

*" c:\Python34\Scripts\pip.exe install c:\LR\_SCR\lr\_lib\whl\install\keyboard-0.13.2-py2.py3-none-any.whl"  
 Запустить утилиту, именно с помощью данной версии python.  
 2)* ***До установить*** *библиотеки из интернета, путем выполнения в cmd:   
 "pip install имя\_Модуля", например:*

*" c:\Python\Scripts\pip.exe install typing"*

*" c:\Python\Scripts\pip.exe install keyboard"*

**2) Задачи, решаемые утилитой.**

Сводное описание основных задач, решаемых при помощи утилиты.

1. Поиск {param} имен в action.c, для дальнейшей их параметризации*.*

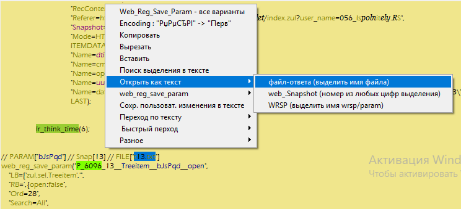
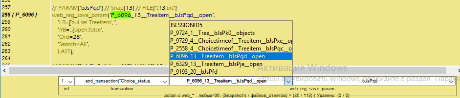
*Есть 6 настраиваемых методов поиска, и много способов ими воспользоваться. Поиск производится в (и/или) Request/ Response/ Action.c/ любых файлах. Просмотр всех {param} в action.c до их параметризации (удалить что-то лишнее). Просмотр web\_reg\_save\_param без создания в action.c (вывод всех вариантов создания, для анализа).*

*Возможность параметризации action.c в автоматическом режиме (action окно), до состояния “готового” скрипта.*

1. До создания wrsp есть возможность: редактирования и выбора всех вариантов LB/RB*, настройки правил создания. Авто-вычисление Ord=. Виджеты (главное окно, и другие), для просмотра/ редактирования/ создания всех возможных вариантов, которыми мог бы быть создан web\_reg\_save\_param для {param}, с возможностью редактирования каждого варианта.*
2. Удаление, web\_ объектов, по шаблону, из action.c*.*

*Например, ошибочно попавшие в тест google счетчики, “dummy”, и другое. Групповое переименование транзакций и {param} имен. Корректное удаление уже сформированных web\_reg\_save\_param.*

1. Осуществлять навигацию по action.c, переходы между объектами, транзакциями и т.д*.*

*При выборе значения – переход в область теста.*

*Отрыть как текст – выделить имя – результат – открытие в блокноте.*

1. ** Анализировать action объекты на наличие WARGING*:*

*Неправильное использование web\_reg\_save\_param, наличие русских символов для перекодировки,*

*поиск web\_ google, и любых других проблемных web\_. Собрать статистику web\_объектов, какие именно в них используются {param}, сколько раз, где внутри action кода находится сам объект, внутри какой транзакции, какой по счету, и тому подобное. Все описанное вывести, в комментарии “//lr:” к объекту в action.c тексте, и на виджеты.*

1. Сделать имена {web\_reg\_save\_param} в action.c хоть как-то осмысленными*:*

*param:* ***LoadRunner*** *web\_reg\_save\_param:* ***Утилита*** *web\_reg\_save\_param:*

*“****bJsPkh****” 🡪 “{****PARAM****\_****7878****}” 🡪 {****P\_7878\_25\_\_Choicestatus\_\_Button\_\_bJsPkh\_\_Ok****}”*

**

*Имя хранит информацию об:*

***Inf*** *= “25”*

***LB*** *= “Button”*

***RB*** *= “Ok”*

***Transantion*** *= “Choicestatus”*

***Param*** *= “bJsPkh”*

*Имеет* ***уникальный*** *номер = “7878”*

***Общую*** *вводную часть = “P\_”*

1. Разное*:*

* *Выполнять бэкапы, перед изменениями.*
* *Формировать и каким-либо образом отображать сопутствующую информацию, например, список созданных web\_reg\_save\_param, номеров объектов, имеющих WARNING, и тому подобное.*
* *Быстрый просмотр Request/Response/web\_/wrsp, по их имени, из меню мыши.*
* *Форматировать action текст.*
* *Другое…*

**3.1) Теория LoadRunner параметризации, в контексте утилиты.**

1. Пользователь осуществляет запись теста в LoadRunner.



