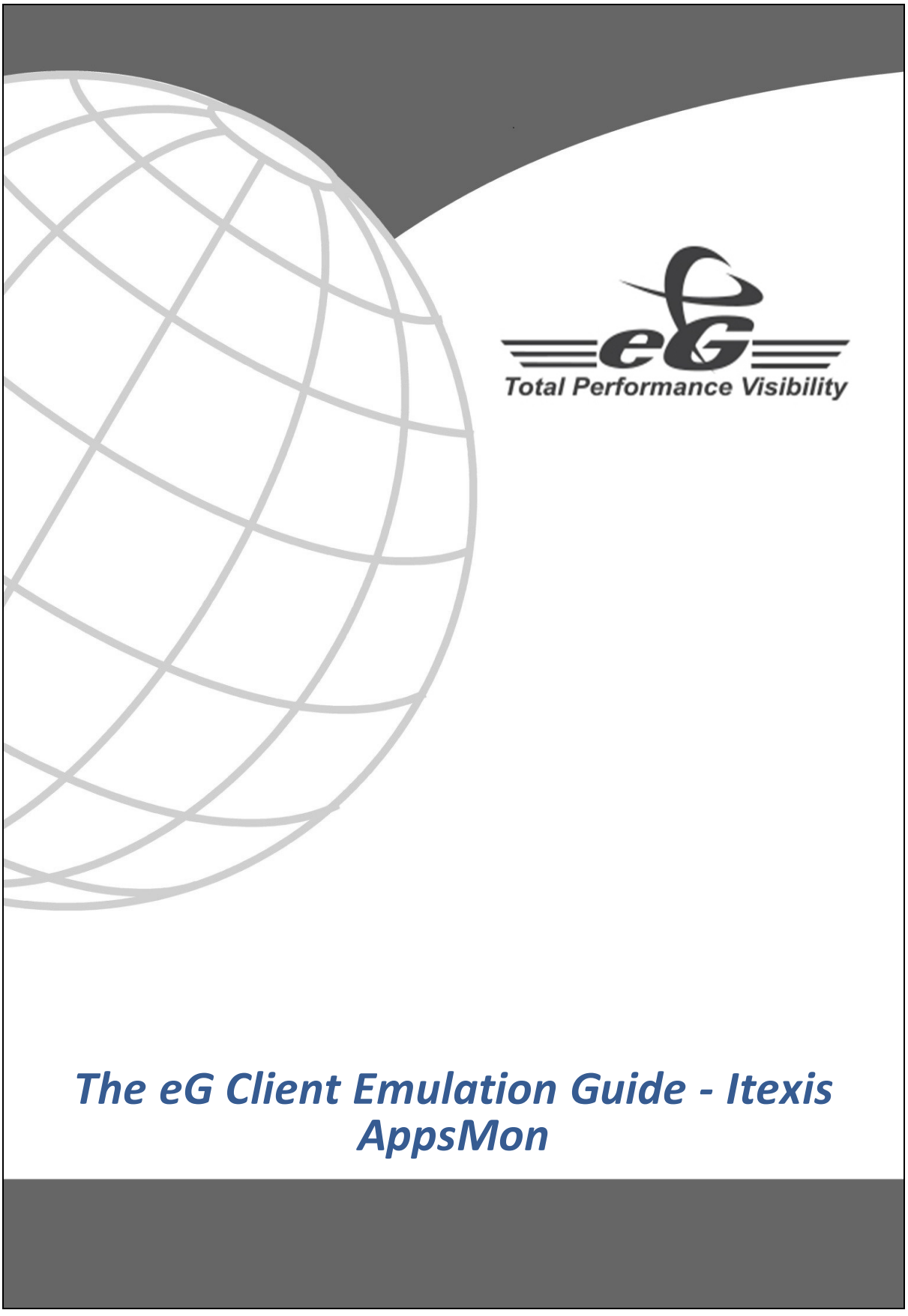

The eG Client Emulation Guide - Itexis AppsMon

Table of contents

Table of Figures

1

Introduction

As IT infrastructures evolve into being business-critical, high availability and peak performance of the IT infrastructure is becoming as critical as reliability is to a telephone network. In such infrastructures, it is imperative to determine in real-time when service failures or slowdown problems occur, and to accurately pin-point which step(s) in the service access are impacting the end-user experience. Speedy problem detection and accurate diagnosis can reduce service downtime and the consequent business impact of critical IT services.

eG's client emulation and monitoring capability goes well beyond basic protocol-level up/down testing. IT administrators can record typical user accesses to mission-critical infrastructure services, and later have eG agents periodically playback the recorded accesses using the exact same client applications that users employ to access a service. The eG client emulator effectively simulates multi-step user interactions with a service (e.g., login, browse, submit data, fill-in forms, etc.).

By collecting availability and response time statistics for the complete service interaction and comparing the metrics with time-of-day, auto-generated or administrator-defined fixed thresholds, the eG client emulator immediately alerts administrators during potential service outages or slowdowns.

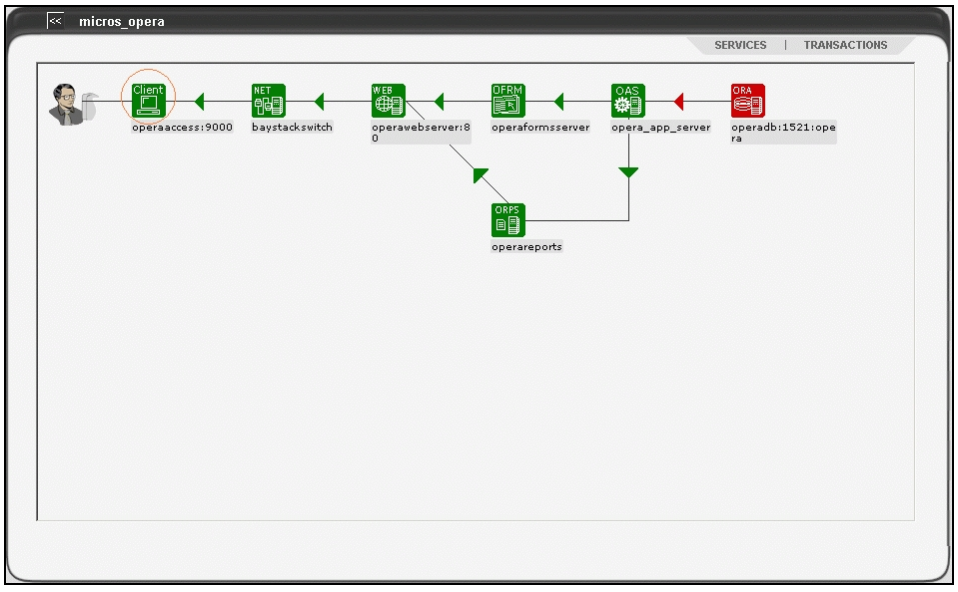

Figure 1.1: A service topology representation comprising of an emulated client component

By comparing the response times across each of the steps of a multi-step service access, administrators can detect the exact step(s) that could be causing user-visible slowdowns. Furthermore, using its patented auto-triage capability, the eG Enterprise suite is able to correlate the end-to-end service performance indicators provided by the eG client emulator with critical indicators of network, system, and application availability, performance, and usage to pin-point where the root-cause of a slow-down lies. Customers can use the

integrated solution to improve the quality of their service offerings, thereby enhancing their competitive positioning, lowering their operations costs, and optimizing the usage of their infrastructure.

