**“LR\_PARAM” v11.6.5 – утилита для работы с LoadRunner скриптом.**

*author: vgonchar\_ext*

*версия файла: “1.1”* ***/ 12.03.19***

Оглавление

[1) Описание 3](#_Toc3233554)

[*1.1)* *Общее описание задач, решаемых утилитой* 3](#_Toc3233555)

[*1.2)* *Теория LoadRunner параметризации, в контексте утилиты* 5](#_Toc3233556)

[*1)* *Выводы* 8](#_Toc3233557)

[2) Общее описание 9](#_Toc3233558)

[поиск и параметризация всех {param} в action.c 11](#_Toc3233559)

[Запуск параметризации конкретным методом 11](#_Toc3233560)

[Найти все варианты, создания {web\_reg\_save\_param}, для выделенного param 12](#_Toc3233561)

[Найти все “одноименные” {param} – имя которых начинается на выделенный текст 12](#_Toc3233562)

[Найти все {param}, имеющие выделенный LB= 12](#_Toc3233563)

[создать {web\_reg\_save\_param}, для выделенного param 12](#_Toc3233564)

[Редактирование LB/RB в уже сформированном web\_reg\_save\_param и пересчет Ord= 13](#_Toc3233565)

[Удаление из action.c уже сформированного web\_reg\_save\_param 13](#_Toc3233566)

[удалить web\_ объекты 13](#_Toc3233567)

[удалить lr\_think\_time объекты 14](#_Toc3233568)

[удалить web\_reg\_save\_param объекты 14](#_Toc3233569)

[переименовать транзакции 14](#_Toc3233570)

[переименовать (+) web\_reg\_save\_param 15](#_Toc3233571)

[переименовать (-) web\_reg\_save\_param 15](#_Toc3233572)

[Декодирование выделенного фрагмента 15](#_Toc3233573)

[Настройка информационных комментариев и имени web\_reg\_save\_param 16](#_Toc3233574)

[настройка общих переменных 16](#_Toc3233575)

[Показ дополнительных управляющих виджетов 16](#_Toc3233576)

[Показ дополнительных навигационных виджетов 17](#_Toc3233577)

[Показ сводной информации 17](#_Toc3233578)

[открытие/сохранение action.c 17](#_Toc3233579)

[Просмотр Response файлов ответов, при записи и воспроизведении 18](#_Toc3233580)

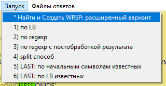
# Описание

*Утилита не призвана заменить LoadRunner. Это “инструмент”, который помогает выполнять задачи, решаемые LoadRunner недостаточно хорошо, либо не решаемые им вовсе. Она предоставляет более настраиваемые/гибкие инструменты, и иногда какую-то, возможно не лучшую, альтернативу, для решения многих задач.*

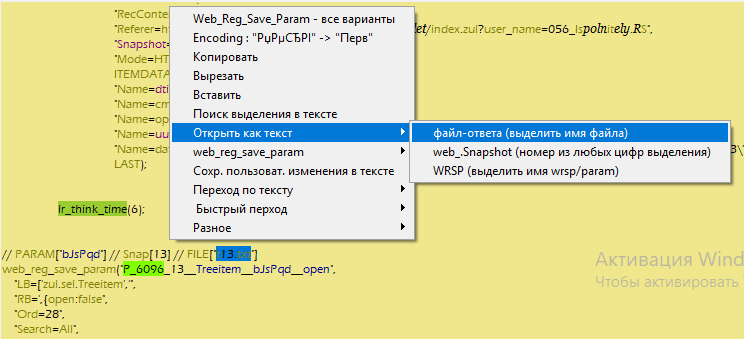
* 1. Общее описание задач, решаемых утилитой:

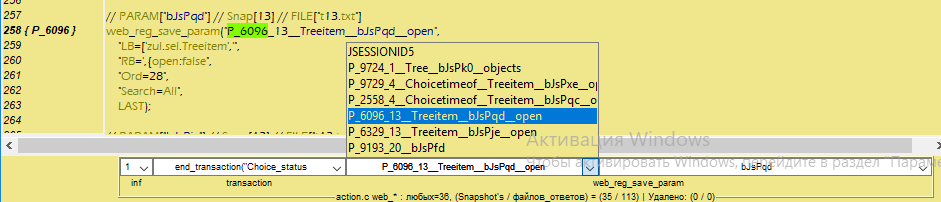
1. *Поиск {param} имен в action.c, для дальнейшей их параметризации.*

*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. Вероятность 100% готовности есть, но небольшая, т.к. иногда некоторые {param} не могут быть созданы LB/RB-способом и необходим regexp-способ, либо потребуется изменение каких-то настроек, то есть потребуется полуавтоматический/итерационный/ручной подход.*

1. *Еще до формирования web\_reg\_save\_param есть возможность: редактирования и выбора всех вариантов LB/RB, настройки правил создания. Автоматическое вычисление Ord=. Фактически это 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.*
3. *Осуществлять навигацию по action.c, переходы между объектами, транзакциями и т.д.*

**

**

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…*
  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 видефайловойструктуры в подкаталог “***data***”:



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 – с*вязь Request/Response, с web\_**объектами внутри “.c”:*

“*t1.inf*”, “*t12.inf*”

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

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

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

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

*Файлы* ***inf*:**



*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*

*action\_two.c # тело2 теста*

*vuser\_end.c # логаут*

*action.c LoadRunner теста, 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 разных типов.*

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)****.*

*web\_reg\_save\_param вытаскивает новое id/имя-переменной {param}, при воспроизведении/итерации скрипта.*

*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}. Переопределение может носить непостоянный характер.*

1. ***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 **запуск**у.
2. Выводы**.**

*Вероятность параметризации action.c, в автоматическом режиме с первого раза, до состояния 100% “готового” скрипта – небольшая.*

2.1) Можно выстроить цепочку:

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

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

2.2) Можно выделить проблемы:

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”.*
6. ******Общее описание:**
7. **слева**: главное окно – *настройка и* ***ручное*** *создание web\_reg\_save\_param*.

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

Отвечает за **проблему**: **№** **4**.

Частично отвечает за **проблему**: **№** **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 **проблему**: **№** **1, 5**.

Отвечает за **проблему**: **№** **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 **проблему**: **№** **3**:

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

**3.1) Главное окно:**

*Можно вообще не использовать это окно.*

*Используется, как окно основных настроек, для 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)****:*

1. **ввести** *{param} имя, для поиска его в Response файлах:*
2. ***результат****: создание и показ web\_reg\_save\_param строки.*