В результате записи создается каталог с файлами скрипта, основные файлы:

1. “data” подкаталог – описан в пункте 2).
2. “.c” файлы *нагрузочного скрипта* – описаны в пункте 3).
3. другие – настройки скрипта, запускающий файл, логи и т.п.
4. LoadRunner сохраняет сетевой “трафик” теста в подкаталог “data”, в видефайловойструктуры:



LoadRunner***data*** файлы делятся на **типы**:

1. “Request” запросы от пользователя к серверу*:*

*“t1\_RequestHeader.txt”, “t10\_RequestBody.txt”.*

1. “Response” ответы сервера, на запросы пользователя*:*

*“t10.txt”, “t10\_ResponseHeader.txt”.*

1. “.c” (action.c) LoadRunner файлы(копии) нагрузочного скрипта*:*

*vuser\_init.c, action.c, vuser\_end.c*

1. “.inf” файлы – связь Request/Response, с web\_ объектами внутри “.c”*:*

“*t1.inf*”, “*t12.inf*”

1. не представляющие ценности Request или Response файлы, бин. и т.д.:

“*resporse\_excel.png*”, “*data-test.zip*”.

1. другие неважные файлы, логи и т.д.:

“*vugen\_modules\_dump.txt*”*, “interfaces.h”.*

### **inf файлы:**



*LoadRunner объединяет Request/Response файлы в группы:* ***inf****-файлы - некую последовательность отдельных “логических” действий. Файл inf может представлять из себя, например, один клик по кнопке, или выборку из нескольких фильтров виджета. Каждый такой inf, содержит в себе ссылки на все файлы (с данными), необходимыми, как для отправки данных на сервер, так и возвращаемые сервером данные (на момент записи скрипта), в ответ на посланный запрос, и любые другие необходимые сведения.*

Пример “t1.inf”*, где прописаны ссылки на принадлежащие ему Request/Response:*



Ошибочно считать, что из названия Request/Response-файла*, можно напрямую узнать номер inf-файла, которому он принадлежит. Например, можно подумать, что файл “t4\_RequestBody.txt” принадлежит “t4.inf”, это так лишь отчасти. На самом деле он может принадлежать сразу нескольким inf-файлам, например “t4.inf” и “t7.inf”, а название отражает лишь первый из них.*

*Следует учитывать, что клик по кнопке, либо любое другое событие, может вызывать другие последующие события, не всегда визуально отображаемые, в следствие чего LoadRunner может разбить, казалось-бы “атомарное” действие, на несколько inf. Разбивка трафика на отдельные файлы, происходит в соответствие с настройками и протоколом записи, выбранными пользователем в LoadRunner, перед началом записи.*

1. LoadRunner создает скрипт - “.c” файлы, в дальнейшем именуемые ***action.c*** :



*Это как минимум 3 файла (action может быть несколько):*

*vuser\_init.c # авторизация*

*action.c # тело1 теста*

*… # телоN теста*

*vuser\_end.c # логаут*

action.c LoadRunner тест*, это в некотором смысле, “представление” множества inf-файлов data-каталога, в виде LoadRunner кода скрипта - множества* ***объектов****, некоторые из которых ссылаются на inf-файлы, в дальнейшем именуемых* ***web\_****,**и иного “неважного” кода.*



Объекты некоторых типов, могут быть связаны с Request/Response файлами*, через ссылку на inf-файл. Связь есть у объектов, содержащих внутри своего тела строку “****snapshot=t{номер}.inf****”, тип таких объектов начинается на “****web\_****”.*

*Например, строка связи*, *файлов из* “*t1.inf*”, с *action.c объектом* *web\_url:* “*snapshot=t1.inf*”.



Обычно inf-файлов, больше, чем snapshot web\_ объектов*. Из текста action.c, LoadRunner или пользователь может удалить некоторые snapshot объекты, например если они не используются, записаны по ошибке, либо по каким-то иным причинам. Для теста важны только “Request” данные. Данные заранее “извлечены” из Request-файлов, и уже содержатся в action.c, в виде кода объектов.*

*Параметры, которые* ***используют*** *snapshot web\_ объекты, нужно искать в файлах* ***Request****.*

*Параметры, которые* ***создают*** *snapshot web\_ объекты, нужно искать в файлах* ***Response****.*

### Можно выстроить **цепочку**:

*каталог /data/: связь: скрипт action.с:*

***Request****/****Response*** *файлы 🡨🡪* ***inf*** *файлы 🡨🡪* ***web\_*** *snapshot объекты*

1. Затем необходимо провести “параметризацию” action скрипта.

