Orasi Performance Test Intelligence Connector (OPTIC)

# Installation of OPTIC

In this step you will download Optic and add the binaries onto the systems which will run the scripts for VuGen development or load generation. The binaries will enable the scripts to communicate with the Windows OS native performance monitoring on the local computer. You will need to gain administrator access to install binaries on each of the LoadRunner machines which will run the scripts. Note that is VuGen for development and testing and a Load Generator for execution during run time.

## Download Options

Optic can either be downloaded as binaries, or downloaded as source and the binaries can be built in Visual Studio. The following describes each method.

1. Download Optic.zip from <https://optic.codeplex.com>

Optic.zip contains the following files

* Optic.dll
* OpticUtil.dll
* OpticVuGenAddin.dll
* OpticVuGenAddin.addin
* DeployOptic.cmd
* Optic Documentation.docx

1. Use Git to clone the project and build it locally

After installing Git execute the following command to clone the repository

* git clone https://git01.codeplex.com/optic Optic

Note – If you are getting timeouts and behind a proxy you may have to run the following commands:

* git config --global http.proxy <http proxy path>
* git config --global https.proxy <https proxy path>

## Load Generator Installation Instructions

There are two DLL’s required as part of the installation process. The both need to be in the %LG\_PATH%\bin directory in these specific locations:

### Optic.dll

Copy Optic.dll to the %LG\_PATH%\bin directory

Example: VuGen or Load Generator

* xcopy Optic.dll "%LG\_PATH%\bin\"

#### Optic.dll Dependencies

Optic.dll requires the Visual C++ Redistributable Packages for Visual Studio 2013

* <http://download.microsoft.com/download/2/E/6/2E61CFA4-993B-4DD4-91DA-3737CD5CD6E3/vcredist_x64.exe>
* <http://download.microsoft.com/download/2/E/6/2E61CFA4-993B-4DD4-91DA-3737CD5CD6E3/vcredist_x32.exe>

### OpticUtil.dll

Copy OpticUtil.dll to the %LG\_PATH%\bin\OpticUtil directory

Example: VuGen or Load Generator

* xcopy OpticUtil.dll "%LG\_PATH%\bin\OpticUtil\"

## VuGen Installation Instructions

There are two files required as part of the installation process. OpticVuGenAddin.dll is currently built for LoadRunner 11.52, to use with LoadRunner 12 it will have to be rebuilt with the correct 12.0 dependency files. The both need to be in the "%VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\" directory in these specific locations:

### OpticVuGenAddin.dll

Copy OpticVuGenAddin.dll to the %VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\ directory

Example: VuGen

* xcopy OpticVuGenAddin.dll %VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\"

### OpticVuGenAddin.addin

Copy OpticVuGenAddin.addin to the %VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\ directory

Example: VuGen

* xcopy OpticVuGenAddin.addin " %VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\"

### OpticUtil.dll

Copy OpticUtil.dll to the %VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\ directory

Example: VuGen or Load Generator

* xcopy OpticUtil.dll %VUGEN\_PATH%\Addins\extra\OpticVuGenAddin\”

# OPTIC Function Descriptions

In these steps we will discuss the Optic functions available to use within VuGen scripts.

## Windows Performance Counters

Windows performance counters paths are referenced in the format of \\Category Name(Instance Name)\Counter Name.

Category names and instance names are determined by the user, there is no specified name that has to be given to either of these. However, the counter names are specific to Optic and cannot be changed. The following information describes this.

### Write Functions

The following are Optic functions that can be used in scripts that write to performance counters:

* IncrementCounter(counter path, value)
* ResetCounter(counter path, value)
* DeleteCounterCategory(counter path)

Only specific counter names are available to write to, so the counter name has to match one of the following or it will throw a runtime error:

* Count
* Rate/Sec

### Read Function

The following Optic function reads performance counters:

* GetCounter(counter path)

# OPTIC Examples

In these steps we will modify the VuGen scripts and validate them locally. Note that updating a Windows performance counter is very fast and will have no measurable impact on script execution.

## Correlating LoadRunner VUsers with AppDynamics

You can increment a counter for each VUser you start in LoadRunner, and read it in AppDynamics. This is done using the Optic IncrementCounter function. The GetCounter function is included in the example but is purely for demonstration of how to read a counter.