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

**

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

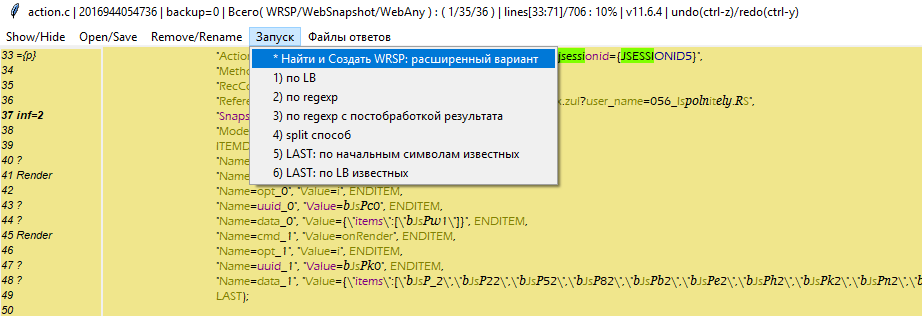


**3.2) Окно action:**

*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.*

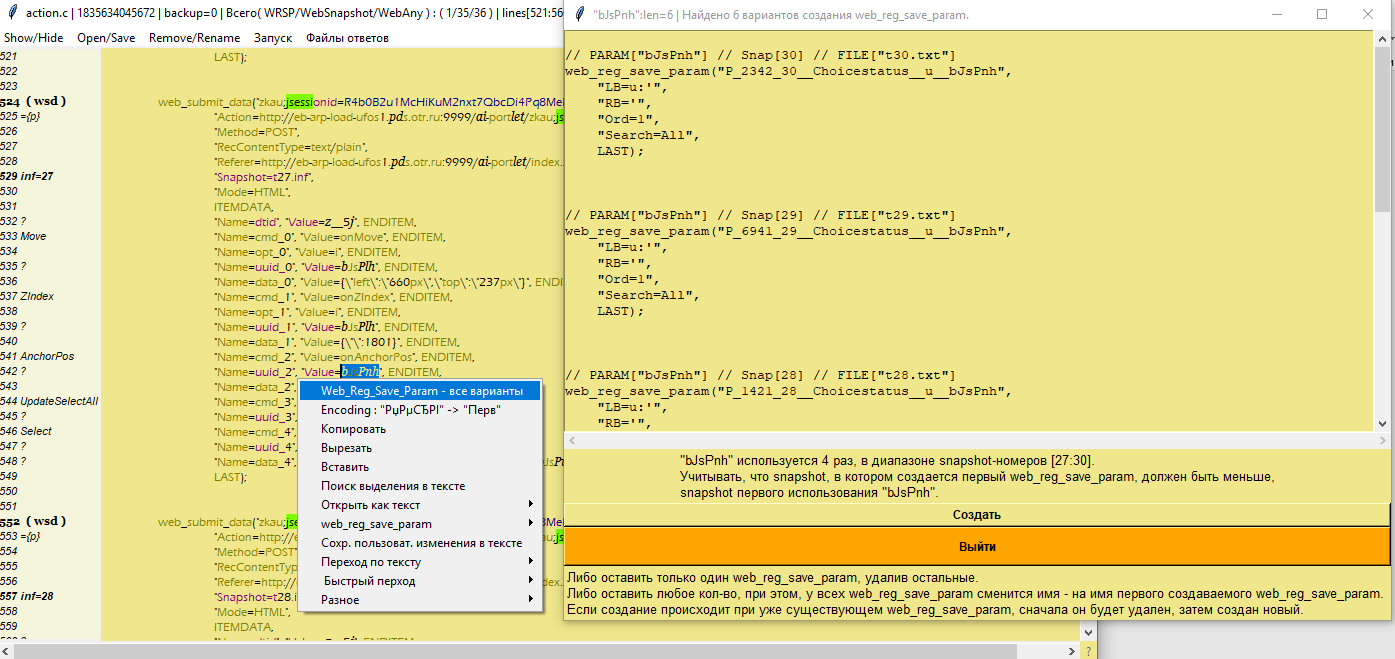
**поиск и параметризация всех {param} в action.c**.

При необходимости, удалить из результатов поиска, “не {param}” слова.