*Это значит, что необходимо:*

1. В коде action.c, найти все id/имена “переменных”*, в дальнейшем именуемые* **{param}**, например “*bJsPw1*”, соответствующие любым виджетам, событиям, и другим *id*, которые были задействованы, при записи скрипта.
2. Выполнить для каждого найденного {param}:
3. **Определить** в *action.c,* **snapshot**-номер **web\_** объекта, в котором {param} встречается/**используется** **в** **первый раз**.
4. **Найти** *подходящий* “***Response***” файл и его ***inf***-номер, затем,***создать******web\_reg\_save\_param*** *согласно правилу:*
   * + *Response-файл из которого создается web\_reg\_save\_param, должен принадлежать, inf-файлу с номером меньшим, чем snapshot-номер web\_ объекта, в котором {param} в первый раз используется -* ***пункт I)****.*
     + *Само создание описано в пункте 3.2) Создание web\_reg\_save\_param.*
5. ***Вставить*** *web\_reg\_save\_param в action.c,* ***перед******web****\_ объектом с snapshot-номером, равным inf-номеру Response файла из* ***пункта II)****.*

***Заменить*** *в action.c, все упоминания оригинального {param}, на имя полученного {web\_reg\_save\_param}.*

*При замене следует учитывать, что если заменяем, например “zkau\_1” на “{zkau\_wrsp}”, то в тесте могут быть и другие, “похожие” {param}, например “zkau\_1\_1” или “Szkau\_10”. И если заменять “вслепую”, из меню “Найти и заменить”, можно ошибочно заменить и эти “похожие” имена {param}, тем самым испортив скрипт так: “S{zkau\_wrsp}0”.*

1. LoadRunner cкрипт готов к запуску. Выполнить прогон скрипта в LoadRunner, при необходимости исправить проблемы, с использованием утилиты.

**3.2) Создание web\_reg\_save\_param.**

*web\_reg\_save\_param вытаскивает новое id/имя-переменной {param} из ответа (Response) action web\_ объекта, при каждом воспроизведении/итерации скрипта. Затем этот новый {param} используется (Request) в других web\_. Все web\_ использующие {param}, должны располагаться строго после web\_ создателя web\_reg\_save\_param, т.к. переменная не может использоваться, до ее определения.*

*web\_reg\_save\_param создается 2 основными способами:*

1. Regexp – с помощью регулярного выражения, здесь не используется.
2. LB/RB – с помощью “относительных координат ( x, y, z )” – ( LB=, RB=, Ord= )

**LB/RB способ**,на примере ‘tY6Q86’:

1. *Для начала, необходимо найти* ***Response****-файл, содержащий {param}. Этот файл, должен принадлежать (связь Response/inf/action) inf-файлу, с номером меньшим, чем snapshot номер web\_ action объекта, в котором {param} в первый раз используется.*



1. Определить “корректный” **LB**



1. Определить “корректный” **RB**



1. Определить **Ord** – сколько раз в тексте надо “взять” LB/RB, чтобы между ними получился {param} из пункта 1).

Т.к. не факт, что выбранные LB/RB, приведут к {param} с первого раза.

В данном примере Ord=2:

// PARAM["tY6Q86"] // Snap[22] // FILE["t22.txt"]

web\_reg\_save\_param("P\_5836\_22\_\_A\_\_tY6Q86\_\_processedDocsScrollerLabel",

"**LB**= 'zul.wgt.A','",

"**RB**=’,{id:'processedDocsScrollerLabel'",

"**Ord**=2",

"Search=All",

LAST);

*Следует учитывать, что иногда {param}, по мере теста может переопределяться/обновляться, например, сначала он указывал на один виджет, затем на другой. В этом случае, для одного и того же {param}, необходимо использовать несколько различных web\_reg\_save\_param, причем уже не обязательно одноименных. Еще более запутанная ситуация может возникнуть, если web\_reg\_save\_param создан, например, в vuser\_init.c, а используется в action.c, при этом идет n-нная итерация action, и в ходе итерации есть переопределение {param}.*

**3.3) Вопросы**

Из всего описанного, можно выделить вопросы:

1. *Как* **найти все {param}***, которые использует action.c.*
2. *Как найти* **Response** *файл и* **inf-номер***, для создания web\_reg\_save\_param, без учета возможности его обновления/переопределения по мере теста.*
3. *Как найти* **inf-номер***, для создания web\_reg\_save\_param, в аспекте того, что он может обновиться/переопределиться по мере выполнения теста.*
4. *Как создать web\_reg\_save\_param, в разрезе* **Lb=, Rb=, Ord=***.*
5. *Как “корректно”* **обработать** *web\_reg\_save\_param – правильно заменить предназначенные только для него {param}, на новое имя {web\_reg\_save\_param} – чтобы не было таких: “{zkau\_wrsp}\_1”.*

******4) Назначение основных окон.**

1. **слева**: главное окно – *настройка и* ***ручное*** *создание web\_reg\_save\_param*.

Отвечает за цепочку: “ *Request/Response файлы 🡨🡪 inf файлы*”.

Отвечает за **вопрос**: **№** **4**.

Частично отвечает за **вопрос**: **№** **2**:

*ничего “не знает” об action.c, поэтому: не ограничивает в автоматическом режиме, диапазон inf/snapshot номеров для поиска {param} в Response файлах - т.к. не может знать в каком именно номере snapshot web\_, первый раз используется искомый {param}, к тому же будет искать файлах с inf, web\_ объекты которых возможно удалены из action.c.*

1. **справа**: дочернее action.c окно, ***автоматическое*** *создание web\_reg\_save\_param*.

Отвечает за цепочку: “ *inf файлы 🡨🡪 web\_ snapshot объекты*”.

Отвечает за **вопрос**: **№** **1, 5**.

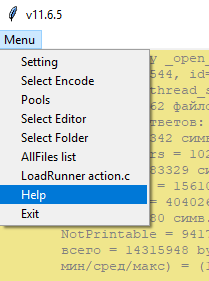
Отвечает за **вопрос**: **№** **2**:

*задает диапазон inf/snapshot номеров для поиска {param} в Response файлах, в зависимости от того, в каком web\_ первый раз используется искомый {param}.*

Частично отвечает за **вопрос**: **№** **3**:

*В ручном режиме, в action.c тексте, можно отобразить/выбрать/заменить любые варианты, которыми можно создать web\_reg\_save\_param.*

**5) Главное окно.**

*Можно вообще не использовать это окно.*

*Используется, как окно для установки основных настроек, и дальнейшей работы в дочернем action-окне. Либо, для получения web\_reg\_save\_param в текстовом виде, с возможностью детальной настройки правил его создания – задача 1), 2).*

Описание работы с окном, есть во всплывающих подсказках.

1. ввести {param} имя*, для поиска его в Response файлах:*

Нажать поиск.

1. результат: создание и показ web\_reg\_save\_param строки*.*



Копировать результат.

1. Возможность редактирования и авто/настройки *всего, что так или иначе задействовано утилитой, при создании web\_reg\_save\_param.*

*Настроить виджеты модификаторы LB/RB.*

1. Возможность выбора [(3), (4)] всех вариантов для всех возможных inf/Response*.*



*Выбрать файл из (3), или Номер вхождения (4).*

**Термины***:*

*param - (1) искомый параметр из LoadRunner, для поиска(2) его в файлах LoadRunner-скрипта*

*"номер вхождения param" - число, меньше или равное числу, сколько раз param(1) встретился в файле(3)*

*LB - (5) определенное кол-во символов слева от ...param в файле(3)*

*RB - (5) определенное кол-во символов справа от param... в файле(3)*

*Ord - номер вхождения строк LB/RB(5) в файл(3), при котором между ними будет искомый param(1)*

*web\_reg\_save\_param - (6) итововый результат для вставки в LoadRunner-тест*

**Основные элементы управления***:*

*(1) - поле ввода param*

*(2) - кнопка поиска param*

*(3) - комбобокс выбора файла c param*

*(4) - комбобокс выбора номера вхождения param в файл*

*(5) - поля для вывода/редактирования LB/RB*

*(6) - кнопка получения web\_reg\_save\_param*

**Окно предназначено для***:*

1. *Поиска param в файлах LoadRunner-скрипта: шаг ввод(1)-поиск(2)*
2. *Определения места положения(номера вхождения) param в файле: шаг комбобокс(3)-комбобокс(4)*
3. *Определения и изменения LB/RB: шаг редактирование(5)*
4. *Определения Ord с учетом редактирования LB/RB(5): шаг редактирование(5)-формирование(6)*
5. *Формирования LoadRunner web\_reg\_save\_param с экранированием символов: шаг формирование(6)*