Example LoadRunner VuGen code

vuser\_init()

{

int retval = 0;

counterValue = 0;

//Load the Optic dll

retval = lr\_load\_dll("Optic.dll");

//Increment the counter when a VUser begins

IncrementCounter("LoadRunner(VUsers)\\Count", 1);

//Optionally read a counter

counterValue = GetCounter("LoadRunner(VUsers)\\Count");

lr\_log\_message("LoadRunner(VUsers)\\Count: %d", counterValue);

return 0;

}

vuser\_end()

{

//Decrement the counter when a VUser ends

IncrementCounter("LoadRunner(VUsers)\\Count", -1);

    return 0;

}

## Correlating LoadRunner Transactions with AppDynamics Transactions

You can directly correlate an AppDynamics business transaction with a LoadRunner transaction (lr\_start\_transaction) by using the LoadRunner function web\_add\_header and then detecting the header in AppDynamics to discover the transaction. The following are steps to do this:

### In LoadRunner

Add a web\_add\_header function call before the web or http call. The following is example code:

Action()

{

lr\_start\_transaction("MyTransactionName\_01");

web\_add\_header("**AppDHeader**", "MyTransactionName\_01");

web\_service\_call( "StepName=MyTransactionName\_01",

           ...

);

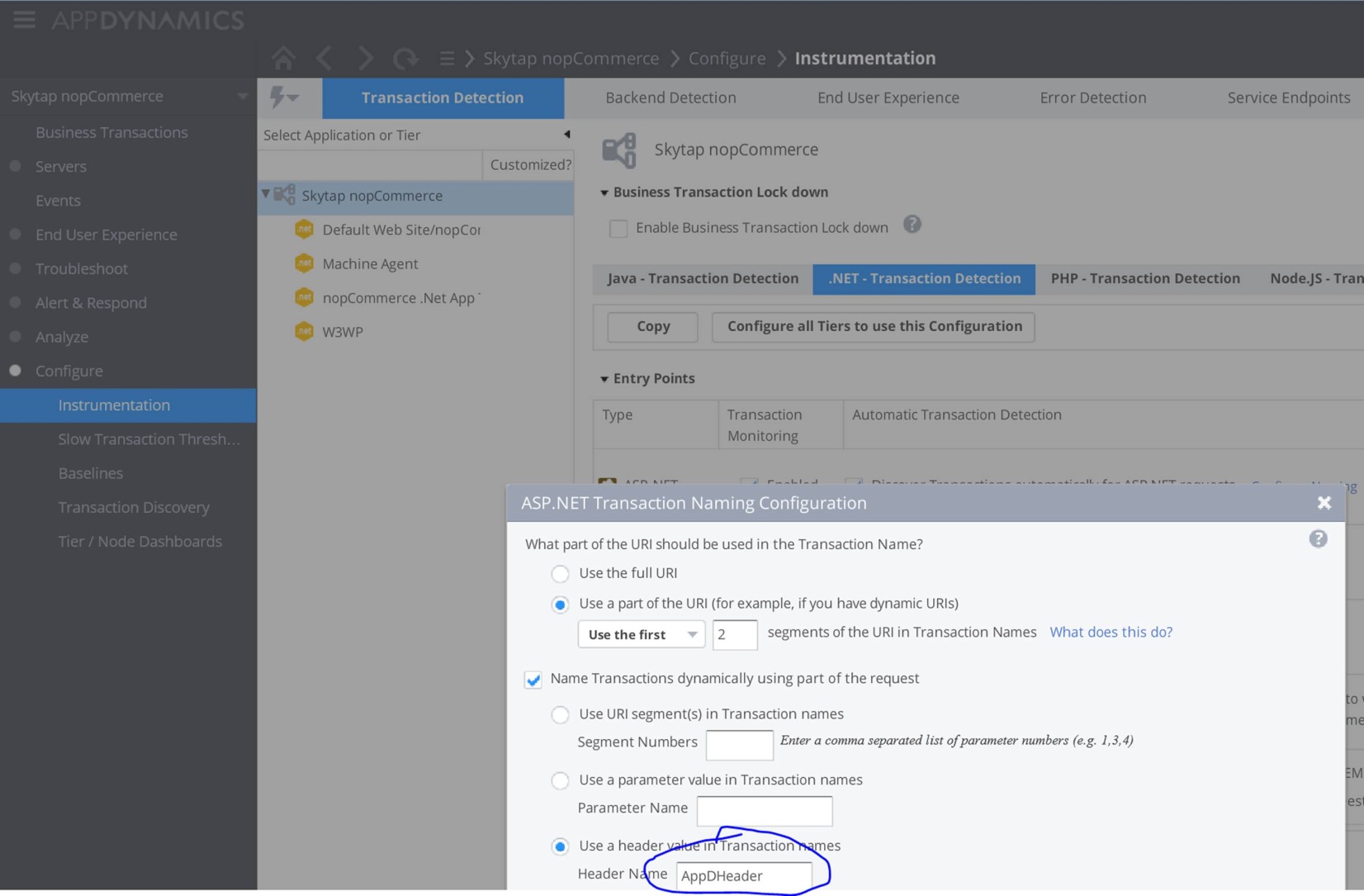
lr\_end\_transaction("MyTransactionName\_01", LR\_AUTO);

