Access Manager Load Testing Best Practices

This guide describes best practices while performing load testing by using Micro Focus LoadRunner. An effective load test can be achieved through the following criteria:

- Understand the exact requirement
- Determine the number of VUsers required to simulate the load
- Follow the tuning guidelines as explained in the *NetIQ Access Manager Performance and Sizing Guidelines*
- Design a proper LoadRunner scenario
- Deploy an effective test monitoring frame work

By following some of the best practices explained in this document, you can conduct a realistic load test on NetIQ Access Manager before deploying it in the production environment. In an actual customer environment, there might be other factors that need to be considered for the load test.

NOTE: For information about tuning each component of Access Manager better performance, see the *NetIQ Access Manager Performance and Sizing Guidelines* .

A typical Access Manager implementation has a number of interconnected systems: Administration Console, Access Gateway, Identity Server, web servers, enterprise networks, LDAP user stores, L4 switches, and client machines. Any of these components can impact the system performance.

This guide is divided into the following three sections to describe phases of the load testing:

1. Test Preparation:
   - Identifying the objective
   - Understanding the use case
   - Planning the test infrastructure
   - Deploying and fine tuning the test infrastructure
2. Test Development: Generating a test script for simulating the exact business use case.
3. Test Execution And Analysis:
   - Generating test scenarios
   - The iterative method of execute, analyze, and debug mode of test execution for better load testing

1 Test Preparation

Test preparation includes the following activities:

- Identifying the test objective
- Understanding the use case
- Planning the test infrastructure
- Deploying and fine tuning the test infrastructure

1.1 Identifying the Test Objective

Type of Test	Objective
Pre-production performance testing	Helps you test performance of Access Manager under user load conditions. The objective is to determine if Access Manager can sustain the requested number of users with acceptable response times.
Endurance testing	Validates the stability and reliability of Access Manager by applying a load for an extended period of time.
Sizing recommendation/ capacity planning	Determines the maximum number of users that can use Access Manager simultaneously before experiencing system failure.

1.2 Understanding the Use Case

Understanding the use case includes the following activities:

- **Identifying the type of a user request** A user can perform more than one operations as part of requests. Identifying the type of user requests helps in designing scripts that can be used by the virtual users.
- **Identifying and estimating the number of concurrent users that can use Access Manager** Helps to design the number of users to be run in each test.
- **Identifying the real user behavior** Helps to design the virtual user ramp up during the load test. For example, there can be users that login and remain connected for some time before logging out. Whereas there can be users that login and after accessing few files, go to idle state and never logout.

1.3 Planning the Test Infrastructure

After arriving at the estimated number of concurrent users, you can design the infrastructure required for the load test based on the recommendations in the performance and sizing guidelines. Ensure that the network devices and the back end web servers used for the tests do not have any limitation.

For information about the hardware platform requirements for setting up the Access Manager infrastructure, see Performance and Sizing Guidelines.

1.4 Deploying and Fine Tuning the Test Infrastructure

After the test infrastructure is ready, tune the various components of the infrastructure to get the optimum and expected performance.

For the tuning recommendations, see Performance and Sizing Guidelines.

2 Test Development

The test development involves the following phases:

- Script generation
- Identifying the transactions
- Parameterization
- Run-time parameters

2.1 Script Generation

LoadRunner VuGen is used to capture user behaviors. During recording, ensure that you perform the same task as a real user does in the production environment.

You can record different scripts for each operation and scenario. For example, you can create separate scripts to simulate a user trying to access the following resources:

- A public web page through Access Manager
- A protected web page

2.2 Identifying the Transactions

After the basic scripts are ready, you can group the individual operations into different LoadRunner transactions. This helps to measure the time taken to complete that transaction during the load testing. A transaction can be a single operation by an end user or a series of operations such as authenticating, accessing the page, and a logout based on your test requirement.

2.3 Parameterization

To accurately simulate the real-user behavior in a load test, it is important to parameterize the user inputs. This helps in making the test inputs random during the load test.

During the load test, you can parameterize login users, web pages, and so forth.

2.4 Run Time Parameters

Consider the following run-time parameters while creating scripts:

Parameter	Purpose
Think Time	To introduce an appropriate delay between each operation. This parameter can be used to simulate the idle time a user spends on a particular web page.
Browser Emulation	To emulate the type of the web browser an end user uses.
Browser Caching	To simulate the browser caching based on the web server behavior.
IP Spoofing	To make each VUser send requests with different source IP addresses by enabling IP Spoofing during the load test. This ensures that a L4 switch distributes load to each node in a correct manner.
Connection Speed	To simulate a situation where a user connects to the server with different network speed.