**Описание***:*

1. *Результат поиска(2) используя param(1):*

*Формируется список файлов, в которых param(1) был найден.*

1. *Список найденых файлов записывается в комбобокс(3), выбирается один из файлов(3) (чекбокс "reverse"), происходит чтение файла(3).*
2. *Кол-во вхождений param в файл, записывается в комбобокс(4) в виде списка [0, 1, n], выбирается одно из вхождений(4).*
3. *В полях LB/RB(5) отображается текст из "частей", слева/справа от номера вхождения(4) param(1), в файл(3) ), с учетом всех виджетов модификаторов LB/RB.*
4. *Пользователь меняет(при необходимости):*
5. *Файл(3) - происходит заполнение комбобокса(4).*
6. *Вхождение(4) - происходит подтягивание LB/RB(5), с учетом всех виджетов модификаторов LB/RB.*
7. *Редактирует поля LB/RB(5) - т.к. в поле может попасть "вариативный" параметр, нежелательные спец символы, и т.д. Использует виджеты модификаторы.*
8. *При формировании web\_reg\_save\_param(6) результата:*

*Текст файла(3) "разбивается на части", по LB(5)*

*В списке "частей", происходит поиск RB(5)*

*Если "часть" содержит RB(5) - увеличить Ord*

*Если "часть" начинается на 'param(1) + RB(5)' - Ord найден*

*Будут экранированы символы: '\\', '"'*

Работа **без чекбокса auto**(OFF) (*ON = шаги (3)-(6) выполняются автоматически, после поиска(2).*)*:*

*(1) Ввести искомый параметр в "поле ввода {param}".*

*(2) Нажать кнопку "поиск {param} файлов".*

 *a) "список файлов", содержащих param, записан в комбобокс(3).*

*в комбобокс(3) выбран "файл" из списка.*

*b) "список номеров", вхождения param в файл(3), записан в комбобокс(4).*

*в комбобокс(4) выбрано "вхождение".*

*c) в полях LB/RB(5) будет отображен текст, с учетом (1)-(5).*

*(3) При необходимости выбрать из комбобокс(3) другой "файл с param".*

*переход в 2.b)-2.c)*

*(4) При необходимости выбрать из комбобокс(4) другое "вхождение param".*

*переход в 2.c)*

*(5) При необходимости редактировать "поля LB/RB"*

*(6) Нажать кнопку "web\_reg\_save\_param".*

*🡪 Вывод web\_reg\_save\_param с учетом (3)-(5) 🡨*

**6) Action.c окно.**

*Все что связано с action.c находится здесь.*

*В работе использует настройки, выставленные в главном окне.*

*Можно открыть несколько, но полноценно работает только с первым.*

### **6.1.1)** **поиск и параметризация всех {param} в action.c**.

Кнопка “Запуск” – старт автоматической параметризации.

Можно вкл./выкл. любой из 6 методов поиска {param} для параметризации, если какие-то из них приносят много мусора – не{param} слов, при необходимости, удалить из результатов поиска, не{param} слова.

Можно выбрать место поиска {param} Request/Response/Action/... файлы.

Кнопка “настраиваемый поиск и создание {param}” – полу автоматическая параметризация, дает возможность настроить метод поиска, и создать web\_reg\_save\_param.

Кнопка “Предпросмотр” – покажет, какие {param} найдет этот метод поиска.

### **6.1.2)** Запуск параметризации конкретным методом



В целях экономии времени, либо если какие-то методы поиска {param} выдают много мусора, можно искать каким-то одним конкретным методом 1) – 6), который даст хороший результат. Тоже можно сделать и из общего меню “Найти и Создать WRSP”.

### **6.1.3) Найти все варианты, создания {web\_reg\_save\_param}, для выделенного param**.

Выделить имя param целиком (“bJsPnh”).

Оставить только один/несколько вариантов создания web\_reg\_save\_param.

### **6.1.4)** **Найти все “одноименные” {param} – имя которых начинается на выделенный текст**.

Выделить часть имени param:

“bJs” для “bJsPc0”.