    return 0;

}

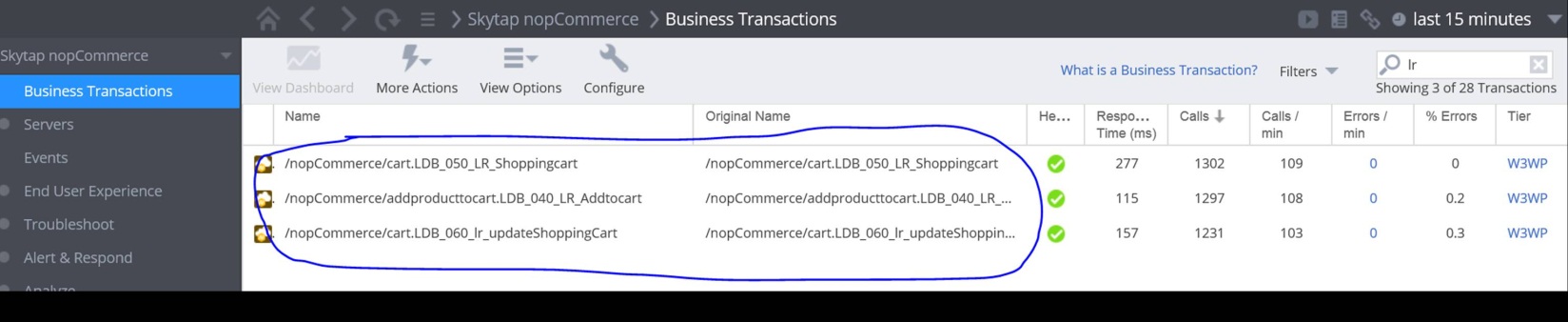
### In AppDynamics

1. Go to Configuration, Transaction Detection, Java (or .Net) Transaction Detection.
2. In Transaction Naming Configuration dialog, check Name transaction dynamically using part of the request
3. Use a header value in transaction names
4. Header Name: AppDHeader



### Validating

1. Run the new code with the web\_addheader in LoadRunner
2. View the new transactions in AppDynamics



### Miscellaneous code

Action()

{

int retval = 0;

lr\_start\_transaction("MyTransactionName");

retval = web\_service\_call( "StepName=MyTransactionName",

           ...

);

lr\_end\_transaction("MyTransactionName", LR\_AUTO);

if(retval == 0)

{

//Pass in the Category, instance, and counter

//Rate/Sec is then recorded in Perfmon

IncrementCounter("LoadRunner(MyTransactionName)\\Rate/Sec", 1);

}

    return 0;

}

## AppDynamics Setup

In this step we will update the AppDynamics server to recognize our new counters and we will also create a basic dashboard to show the LR results within the AppDynamics UI.

### AppDynamics Machine Agent Setup

This step we will install the machine agent with the controller.

1. Install the AppDynamics .Net Machine Agent by using the dotNetAgentSetup64.msi
   1. <https://download.appdynamics.com/onpremise/public/latest/dotNetAgentSetup64.msi>
2. Run the Agent Wizard to set the AppDynamics Controller name, or update the config.xml file directly with the controller information.

### AppDynamics Counter Configuration Update

This step will update the agent configuration file to recognize our new counters. It can be done with the help of a tool, or by manually updating the config.xml file.