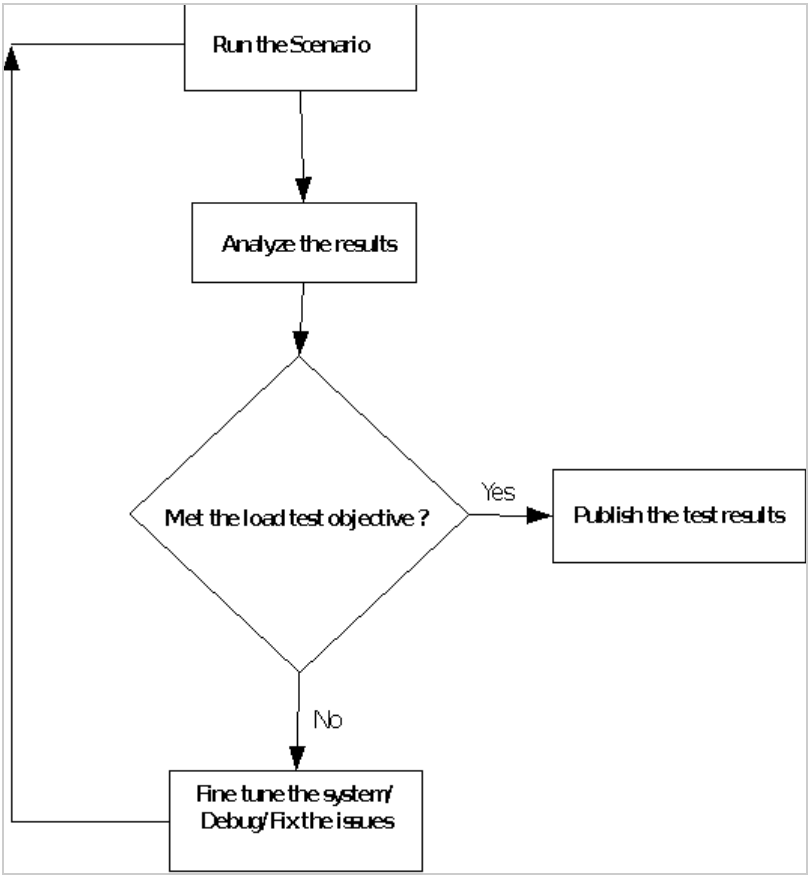
NOTE: If the backend web server uses Authentications, Formfill, or Identity Injection, the LoadRunner test design does not change. The test design changes only when the user behavior changes.

3 Test Execution And Analysis

The test execution approach is as follows:

- **Component-wise load testing:** Each integrated Web portal is performance tested with Access Manager. The objective of this approach is to ensure performance of the application and to determine the optimal configuration for the end-to-end integrated performance test.
- **End-to-end integrated performance test:** All applications are integrated during the end-to-end load test to simulate a production behavior.

The test execution and analysis is an iterative process and is executed in the following order:



4 Monitoring The Test Infrastructure

This phase includes monitoring the following components of the test infrastructure:

- Identity Server and ESP
- Access Gateway
- Back-end web servers

- LDAP user stores
- L4 switch

4.1 Identity Server and ESP

- On Identity Server and ESP, enable the statistics logging and monitor the *catalina.out* file to watch for sessions, number of threads, memory consumed, and so forth. Ensure that ramp up of users are tuned so that total number of sessions with any scenario does not cross the acceptable limit per setup.
- The Administration Console statistics page also provides useful Identity Server/ESP statistics.
- Use *'top'* and *'ps eIF'* commands to see the systems CPU and memory utilization and subsequently use these to fine-tune tests and test infrastructure.
- Use the *netstat* command to capture the connections build up to the server. For example, netstat -natpl | grep 443 | grep ESTAB | wc -l

4.2 Access Gateway

- The apache server_status page (http://127.0.0.1:8181/server-status) can be used to monitor the requests per sec, apache thread status, current load, CPU usage, and so forth.
- U*se 'top'* and *'ps eIF*' commands to see the systems CPU and memory utilization.
- Use the *netstat* command to capture the connections build up to the server. For example, netstat -natpl | grep 80 | grep ESTAB | wc -l

4.3 Backend Web Servers

Monitor the backend web servers to ensure that that these do not affect the performance during the load test.