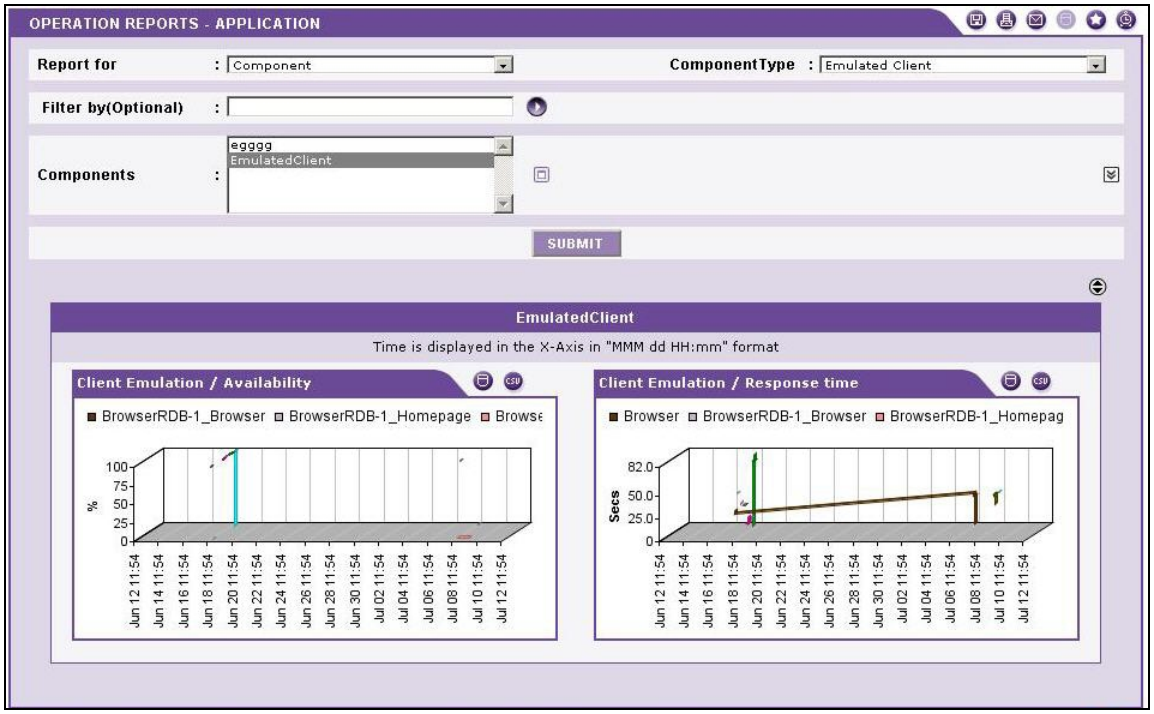

Figure 1.2: Comparing the availability and response times for each step of a service interaction

Since it is capable of supporting web-based or native client applications, the eG Enterprise suite can handle web-based as well as proprietary/native client applications (and even thin-client/Citrix applications) equally well.

1.1 Benefits of the eG Client Emulator

- Emulate real user interactions to Web/non-Web services and record service availability and end-to-end response times;
- Identify which step(s) of a service interaction are causing slowdowns;
- Provide instantaneous alerts on service outages and slowdowns;
- Facilitate root-cause diagnosis and infrastructure optimization through correlation with other network, server, application performance indicators;
- Offers a reliable, cost-effective way of automating routine service health checks;

1.2 Architecture

Typically, the first step in using the eG client emulator is to record a sequence of user activities when accessing the service(s) to be monitored. The end result of the recording process is a script that can be later played back to emulate user activity. The recorded script not only includes the sequence of transactions

necessary to emulate the user activity, but also has information on those transactions - each transaction corresponds to a single user interaction (e.g., logging in, adding to shopping cart, etc).

To record the script, eG integrates with one of the three client emulation tools: Tevron's CitraTest, Seapine Software's QA Wizard and Itexis' AppsMon for Windows. A development environment for the client emulation tool is necessary to perform the recording.

The recorded script can be played back on a dedicated system, and the results are reported as availability and response time measures for the overall playback activity. In addition, response times of each of the transactions involved can be obtained so as to assist the IT service manager in pin-pointing which of the transactions of the multi-step user interaction with the service could be causing a slow-down of the overall service.

The integration of the eG Enterprise suite with the client emulation tools is performed at the eG agent side. An eG agent is provided with the location of the recorded script that it has to execute to emulate a user activity. Based on the pre-specified frequency of the test, the agent executes the script, analyzes the script results, and reports availability and response time information to the eG manager. The eG manager receives these client emulation reports and correlates them in real-time with critical in-depth resource usage and server-side processing metrics that it receives from the other agents, to report on potential bottlenecks in the target IT infrastructure.

The eG external agent implements the integration with the client emulation tools. Since the client emulation tools require a dedicated system to operate on, an external agent can either perform client emulation tests or the other eG protocol emulation tests, but not both. When adding a new external agent, depending on the eG license available, the administrator can specify whether the agent must be allowed to perform client emulation activity or not.

Figure 1.3 depicts how the integration of the eG external agent with the client emulation tools works. The recorded script file must be made available (this is a manual process) on the system that the eG external agent is running on. Also, for the agent to execute the recorded script, the runtime environment of the client emulation tool must be installed.

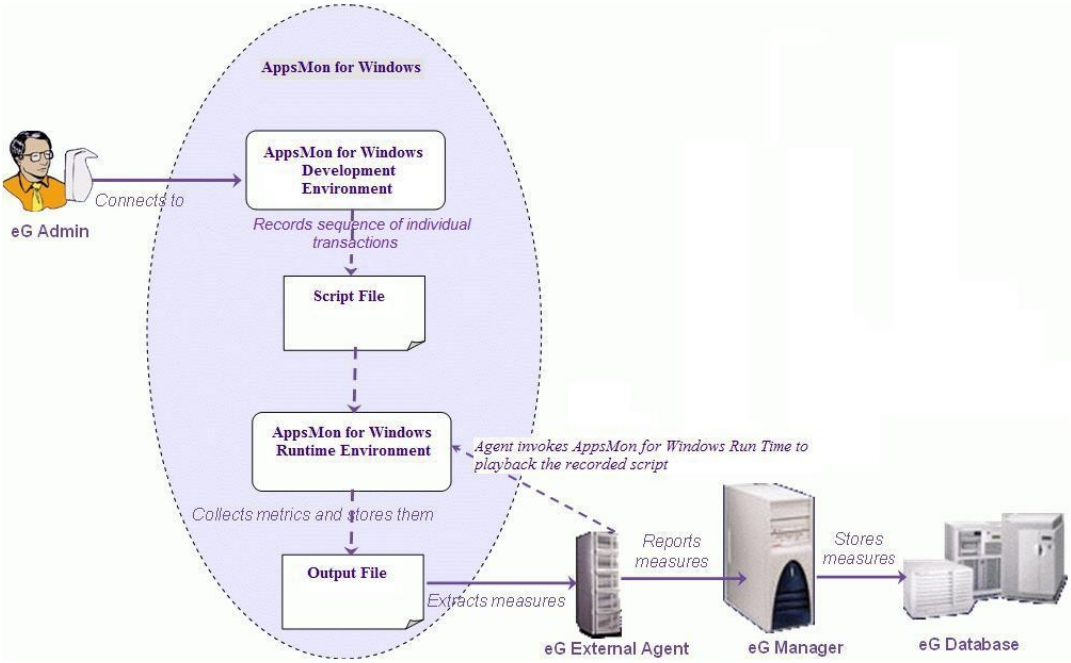

Figure 1.3: A typical eG-Client emulation tool Integration

1.3 Licensing

- An eG external agent is required for monitoring emulated clients
- eG external agents assigned to handle emulation tests cannot execute other eG tests
- Client emulation is supported on Windows platforms only
- For the integration to work, the eG license should have the **Client Emulation** capability enabled
- The Client emulation tool must be installed when an eG external agent with client emulation is started
- The Client emulation tool's development environment is mandatory for building a script

The chapters that follow will discuss how eG is integrated with **Itexis – AppsMon for Windows** tool.

2

Integrating eG Enterprise with Itexis-AppsMon for windows

Itexis - AppsMon for windows is an automated performance testing tool for Web and non-web-based windows applications. This tool emulates a user request to an application by recording the sequence of transactions that constitute a typical user access, in a script file. When this script is executed, the recorded sequence of transactions is replayed, and the availability and response time of the application are measured.

2.1 Steps for Integrating eG with Itexis- AppsMon for Windows

The primary steps involved in the eG - Itexis-AppsMon for Windows integration are installing the AppsMon for Windows development environment and building a script file using it. To ensure a smooth and effective integration, **AppsMon for Windows v4.1.6** needs to be installed. For an elaborate procedure on the installation of Itexis - AppsMon for Windows, refer to the AppsMon for Windows manuals.

Note:

Since the AppsMon for Windows emulates real user transactions by opening client applications and emulating user clicks on the client application, AppsMon for Windows requires complete control of the system on which it executes. Therefore, the AppsMon for Windows should be installed on a dedicated system.