1. Use the tool named Windows Performance Counter - Configuration Extension
   1. <http://community.appdynamics.com/t5/eXchange-Community-AppDynamics/Windows-Performance-Counter-Configuration-Extension/idi-p/9713>
2. Modify the Config.xml manually
   1. Open %ProgramData%\AppDynamics\DotNetAgent\Config\config.xml
   2. Edit the file to upload the appropriate counters to the AppDynamics Controller
   3. Notepad %ProgramData%\AppDynamics\DotNetAgent\Config\config.xml

  <machine-agent>

    <perf-counters>

      <perf-counter cat="LoadRunner" name="Rate/Sec" instance="MyTransactionName" />

      <perf-counter cat="LoadRunner" name="Count"    instance="MyTransactionName" />

    </perf-counters>

  </machine-agent>

1. In services.msc, restart the AppDynamics.Agent.Coordinator service

### Validate the Counters in Machine Agent UI

Follow these steps to insure proper correlation of AppDynamics and the data gathered from each of the machines running VuGen.

1. Start the scripts for all of the systems running VuGen that have been updated.
2. View the results in AppDynamics Metric Browser
   * Analyze
   * Metric Browser
   * Select a Metric:
     + Application Infrastructure Performance
     + Machine Agent
     + Individual Nodes
     + <My Node Name>
     + Custom Metrics
     + Performance Monitor
     + LoadRunner
     + Rate/Sec
     + <MyTransactionName>
   * Double-click to add to graph

### Create an AppDynamics Custom Dashboard

* Custom Dashboards
* Create Dashboard
* Add Metric Data Graph
* Add Data Series
* Select Application:
* <Application Name>
* Select a Metric Category:
* Custom (use any metrics)
* Select a Metric:
  + Application Infrastructure Performance
  + Machine Agent
  + Individual Nodes
  + <My Node Name>
  + Custom Metrics
  + Performance Monitor
  + <My Category Name>
  + Rate/Sec
  + <MyTransactionName>

# Appendix 1: Validated Platforms of HP LR

OPTIC has been verified to work on the following versions of HP LoadRunner and ALM/Performance Center

* 11.50
* 11.52
* 12.0

# Troubleshooting Tips

* Make sure you get Optic to work in a single execution of Vugen before deploying to Load Generators
* After creating a controller scenario, run it against the local load generator before running against load generators on other machines as it is easier to debug issues on a single machine.
* Double check the directories in the installation instructions
* Verify everything is in the %LG\_PATH% directory as specified in the installation instructions above.
* You can use the Windows Dependency Walker (<http://www.dependencywalker.com/>) to check if any dependent DLL’s are not available.
  + It is common to see GPSVC.DLL, DBGHELP.DLL, and SHDOCVW.DLL listed as missing dependencies in Windows Dependency Walker
* If you are getting an error from the Controller only, make sure that you copy the DLL to the same directory of all the Load Generator machines.

Error

Cannot find class: OpticVuGenAddin.IsValidCodeSelectedCondition

Future missing objects will not cause an error message.

The addin 'OpticVuGenAddin.dll' could not be loaded:

System.IO.FileLoadException: Could not load file or assembly 'file:///C:\Program Files (x86)\HP\LoadRunner\AddIns\extra\OpticVuGenAddin\OpticVuGenAddin.dll' or one of its dependencies. Operation is not supported. (Exception from HRESULT: 0x80131515)

File name: 'file:///C:\Program Files (x86)\HP\LoadRunner\AddIns\extra\OpticVuGenAddin\OpticVuGenAddin.dll' ---> System.NotSupportedException: An attempt was made to load an assembly from a network location which would have caused the assembly to be sandboxed in previous versions of the .NET Framework. This release of the .NET Framework does not enable CAS policy by default, so this load may be dangerous. If this load is not intended to sandbox the assembly, please enable the loadFromRemoteSources switch. See http://go.microsoft.com/fwlink/?LinkId=155569 for more information.

Must use Windows Explorer to change file properties to safe by clicking on “Unblock”. It was downloaded from web so is considered unsafe by default.

Custom Time Events Programmatically

You can't do this right now, but in the future you will be able to create custom time line events programmatically. You should tag the start and end time of the load test from LR in AppDynamics so this shows up clearly and can be excluded from the baseline

<http://community.appdynamics.com/t5/ideas/v2/ideaexchangepage/blog-id/extensions>