4.4 LDAP Use Stores

Monitor the LDAP users stores to ensure that these do not affect the performance during the load test.

4.5 L4 Switches

- Monitor the L4 switch to see if the load is properly shared across all nodes based on the load balancing algorithm selected in the L4 switch.
- Monitor the session stickiness in the L4 switch. If the session is not sticking correctly, it results in a number of proxy requests that impacts the performance.

5 Use Cases

Let's consider an example of a typical organization that has the following web resource protected by Access Manager. This use case is designed after studying multiple customers and their use cases.

- Web Applications/Portal: mycompany.com protecting
- email Web Access - myMail
- Bug tracking tool - myBugzilla
- Web Based Social networking -myTeaming

- Employee Self Service Portal – myInnerweb
- Company Intranet Site – myIntranet

The following usage pattern is used as a reference throughout in this document. It is important to come up with a table of data like this before designing the load tests. Use the historical data if available. If the historical data is not available, prepare the table based on certain assumptions and future needs.

Portal	Page visits per day	Average Page Views per hour	Peak page visit per hour	Users login per day	Average Users Per hour	Peak Users in a hour	Users Average logged in time (min)	Number of pages viewed during the logged in period
MyInnerweb	6,720,000	280,000	400,000	1,680,000	70,000	100,000	60	4
myBugzilla	4,800,000	200,000	2,80,000	720,000	30,000	40,000	120	7
myTeaming:	1,440,000	60,000	80,000	408,000	17,000	20,000	320	4
myMail	1,080,000	45,000	60,000	360,000	15,000	20,000	10	3
myIntranet	6,000,000	2, 250,000	400,000	720,000	30,000	40,000	50	10

In this case study, only the test design for *myInnerweb* is explained. Using the same method, you can design the load testing for other applications also. The objective of the load tests is to meet the tabular data made for myInnerweb application. This objective can be broken down into multiple objectives:

- Login test to find highest new user logins
- Application access tests to find highest number of requests made by existing sessions
- Endurance test to see the system can withstand the load for a longer duration

NOTE: Tests may vary based on the test objective and projected deployment.

5.1 Login Test to Find Highest Number of Users Logins

The objective of this test is to achieve the highest number of logins that Access Manager can support. In this example, the highest users' logins per hour of the day is 100,000. Tests should be first designed to achieve the expected numbers and then another round of test must be done to see the limit of the system by increasing the expected number by approximately 25%. The second test will ensure that the system can support a sudden surge in the user logins or user requests.

Out of these 100,000 logins, let us consider the following pattern of user behavior:

1. Users who login and remain idle for the rest of the day - 50,000
2. Users who login, access some basic test page, and log out – 25,000
3. Users who login and generate additional requests for more pages – 25,000

This can be achieved by using the following three scripts:

- login_keep_idle_script
- login_access_logout_script
- login_access_multiple_pages_script

After the scripts are ready, arrive at total VUsers based on the following calculation:

Tuning: Set the session time out to 15 seconds in the Identity Server

Idle users – 50,000 (login_keep_idle_script)

Time taken for a single VUser login: 1 second

In 1 hour, single VUser can simulate (60 * 60)/1 = 3,600 users login

To reach 50,000 users login = 50,000/3,600 ~ 14 VUsers are required

Users who login, access some pages, and log out – 25, 000 users (login_access_logout_script)

1. Time taken for a single user to login: 1 second
2. Time taken for a user to access some page: 1 second
3. Time taken for a user to logout: 1 second

Time taken for a single VUser to complete the operations = [a] + [b] + [c] = 3 second

In 1 hour, the single users can simulate (60*60)/3 = 1,200 users transactions

To simulate 25,000 users transaction = 25,000/1,200 ~ 21 VUsers

Users who login and do some additional request -25,000 Users (login_access_multiple_pages_script)

1. Time taken for a single user to login: 1 second
2. Time taken for a user to access some more pages: 8 second
3. Time taken for a user to logout: 1 second

Time taken for a single VUser to complete the operations = [a] + [b] + [c] = 10 second

In 1 hour, the single users can simulate (60*60)/10 = 360 users transactions

To simulate 25,000 users transaction = 25,000/360 ~ 70 VUsers

To simulate a load test for this scenario for one hour, 105 VUsers (14 + 21 + 70) are required.