System requirement for installing the AppsMon for Windows:

- Any one of the Windows 2008 and above or Windows 7 and above server/workstation with the following software installed:
- A web browser for e.g., Google Chrome, Microsoft Internet Explorer (IE) version 8.0 or higher, etc.
- Microsoft .Net Framework 4.5 or above.
- Visual Basics Express 2010, 2012 and 2013
- Microsoft SharePoint Designer 2007. Ensure that you enable only the following features while installing the SharePoint Designer 2007:
  - Microsoft Office Document Imaging
  - Microsoft Office Document Scanning

Once the development environment is installed, proceed to record a test script. A test script is a sequence of actions that are recorded as a user accesses one of the services in the target infrastructure. The recorded script can then be played back to emulate user accesses to the service. Image and text recognition techniques are used during playback to determine whether playback of a script succeeded or not.

After building the script file, use the eG administrative interface to configure the script playback. Next, start the external agent which will playback the script, and finally, view the measures returned by **AppsMon for Windows** in the eG monitor interface.

The sections to come will take the help of two illustrated examples to explain how eG client emulation works. While the first example deals with a web application, the second example targets a Citrix application.

2.2 Client Emulation for a Web Application

The script file that is built in the first example will emulate a user request to http:// www.eurail.com website hosted by the server 192.168.9.194 using which the user checks the availability of high-speed trains, pass details and the deals and discounts. The example will be used to monitor critical transactions along the way.

2.2.1 Building a Script File

To build the script file, do the following:

1. The first step towards building a script is to open a session using which the transactions can be recorded. To open a session, follow the menu sequence: Programs - >Itexis -> Session

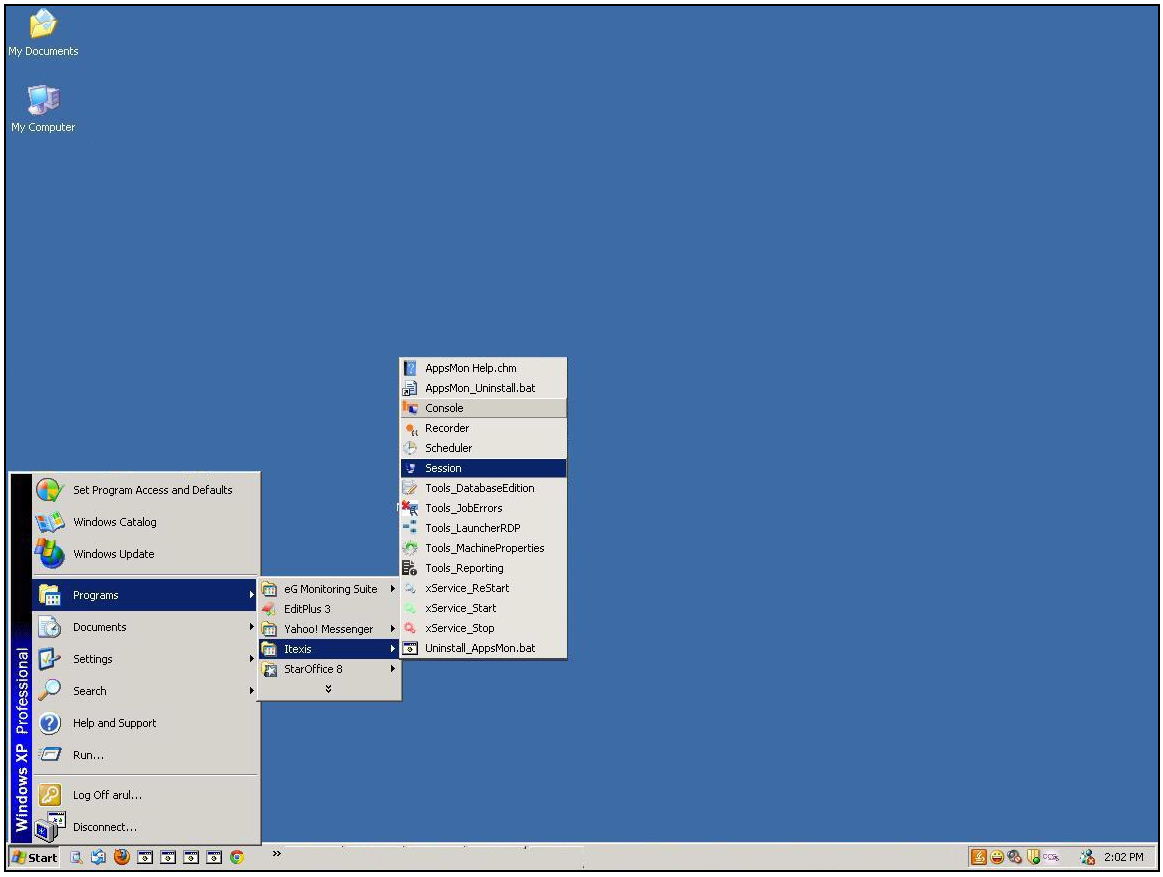

Figure 2.1: Opening the Session

2. Once the **Session** window is opened, an icon appears on the system tray with a callout message

stating **Session Locked**. In order to unlock this session, provide the credentials of a valid **User** and **Password** in the **UnLock Session** section (see Figure 2.2).

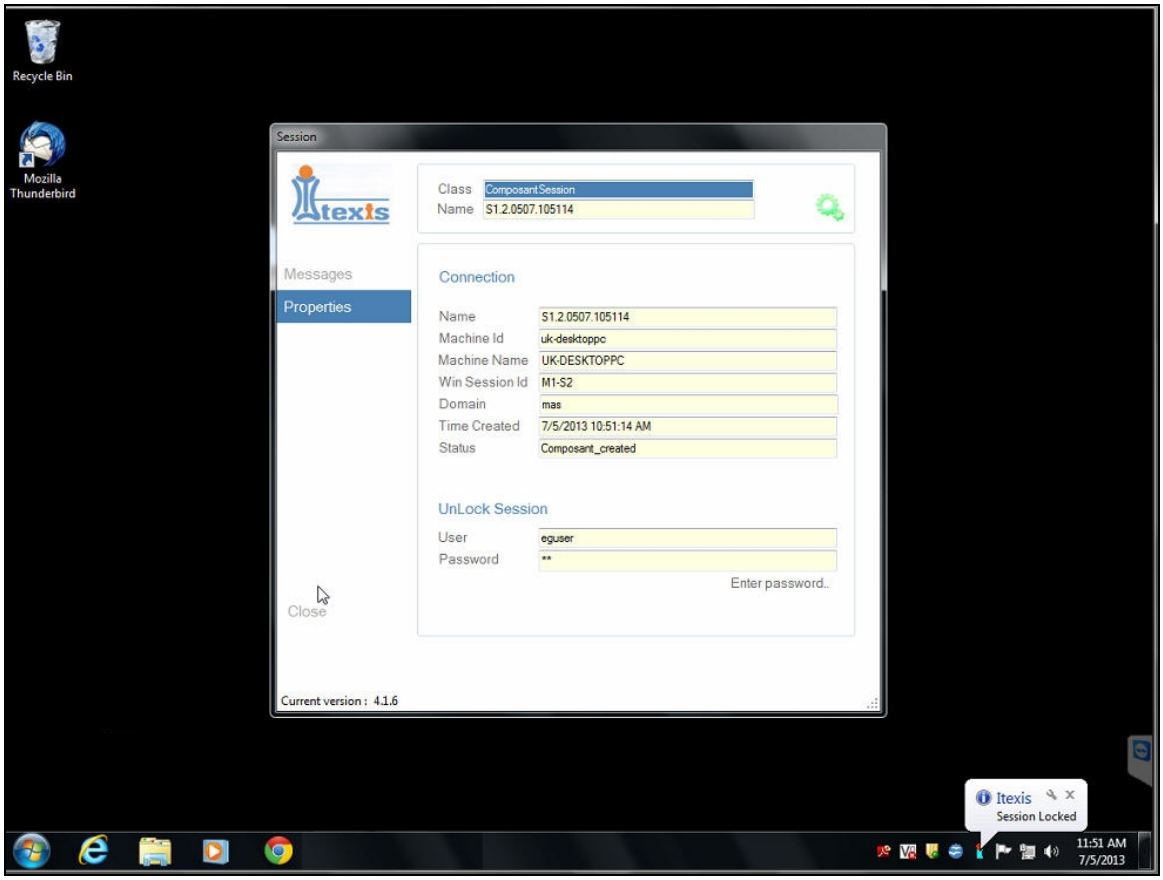

Figure 2.2: The Session window

3. The **Session** window will then disappear and the callout message stating **Session Unlocked** will appear above the icon.