*“****bJs****” 🡪 [“****bJsPxe****”, “****bJsPg3****”, …]*

Удалить из диалог окна “не {param}” слова.

### **6.1.5)** **Найти все {param}, имеющие выделенный LB=**.

Выделить param LB:

“value=” для “ value=bJsPnh”.

*“****value=****” 🡪 [“****bJsPkh****”, “****bJsP2h****”, …]*

Удалить из диалог окна “не {param}” слова.

### **6.1.6)** **создать {web\_reg\_save\_param}, для выделенного param**.

Выделить имя param целиком “bJsPk0”.

### **6.2) Редактирование LB/RB в уже сформированном web\_reg\_save\_param и пересчет Ord=**.

Например, web\_reg\_save\_param уже сформирован и находится в action.c, но в “LB=” попал другой ID, и мы ходим удалить из wrsp этот ID, и пересчитать новый “Ord=”.

Сначала раз/экранируем и редактируем LB/RB, затем выделяем весь блок текста param, целиком с комментариями, и выбираем в меню указанный пункт.

Для восстановления “оригинальных”LB/RB – выбрать “с оригинальными”.

### **6.3.1) Удаление из action.c, уже сформированного web\_reg\_save\_param**.

Выделить целиком имя param или имя web\_reg\_save\_param.

Объект web\_reg\_save\_param будет удален, все {web\_reg\_save\_param} в action.c, будут заменены на оригинальное имя param.

Может корректно удалить, только те web\_reg\_save\_param, которые содержат в своих комментариях, оригинальное имя param, по которому они созданы.

### **6.3.2) Удаление из action.c, всех сформированных web\_reg\_save\_param**.

Удалит любые, созданные на текущий момент, {web\_reg\_save\_param}, вместо них, вернет оригинальные имена param.

Может корректно удалить, только те web\_reg\_save\_param, которые содержат в своих комментариях, оригинальное имя param, по которому они созданы.

### **6.3.3)** удалить web\_ объекты

Способ №1: несколько шаблонов, за пускающихся по очереди - теоретически можно изменить шаблон, для удаления любых web\_.

Шаблоны: удаление “dummy” web\_.

какой-то “старый способ” удаления.



Способ №2: один шаблон, и пользователь сам может прописать шаблон для поиска и удаления любых web\_.

Шаблон: удаление “dummy” web\_.

### **6.3.4)** удалить lr\_think\_time объекты



После записи теста, остаются lr\_think\_time объекты, в случайных местах. Их надо удалить. После параметризации скрипта, расставить их вручную.

### **6.4.1)** переименовать транзакции

Обычно, при записи скрипта, названия транзакций сокращают. Бывает, что, из-за кодировки системы, LoadRunner сохраняет русские имена, как “???”.

Поэтому после, транзакции переименовывают.

Виджет выведет названия всех транзакций, в два столбца.

В первом столбце – не трогать (это пример оригинального названия).

Во втором столбце – прописать новое имя транзакции (или оставить старым).

*Переименует lr\_start\_transaction() и lr\_end\_transaction().*

### **6.4.2)** переименовать (+) web\_reg\_save\_param

Способ №1: Автоматический. web\_reg\_save\_param могут быть созданы LoadRunner или пользователем, в следствие чего они будут иметь бессмысленные имена.

Виджет выведет названия всех web\_reg\_save\_param, в два столбца.

В первом столбце – не трогать (это пример оригинального названия).

Во втором столбце – автоматически сформирует и выведет новое “осмысленное” имя WRSP (можно поменять).

### **6.4.3)** переименовать (-) web\_reg\_save\_param

Способ №2: Ручной. web\_reg\_save\_param могут быть созданы LoadRunner или пользователем, в следствие чего они будут иметь бессмысленные имена.

Виджет выведет названия всех web\_reg\_save\_param, в два столбца.

В первом столбце – не трогать (это пример оригинального названия).

Во втором столбце – поменять или оставить прежним.

## **6.5)** **Декодирование выделенного фрагмента**.

Выделить текст.

Выбирать тип декодирования, пока он не расшифруется.

Нажать “Заменить”, при необходимости.