After identifying the scripts and the total VUsers, plan the LoadRunner test scenario for the same.

Create the LoadRunner scenario by including the above three scripts and assigning the number of VUsers for each scripts obtained by using the above calculations.

Script	Quantity(VUsers)
login_keep_idle_script	14
login_access_logout_script	21
login_access_multiple_pages_script	70

Schedule LoadRunner with the following parameters:

Parameter	Value
Start Group	Start immediately after the scenario begins
Initialize	Initialize each VUser just before it runs.
Start VUsers	105 VUsers, 1 VUser every 1 second
Duration	1 hour
Stop VUsers	Stop all VUsers simultaneously

These parameters allow to ramp up a VUser (1 VUser per second), run for one hour, and stop the VUser immediately after the test.

Run multiple iterations of the test scenarios to determine the exact ramp-up pattern in the following sequence:

1. Run the test
2. Analyze the logs and results to see whether the test objective is met
3. If no, fine tune the test scenario (ramp up, environment) and rerun the test
4. If yes, publish the test results

5.2 Application Access Tests: The Highest Number of Page Requests to an Application

The objective of this test is to find the highest number of requests Access Manager can handle from existing logged in sessions for a period of 1 hour. In a real life scenario, users request for pages with a delay in between each pages. While simulating this test, this can be achieved by using the following three scripts:

- **login_script:** to ensure all users are already logged in to Access Manager - 100,000 Users
- **login_and_request_script:** to simulate number of requests coming to Access Manager at any point of time from existing sessions – 400,000 page access per hour.
- **session_renewal_script:** to simulate sessions renewal under load for sessions spanning for more than the configured session timeout period

Ensure that all 100,000 users are logged in to the system (login_script)

Configure a protected resource – PR1 with an authentication contract of session time out of 2 Hours.

To calculate the number of VUsers required:

Time taken for one user to login: 1 second

In 1 hour a single VUser can simulate: 60 *60 = 3,600 Users login

To reach 100,000, we need: 100,000/3,600 = 28 VUsers

To reach the highest number of transaction of 400,000 page access per hour (login_and_request_script)

To calculate the number of VUsers required:

1. Time taken for one user to login: 1 Second
2. Time taken for one user to access 4 pages: 4 second
3. Time taken for the user to logout: 1 second

Time taken for 1 user to complete 1 transaction: [a] + [b] + [c] = 6 Seconds

In 1 hour a user can do: 60*60/6 = 600 transactions

To do 400,000 transactions per hour, we need: 400,000/600 = 667 VUsers

To achieve sessions' renewal under load (session_renewal_script)

Configure a protected resource PR2 with an authentication contract of session timeout of 5 minutes.

Time taken for a single login: 1 second

Session will be timed out after 5 minutes

In 1 hour: 60/5 = 12 sessions will be timed out

To achieve 240 sessions renewal per hour, 204/12 = 20 VUsers are required.

After identifying the scripts and the total VUsers, plan the LoadRunner test scenario.

Create two LoadRunner scenarios to achieve the test objective.

Scenario#1: To ensure 100,000 users logged in to the system

Script	Quantity
login_script	28

Schedule LoadRunner with the following parameters:

Parameter	Value
Start Group	Start immediately after the scenario begins
Initialize	Initialize each VUser just before it runs.
Start VUsers	28 VUsers, 1 VUser every 1 second
Duration	1 hour
Stop VUsers	Stop all VUsers simultaneously

These parameters will allow the ramp up of the VUser (1 VUser per second), run for one hour, and stop VUsers immediately after the test.

Scenario#2: To reach the highest number of transaction of 400,000 pages per hour and to have some session's renewal under load conditions

Script	Quantity
login_and_request_script	667
session_renewal_script	20

The LoadRunner scenario with the following parameters:

Parameter	Value
Start Group	Start immediately after the scenario begins
Initialize	Initialize each VUser just before it runs
Start VUsers	687 VUsers, 5 VUsers per second
Duration	1 hour
Stop VUsers	Stop all VUsers simultaneously

These above parameters will allow the ramp up of the VUsers (5 VUsers per second), run for one hour, and stop the VUsers immediately after the test.