   Note:

   Initially, when the **Session** window is opened, you will be allowed to key in the user credentials alone. All other activities such as moving your mouse, clicking on the Start button, opening a browser etc, will not be allowed.

4. The second step towards building a script is to start the recording. To do so, right click on the icon and select the **Show Recorder** option as shown in Figure 2.3.

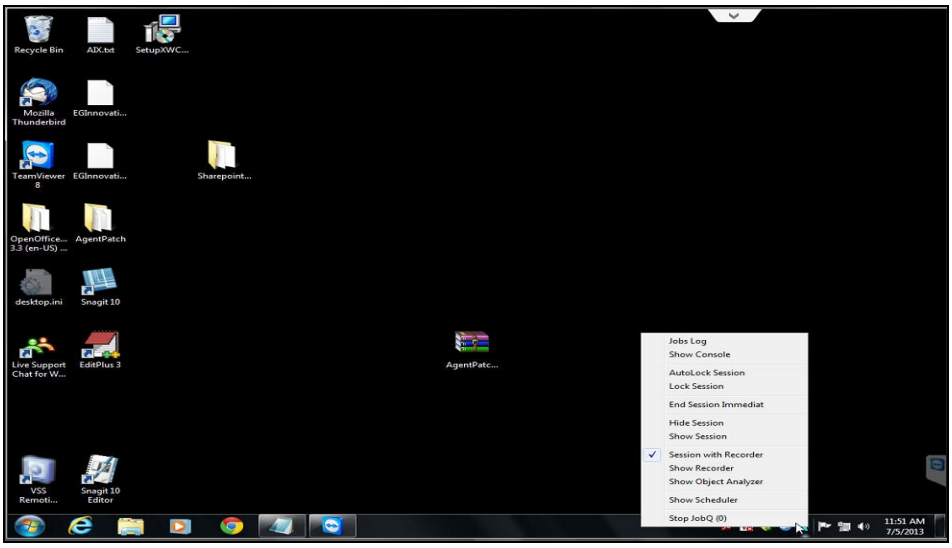

Figure 2.3: Selecting the Show Recorder option

5. The **Recorder** window will then appear as shown in Figure 2.4. By default, the **Recorder** window consists of two sections - a **Components** section at the left and a context sensitive section at the right which by default, displays the **Default Properties** of the **Scripts**.

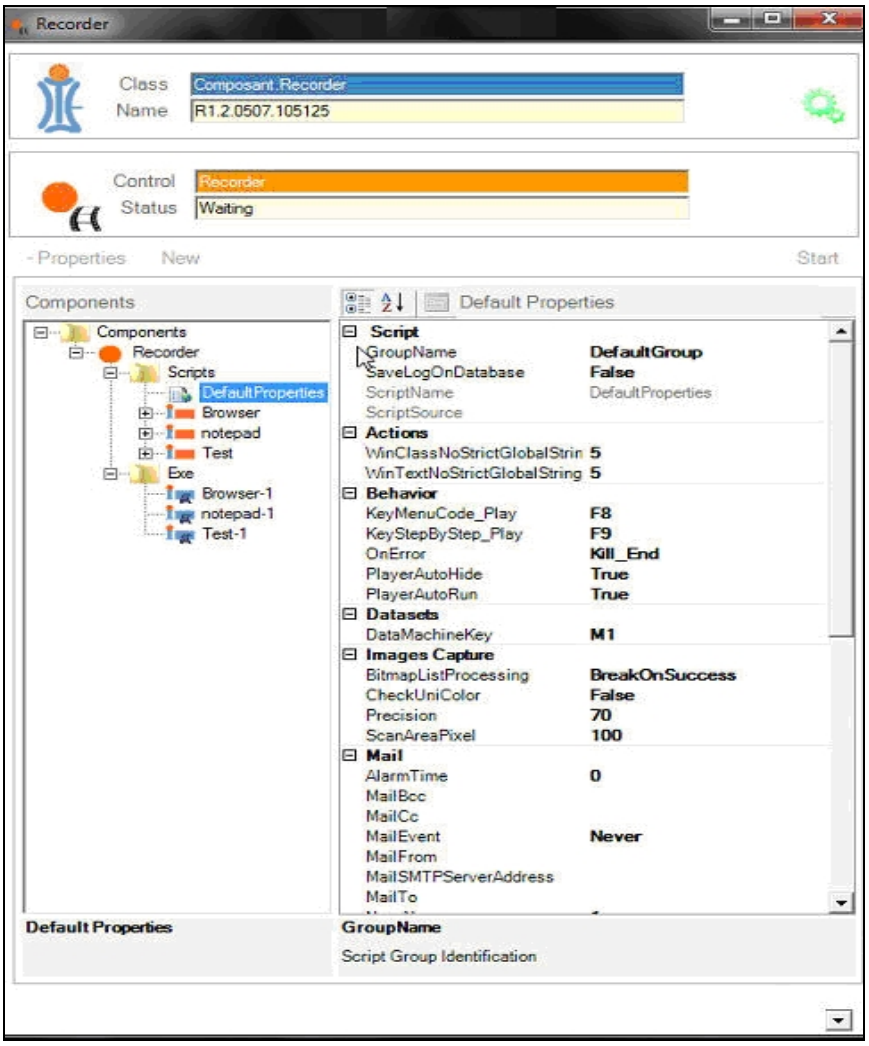

Figure 2.4: The default Recorder window

6. To create a new script, either click the **New** option by right clicking the **Scripts** tree in the **Components** section or click **New** in the **Recorder** window as shown in Figure 2.4.
7. A **New Script** pop up window will then appear as shown in Figure 2.5. Here, enter the name of the **Script** that needs to be built and click **OK**.

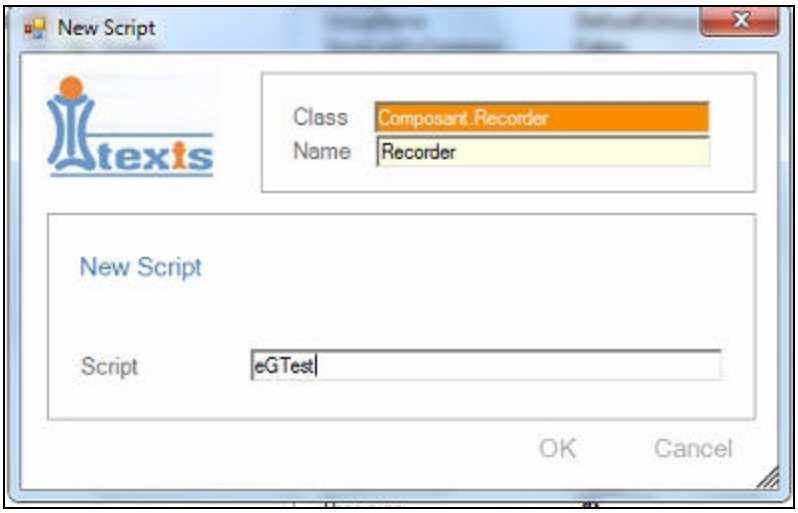

Figure 2.5: Creating a new script

8. The script thus created will be listed automatically under the **Scripts** folder of the **Components** section as shown in Figure 2.6.