## Запуск параметризации конкретным методом

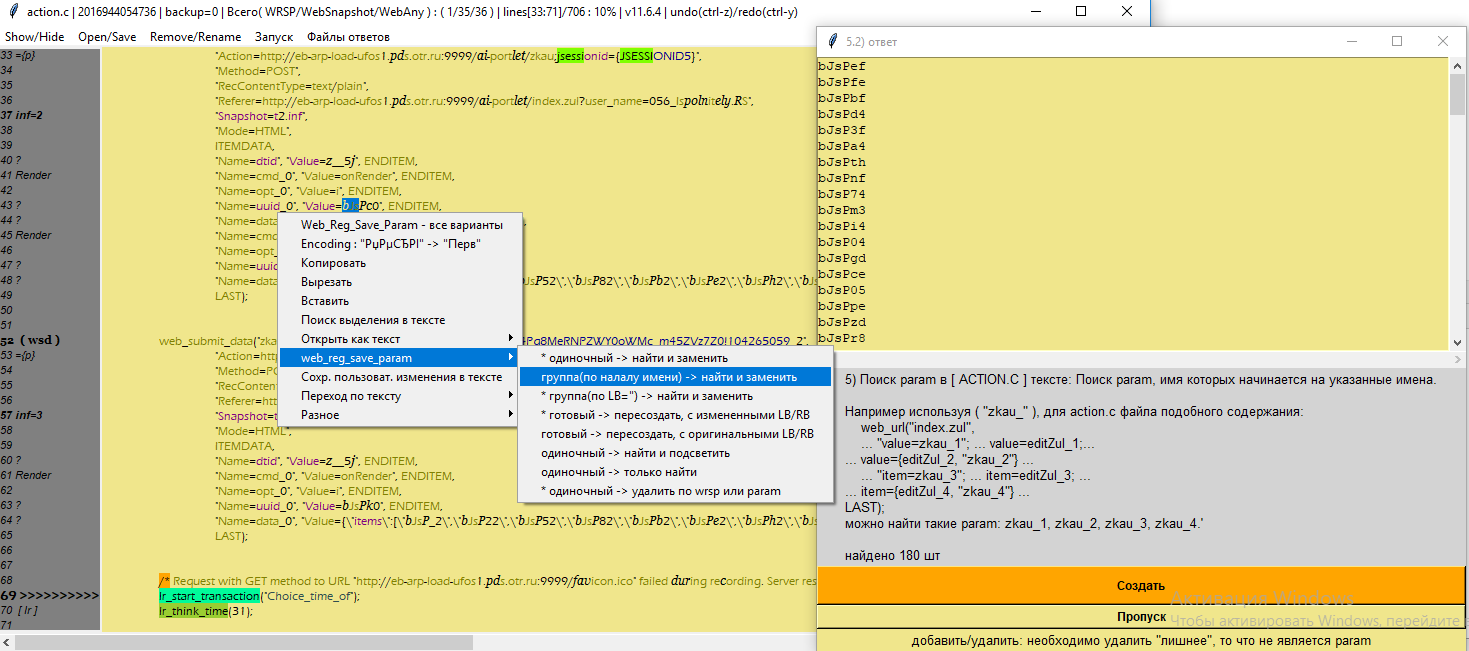


В целях экономии времени, либо если какие-то методы поиска {param} выдают много мусора, можно искать каким-то одним конкретным методом 1) – 6), который даст хороший результат. Тоже можно сделать и из общего меню “Найти и Создать WRSP”.

**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.

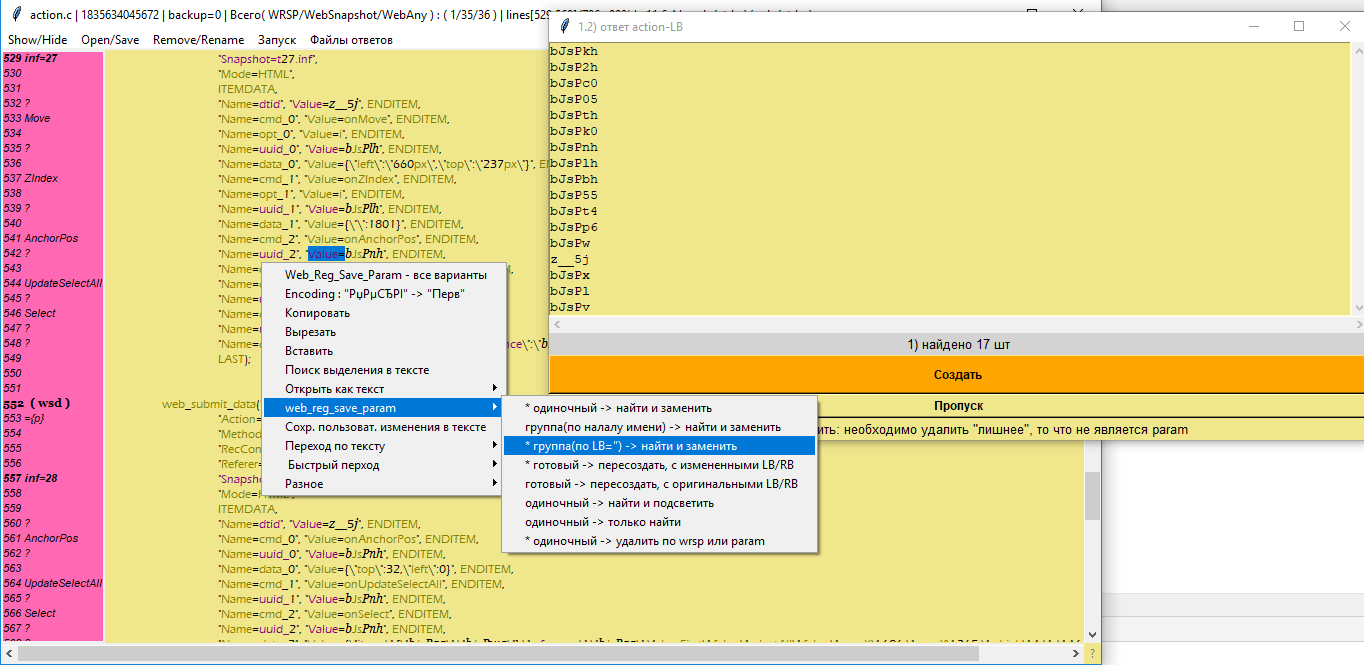
**Найти все “одноименные” {param} – имя которых начинается на выделенный текст**.