NOTE: Scenario#2 has to be run immediately after Scenario#1 so that 100,000 users are logged in to the system during the second scenario.

5.3 Endurance Test

The objective of the test is to discover any performance issues in the system under test when subjected to the average load for a longer duration of time. This will test for any memory leaks in the application and whether the application is able to free up used heap memory when the test progresses.

In this use case, the following data are available for myInnerweb:

Average page views per hour: 280,000

Average users per hour: 70,000

The following scripts have been arrived upon for the endurance test:

- login_script: to simulate average users logins per hour to the system.
- transactions_script: to simulate average page views per hour to the system

After the scripts are ready, identify the number of VUsers required to run the endurance test.

login_script

Objective is to achieve 70,000 logins per hour.

Time taken for a single VUser login: 1 second

In 1 hour, single VUser can simulate (60 * 60)/1 = 3600 users Login

To reach 70,000 users login = 70,000/3,600 ~ 20 VUsers

transactions_script

Objective is to achieve 280,000 page views per hour.

1. Time taken for one user to login: 1 second
2. Time taken for one user to access 4 pages: 4 second
3. Time taken for the user to logout: 1 second

Time taken for 1 user to complete 1 transaction: [a] + [b] + [c] = 6 second

In 1 hour a user can do: 60*60/6 = 600 transactions

To do 280,000 transactions per hour, we need: 280,000/600 = 467 VUsers

After identifying the scripts and the total VUsers, plan the LoadRunner test scenario for the same.

Create the LoadRunner scenario for this by including the above two scripts and assigning the number of VUsers for each scripts obtained by using the above calculations.

Script	Quantity
login_script	20
transactions_script	467

Schedule the LoadRunner scenario with the following parameters:

Parameter	Value
Start Group	Start immediately after the scenario begins
Initialize	Initialize each VUser just before it runs
Start VUsers	487 VUsers, 5 VUsers every 1 second
Duration	8 hour
Stop VUsers	Stop all VUsers simultaneously

These parameters allow the ramp up of VUsers (5 VUsers per second), run the endurance test for eight hours, and stop the VUsers immediately after the test.

Endurance tests have to be run for sufficiently longer time while monitoring the following items:

- LoadRunner transactions per second: There should not be any decrease in the transactions per second over the period of time.
- LoadRunner failures in the transactions: There should not be any errors due to Access Manager component failures. If there are any such errors, analyze Access Manager log files to identify the cause of the error.
- Access Manager components (resource utilization): There should not be any increase in memory utilization, CPU utilization, and threads consumed by Access Manager components over the period of time.

Other parameters impacting endurance test:

- **Session time out:** Session time outs have to be properly configured for authentication contracts. Because endurance tests run for extended period of time, a misconfigured time out can lead to number of idle sessions and result in decreased performance of the system over the period of time.
- **Monitoring:** Automated monitoring mechanism can be used to monitor the systems behavior over the extended period of time. This will help to analyze the systems behavior (request per second, CPU usage, memory usage, and so forth) over the period of time. A basic shell script can also be used.