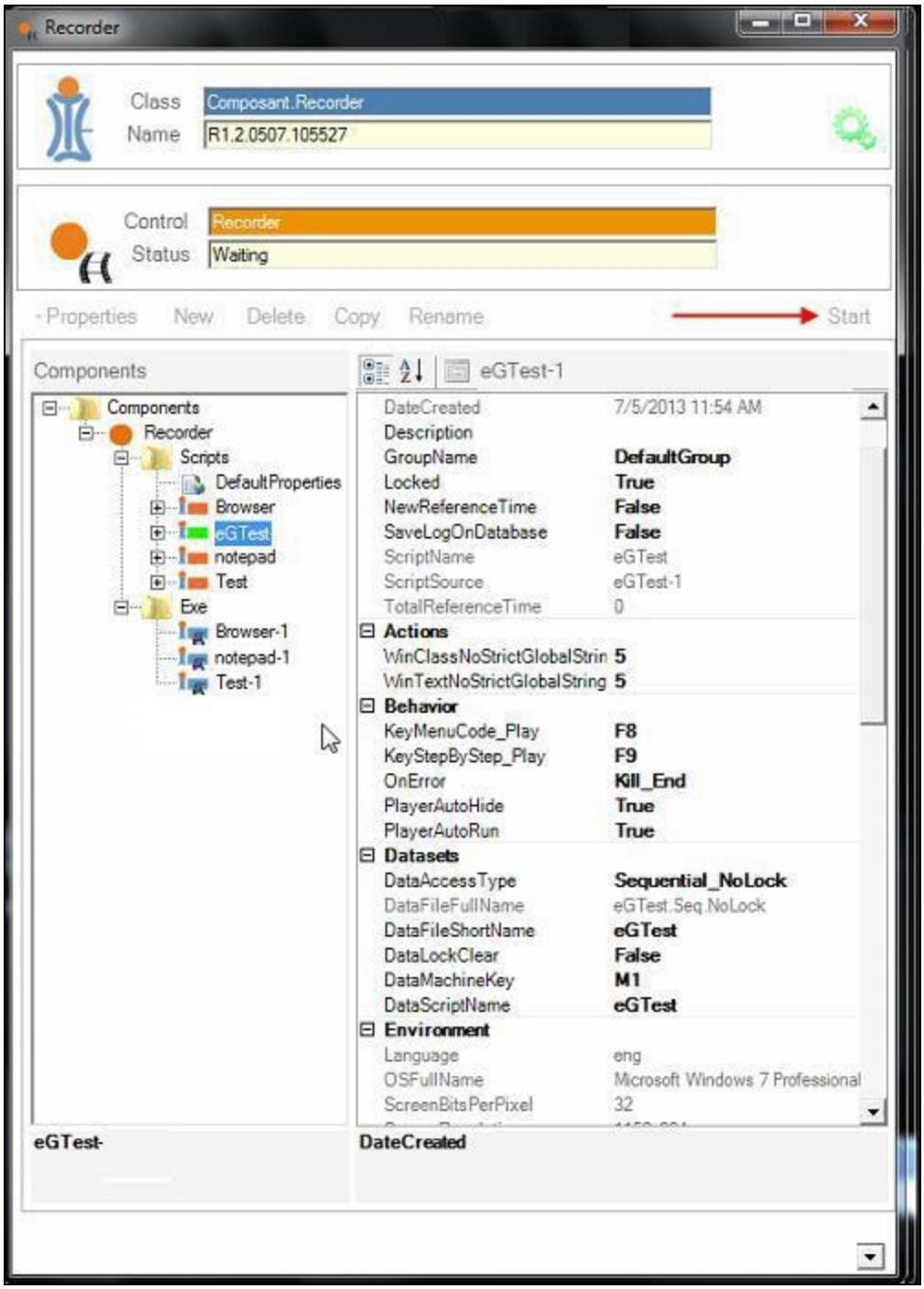

Figure 2.6: Viewing the created Script

The next step towards building the script is to start the recording. The primary purpose of the recording is to start capturing the user transactions (as images or as text) that are to be emulated. Before commencing image capturing, it would help if you prepare a list of broad steps that need to be followed to access the application to be monitored. Before proceeding to the recording, let us be familiar with the keys that are commonly used during the recording process.

Key	Control
A valid key for e.g., F8	To access the **Menu** pop up window while recording (see Figure 2.9)
Tab	To navigate through the **Bitmaps, OCR, Anchors..**page
Enter	To validate the chosen option in the **Menu** pop up window
Escape	To unlock the current session
Arrow keys	To navigate across the options in the **Menu** pop up window

Note:

By default, the key used to access the **Menu** pop up window is configurable. To change the key as per your wish, do the following:

- Click on the **Recorder** option in the **Components** section of Figure 2.7.
- Edit the **KeyMenuCode_Rec** field with the key of your choice, say for example F12 in the context sensitive right section as shown in Figure 2.7.
- If the Microsoft SharePoint designer 2007 is successfully installed, then the **OCR_Is Active** field is by default, set to **True** as shown in Figure 2.7. This ensures you that the AppsMon for Windows is ready for capturing the transactions.

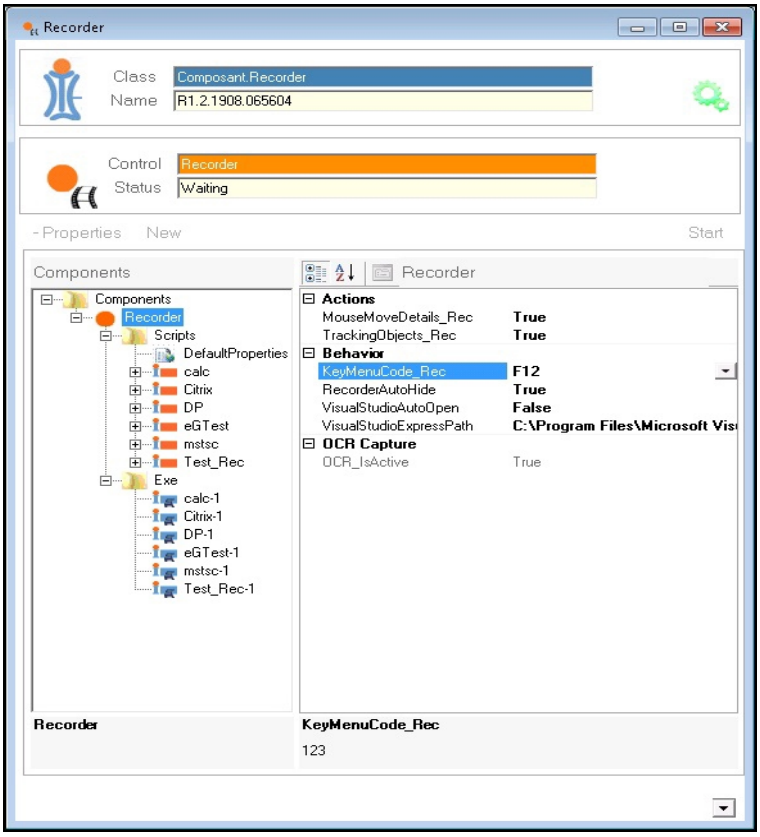

Figure 2.7: Editing the KeyMenuCode_Rec field

9. Now, let us begin the recording by clicking the **Start** button in Figure 2.6. Since our example involves accessing the website: http://www.eurail.com/ (Europe Railways), the first step would be to open a browser through which the homepage of this website can be accessed.
10. The **Eurail** home page will now be loaded successfully as shown in Figure 2.8.
11. Merely placing the mouse cursor on the **EuRail** logo will help you to capture the first transaction (see Figure 2.8).