## **6.6)** Настройка информационных комментариев и имени web\_reg\_save\_param

Анализировать action web\_ объекты на наличие WARGING:

неправильное использование web\_reg\_save\_param, наличие русских символов для перекодировки, собрать статистику объектов, какие именно в них используются {param}, сколько раз, где внутри action кода находится сам объект, внутри какой транзакции, какой по счету, и тому подобное.

Все описанное вывести, в комментарии “//lr:” к объекту в action.c тексте, и на виджеты.

**

## **6.7)** настройка общих переменных

Утилита использует множество переменных, которые возможно хотелось бы настраивать прямо из нее “на лету”. На всякий случай, в меню выводятся все переменные из файлов настроек, с возможностью их изменения, и показом подсказок, из комментариев исходного кода. Но на самом деле это не очень полезный функционал.

## **6.8.1)** Показ дополнительных управляющих виджетов

Показать/скрыть виджеты внизу окна.

Почти все, что там есть, можно вызвать из меню и описывается в документе. Остальное там – интуитивно понятно/устарело/не особо важно.

## **6.8.2)** Показ дополнительных навигационных виджетов

Показать/скрыть виджеты внизу окна.

При выборе значения из “комбо боксов” виджетов – происходит переход в область action.c текста.

## **6.9)** Показ сводной информации

Вариант №1

Краткий отчет о внутреннем представлении action.c.

Вариант №2

Расширенный отчет о внутреннем представлении action.c.

## **6.10)** открытие/сохранение action.c

Текст action.c хранится в двух видах:

1. внутреннее представление
2. Текст на экране

Утилита работает с “внутренним представлением”, а “Текст на экране” только отображает его актуальное состояние.

Это значит, что если пользователь внес изменения в “Текст на экране”, то он должен перенести эти изменения во “внутреннее представление”, одним из способов:

* + - сохранение на диск: “Save”.
    - сохранение без диска: “Перенести текст на экране, во внутр. представление”
    - сохранение без диска: в меню правой кнопки мыши: “Сохр. пользоват. изменения в тексте”.

## **6.11)** Просмотр Response файлов ответов, при записи и воспроизведении



Можно увидеть список всех Response файлов нагрузочного скрипта, как при записи, так и при воспроизведении (необходимо хотя бы раз запустить скрипт). Открыть их на просмотр, увидеть, как они сгруппированы в inf файлы.

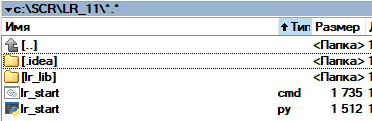
 Подряд: только при записи по Snapshot: при записи и воспроизведении

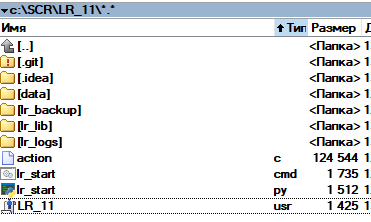
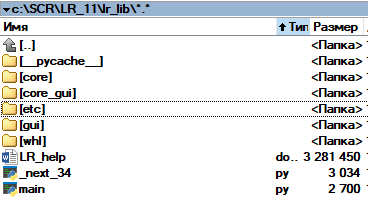
## 6.12) Отображение схемы распределения web\_reg\_save\_param

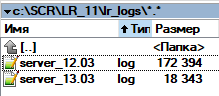
Сверху и снизу – snapshot web\_ action объекты. Стрелками показаны web\_reg\_save\_param: выходят ОТТУДА, где создаются, приходят туда, где используются. Нажатие – переход в область текста.

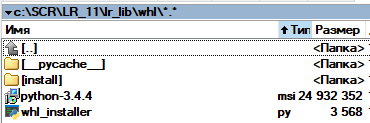
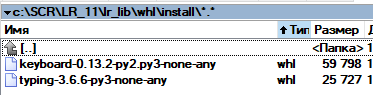
Если линии идут слева – направо, значит web\_reg\_save\_param неправильно используется: т.е. создается позже, чем его пытается использовать какая-то web\_ в своем теле.

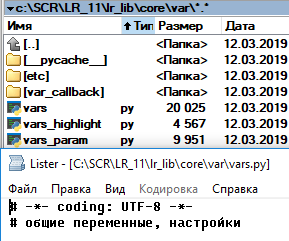


**Приложение: Файлы утилиты**.









**Для облегчения веса**, можно удалить .**msi** файл и .**docx** файлы ~ 28Mb, и .**pyc**, если они есть.