6 Appendix: Sample Scripts

6.1 Script 1

A user logs in to https://myinnerweb.mycompany.com/ and accesses eGuide (Employee Information Portal)

```
Action()
 {
 web_url("myinnerweb.mycompany.com",
 "URL=https://myinnerweb.mycompany.com/",
 "Resource=0",
 "RecContentType=text/html",
 "Referer=",
 "Snapshot=t1.inf",
 "Mode=HTML",
 LAST);
 lr_start_transaction("User Login");
 web_submit_form("sso",
 "Snapshot=t2.inf",
 ITEMDATA,
 EXTRARES,
 "Url=https://myinnerweb.mycompany.com/img/login/btn_password_blue.png",
"Referer=https://login.myinnerweb.mycompany.com/nidp/idff/
sso?id=4&sid=0&option=credential&sid=0", ENDITEM,
 "Url=https://myinnerweb.mycompany.com/img/login/btn_help_blue.png",
"Referer=https://login.myinnerweb.mycompany.com/nidp/idff/
sso?id=4&sid=0&option=credential&sid=0", ENDITEM,
 LAST);
 // lr_think_time(9);
 web_submit_form("sso_2",
 "Snapshot=t3.inf",
 ITEMDATA,
 "Access Manager=Ecom_User_ID", "Value={USER_Access Manager}", ENDITEM,
 "Access Manager=Ecom_Password", "Value=password", ENDITEM,
 "Access Manager=submit.x", "Value=47", ENDITEM,
 "Access Manager=submit.y", "Value=9", ENDITEM,
 LAST);
 web_url("sso_3",
 "URL=https://login.myinnerweb.mycompany.com/nidp/idff/sso?sid=0",
 "Resource=0",
 "RecContentType=text/html",
 "Referer=",
 "Snapshot=t4.inf",
 "Mode=HTML",
 EXTRARES,
 "Url=https://myinnerweb.mycompany.com/inc/metrics.js", "Referer=https://
myinnerweb.mycompany.com/", ENDITEM,
 "Url=https://myinnerweb.mycompany.com/favicon.ico", "Referer=", ENDITEM,
 LAST);
 lr_end_transaction("User Login", LR_AUTO);
 // lr_think_time(29);
 lr_start_transaction("Searching eGuide");
 web_url("eGuide",
 "URL=https://myinnerweb.mycompany.com/eGuide/servlet/
eGuide?val1=employee1&attr1=LastAccess Manager&doOnload=true",
 "Resource=0",
 "RecContentType=text/html",
```

```
 "Referer=",
 "Snapshot=t5.inf",
 "Mode=HTML",
 EXTRARES,
 "Url=/img/ni_h_leftnav_grayarrow.gif", "Referer=https://myinnerweb.mycompany.com/
eGuide/servlet/
eGuide?User.context=kwkwLnnhkmOq&Action=eGuideHeader&Search.rows=1", ENDITEM,
 "Url=/favicon.ico", "Referer=", ENDITEM,
 LAST);
 web_submit_form("eGuide_2",
 "Snapshot=t6.inf",
 ITEMDATA,
 "Access Manager=objectType", "Value=Company Employees", ENDITEM,
 "Access Manager=attr1", "Value=Last Access Manager", ENDITEM,
 "Access Manager=crit1", "Value=Starts with", ENDITEM,
 "Access Manager=val1", "Value={SEARCH_EMPLOYEE}", ENDITEM,
 LAST);
 lr_end_transaction("Searching eGuide", LR_AUTO);
 return 0;
 }
```

NOTE

- Two transactions are used for capturing the performance data: *User Login* and *Searching eGuide*.
- USER_AccessManager and SEARCH_EMPLOYEE input data are parameterized to simulate randomness in employee login and employee being searched.
- The lr_think_time function is commented to avoid idle time.

6.2 Script 2

A user accesses a public page without reusing the connection

```
Action()
 {
 web_url("1.test.html",
 "URL=https://mag.perftest.com/\{page_num}.test.html",
 "Resource=0",
 "RecContentType=text/html",
 "Referer=",
 "Snapshot=t1.inf",
 "Mode=HTML",
 EXTRARES,
 "Url=/img/bodybg.jpg", ENDITEM,
 "Url=/img/menuhover.jpg", ENDITEM,
 "Url=/img/footerbg.jpg", ENDITEM,
 "Url=/favicon.ico", "Referer=", ENDITEM,
 LAST);
 return 0;
 }
```

6.3 Script 3

A user accesses a public page by reusing the connection to create subsequent transactions

```
Action()
 {
 int i;
 lr_start_transaction("request");
 web_url("1.test.html",
 "URL=https://mag.perftest.com/1.test.html",
 "Resource=0",
 "RecContentType=text/html",
 "Referer=",
 "Snapshot=t1.inf",
 "Mode=HTML",
 EXTRARES,
 "Url=/img/bodybg.jpg", ENDITEM,
 "Url=/img/menuhover.jpg", ENDITEM,
 "Url=/img/footerbg.jpg", ENDITEM,
 "Url=/favicon.ico", "Referer=", ENDITEM,
 LAST);
 lr_end_transaction("request",LR_AUTO);
 //lr_think_time(36);
 for (i=1;i<100;i++) {
 lr_start_transaction("subsequent_request");
 web_url("2.test.html",
 "URL=https://mag.perftest.com/\{page_num}.test.html",
 "Resource=0",
 "RecContentType=text/html",
 "Referer=",
 "Snapshot=t2.inf",
 "Mode=HTML",
 LAST);
 lr_end_transaction("subsequent_request", LR_AUTO);
 }
 return 0;
 }
```