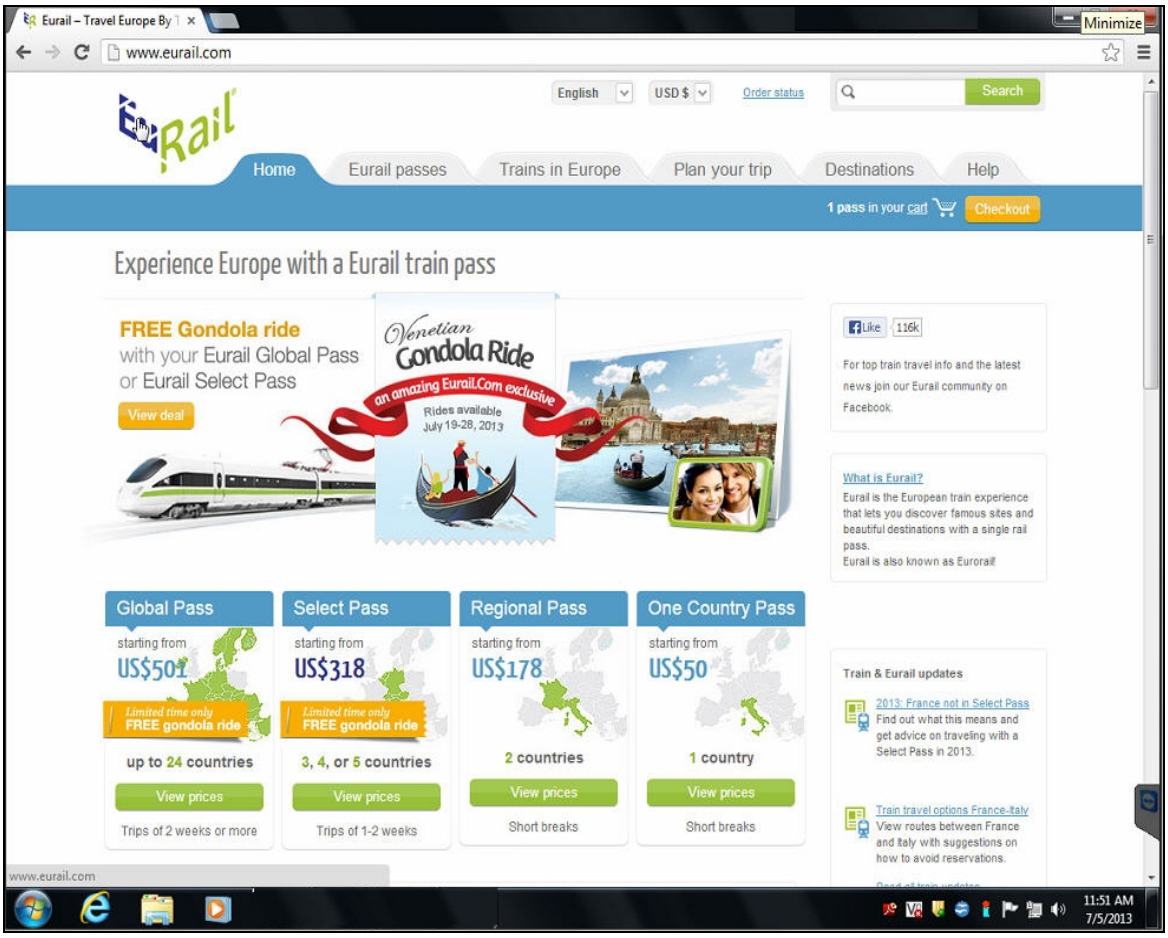

Figure 2.8: The EuRail home page

12. To save the transaction, access the **Menu** pop up window by pressing the F8 key. Figure 2.9 will appear.

Figure 2.9: The Menu pop up window

13. By default, the **Bitmaps, OCR, Anchors..** option will be selected in the **Menu** pop up window (see Figure 2.9). Pressing the **Enter** key will lead you to the **Capture Bitmap/OCR/Transactions** page as shown in Figure 2.10.

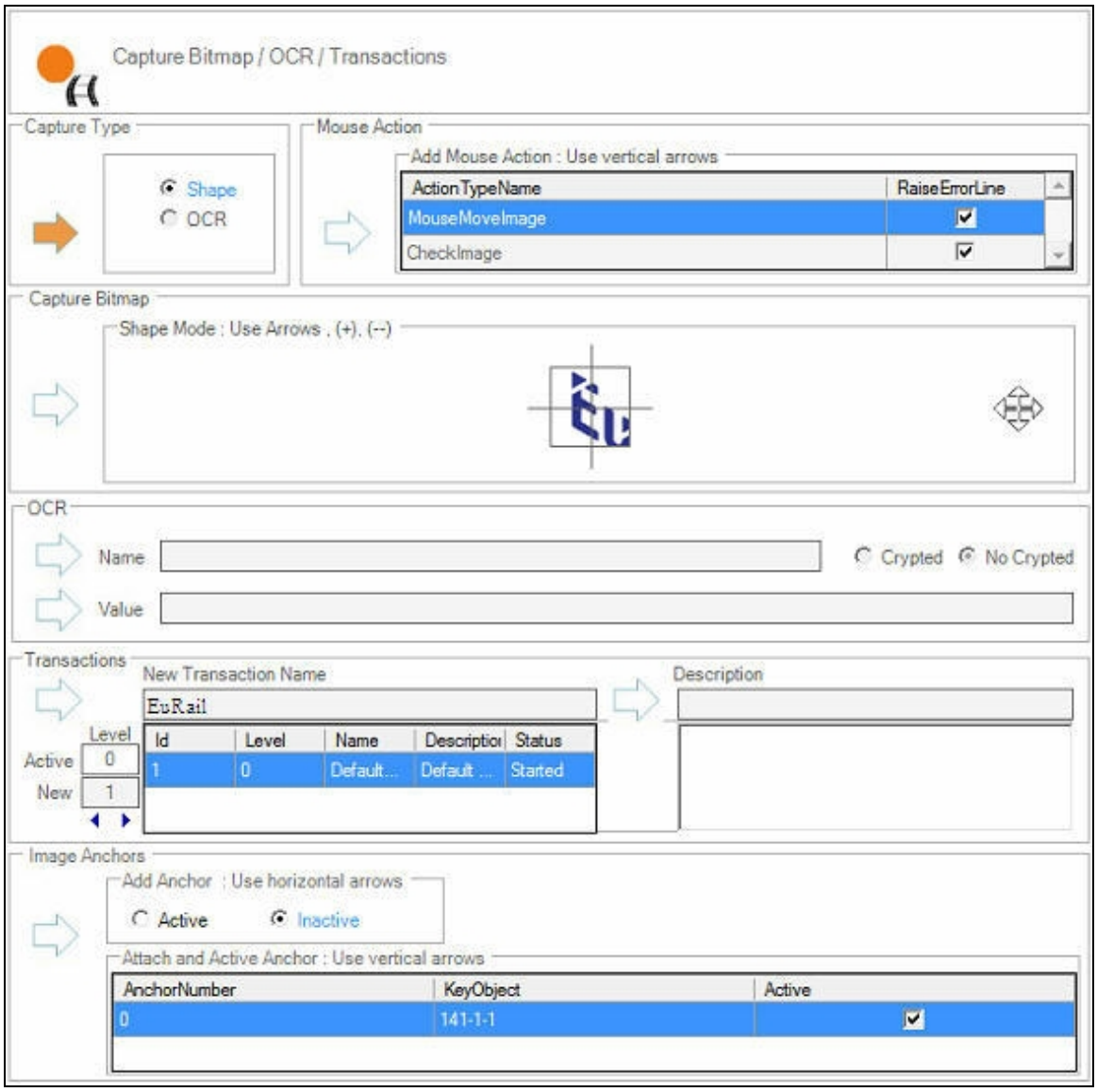

Figure 2.10: Capturing the EuRail logo

14. Figure 2.10 allows you to save the captured **EuRail** logo as a transaction. By default, the **EuRail** logo that is captured will appear in the **Capture Bitmap** section. This capture will be associated with the **Shape** option of the **Capture Type** section and the **MouseMoveImage** option of the **Mouse Action** section, by default. Typically, when you select the **Shape** option, the AppsMon for Windows recognizes the capture as an image and saves the capture in BMP format. Likewise, you can choose any of the option from **Mouse Action** section depending on your requirements to record a transaction.

15. Alternately, if you select the **OCR** option from the **Capture Type** section, the AppsMon for Windows recognizes the capture as a text and automatically displays the captured text in the **Value** text box of the **OCR** section. Suppose if the capture is not successful, then instead of the captured text, **Empty** will be displayed in the **Value** text box. This way, you can identify that your capture was not successful and you can either capture the text once again or adjust the coordinates of the capture – i.e., in our case the **EuRail** logo by moving the '+' or '-' keys in the **Capture Bitmap** section (see Figure 2.10).

16. To save the captured **EuRail** logo as a transaction, enter the name of the transaction in the **New Transaction Name** text box of the **Transactions** section and press the **Enter key**.

17. Next, capture the **Trains in Europe** link (see Figure 2.8) as your next transaction. Follow the steps 13 to 16 to save the captured transaction.