Выделить часть имени param:

“bJs” для “bJsPc0”.

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

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

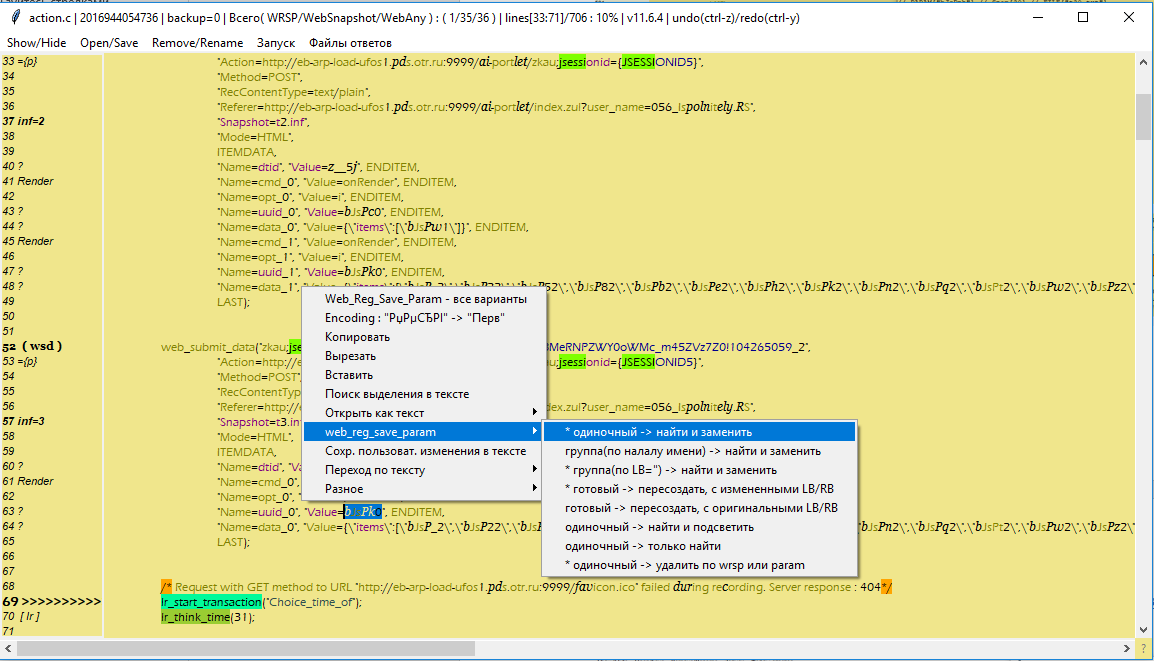
**Найти все {param}, имеющие выделенный LB=**.

Выделить param LB:

“value=” для “bJsPnh”.

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

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

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

Выделить имя param целиком “bJsPk0”.

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

Сначала раз/экранируем и редактируем LB/RB, затем выделяем весь блок текста param целиком.