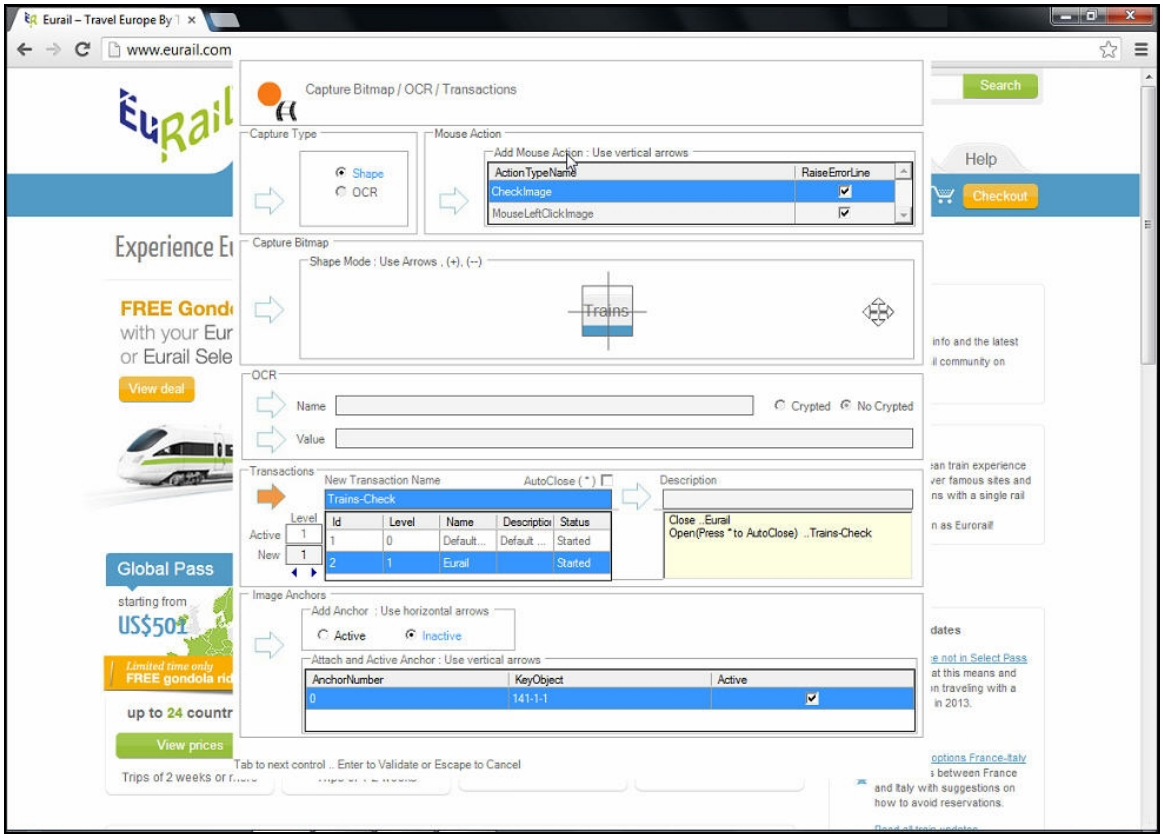

Figure 2.11: Capturing the Trains in Europe link

18. Next, try capturing the **High-speed trains** link as a transaction. To do so, follow the steps 13-16 described above (see Figure 2.12).

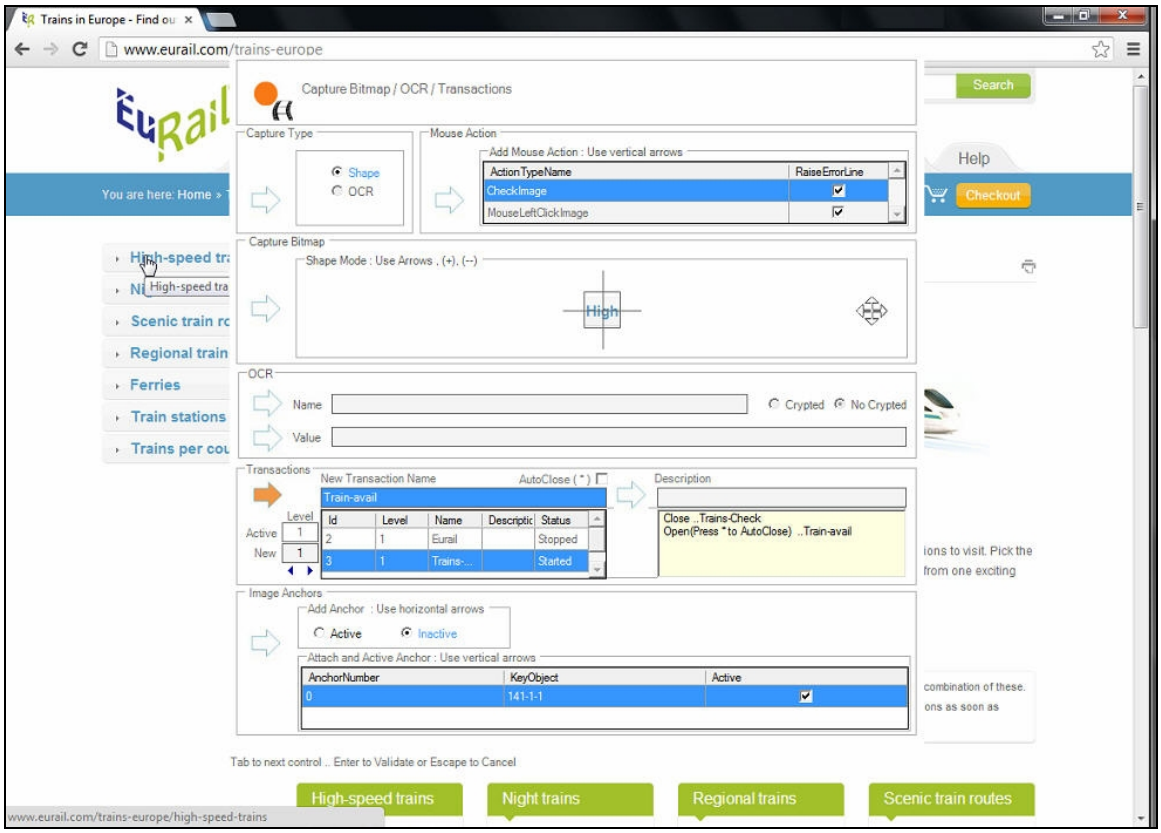

Figure 2.12: Capturing the High-speed trains link

19. To find the high speed trains that operate in Italy, click on the **Le Freece (Eurostar Italia)** link. To capture this as your next transaction, follow the steps 13-16 above (see Figure 2.13).

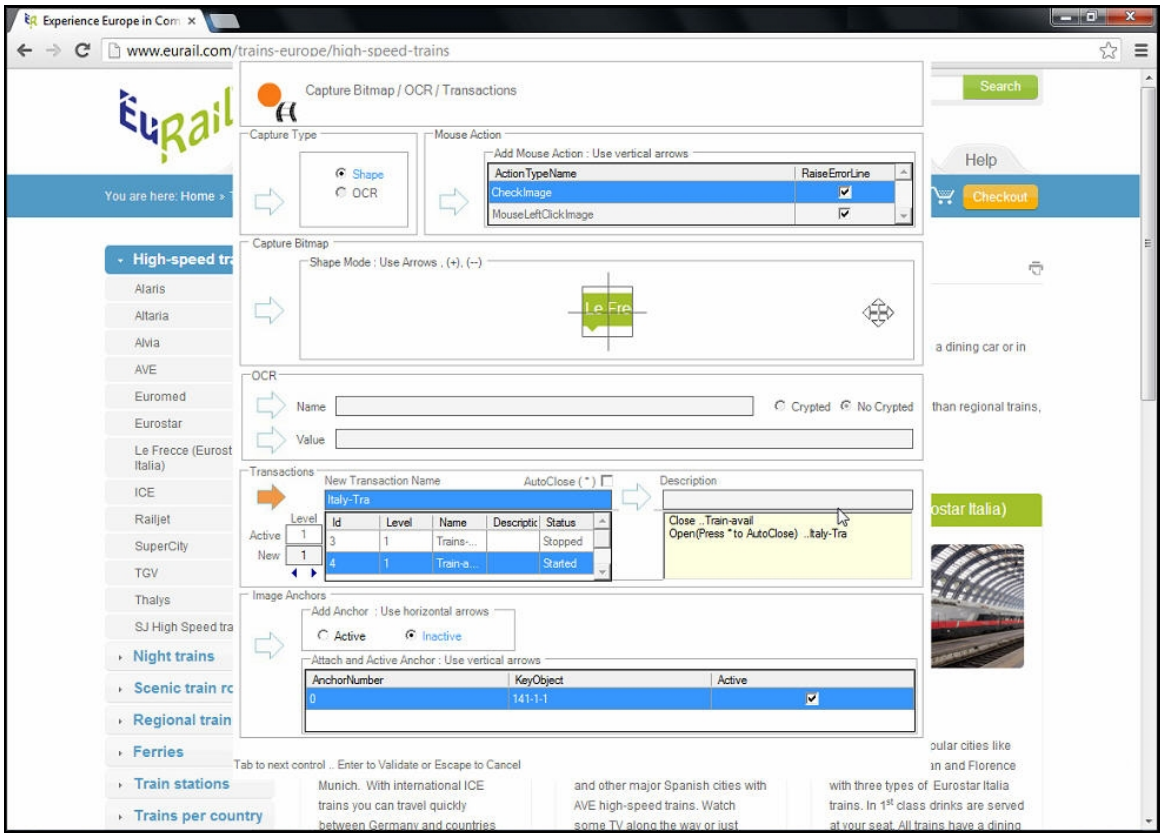

Figure 2.13: Capturing the Le Freece (Eurostar Italia) link

20. Now to view the pass price for travel, click on the **Select Pass**. To know the pass price for three different countries, click on the **3 countries** link. To capture this as your next transaction, follow the steps 13 - 16 described above (see Figure 2.14).