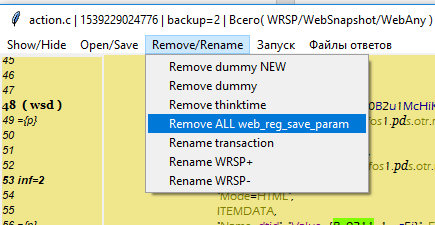
Для восстановления оригинальных LB/RB – выбрать “с оригинальными”.

**Удаление из action.c уже сформированного web\_reg\_save\_param**.

Выделить целиком имя param или 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.

Или можно удалить все {web\_reg\_save\_param}:



## удалить web\_ объекты

Способ №1: несколько шаблонов для поиска и удаления “dummy” web\_



Способ №2: пользователь сам прописывает шаблон для поиска и удаления web\_

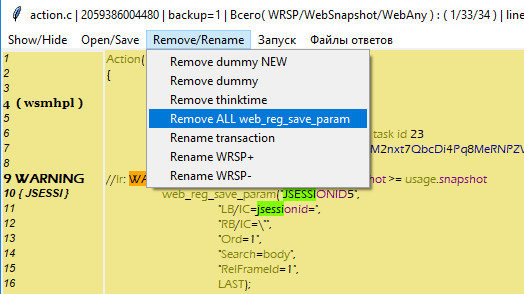
## удалить lr\_think\_time объекты



После записи теста, остаются ненужные lr\_think\_time объекты.

Их надо удалить, а после параметризации скрипта, расставить их вручную.

## удалить web\_reg\_save\_param объекты



Удалит любые созданные на текущий момент web\_reg\_save\_param, и вернет оригинальные названия param.

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

## переименовать транзакции

Обычно, при записи скрипта названия транзакций сокращают, и уже после их 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 / lr\_end\_transaction.

## 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).

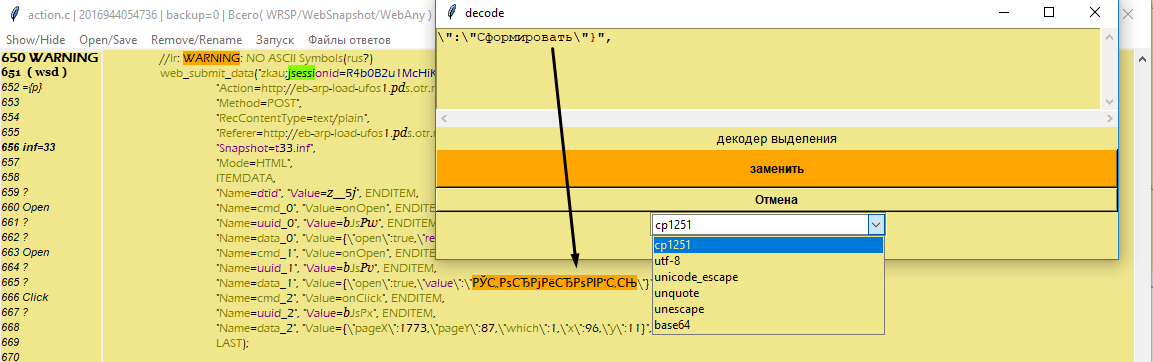
## переименовать (-) web\_reg\_save\_param

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.

******Декодирование выделенного фрагмента**.

Выделить текст.

## Настройка информационных комментариев и имени web\_reg\_save\_param

Анализировать action объекты на наличие WARGING:

неправильное использование web\_reg\_save\_param, наличие русских символов для перекодировки, 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.

## 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. Но на самом деле это не очень полезный функционал

## Показ дополнительных управляющих виджетов

Показать/скрыть виджеты внизу окна.

Почти все что там есть, можно вызвать из меню, и описывается дальше. Остальное там – не особо важно.

## Показ дополнительных навигационных виджетов

Показать/скрыть виджеты внизу окна.

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

## Показ сводной информации

Вариант №1

Вариант №2

## 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”.

## 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, посмотреть, как они сгруппированы в inf файлы.

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