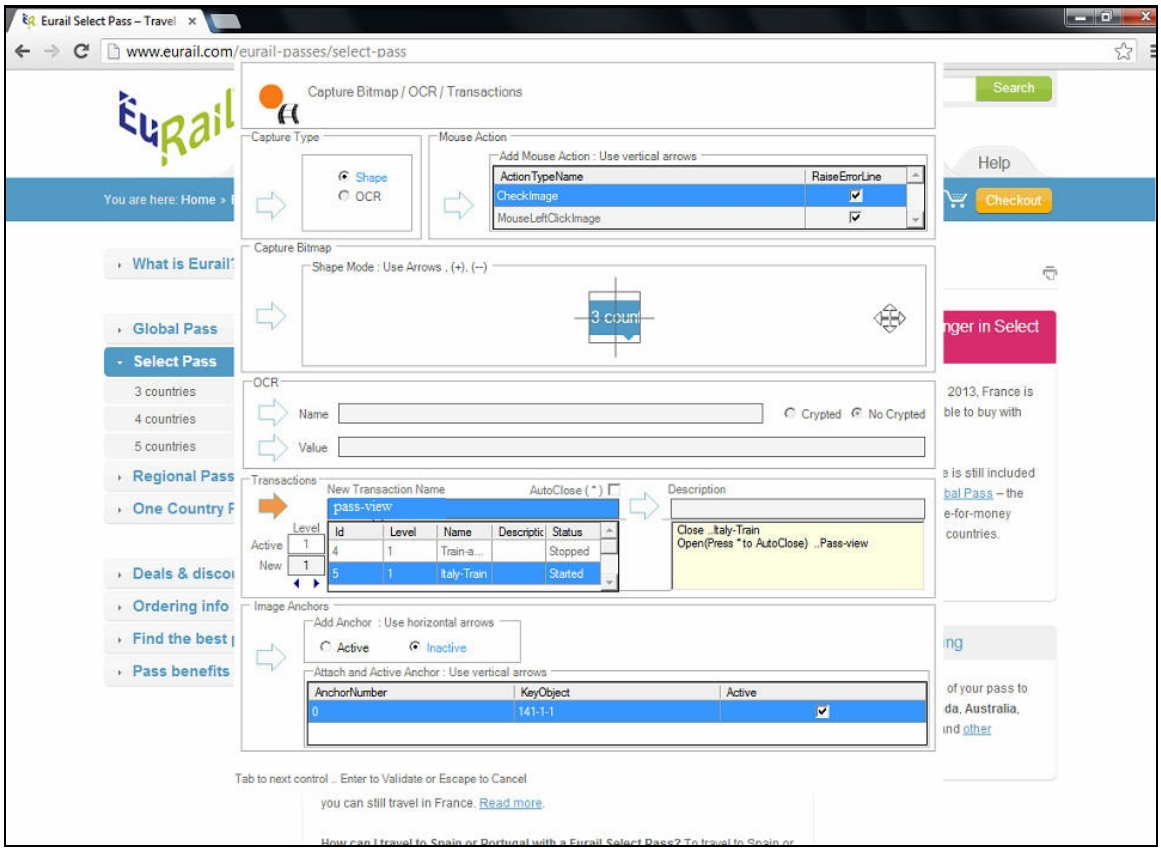

Figure 2.14: Capturing 3 Countries link

21. Finally, to know the deals and discounts available for travel to **3 countries**, click on the **Deals and discounts** link. To capture this as your final transaction, follow the steps 13-16 described above and then close the browser (see Figure 2.15).

22. Suppose if you wish to stop tracking certain delicate **Windows objects** like login credentials, passwords etc, during recording a transaction, you can do so by selecting the **Windows Objects**.. **Stop Tracking** option from the **Menu** pop up window (see Figure 2.9). Say for example, you are trying to log into a remote system whose credentials are strictly confidential. In such a case, provide the credentials and then invoke the **Menu** pop up window, select the **Windows Objects**.. **Stop Tracking** option and then try validating the credentials. Once the login is successfully validated, you can select the **Windows Objects**.. **Start Tracking** option from the Menu pop up window.

23. Once you have saved all the required transactions, you can stop the recording by selecting the Stop Recording option from the Menu pop up window that appears when you press the F8 key (see Figure 2.16).

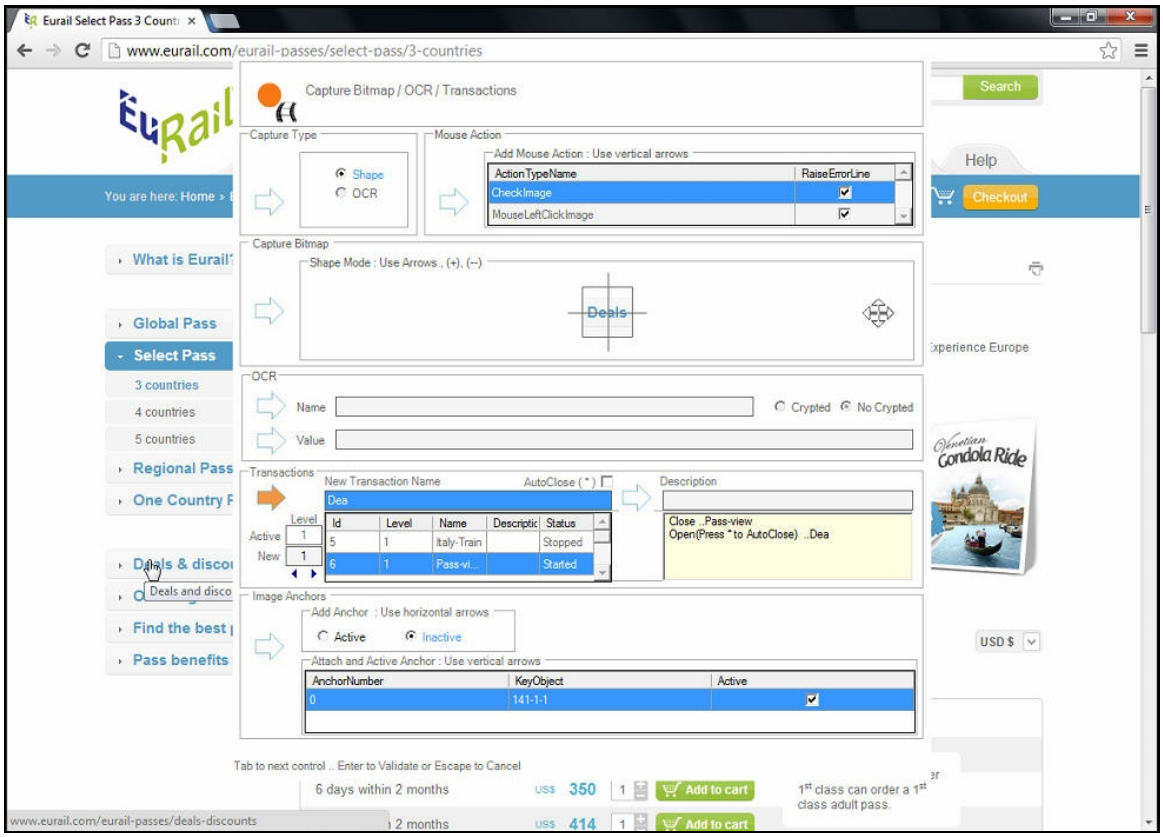

Figure 2.15: Capturing the Deals and Discounts link

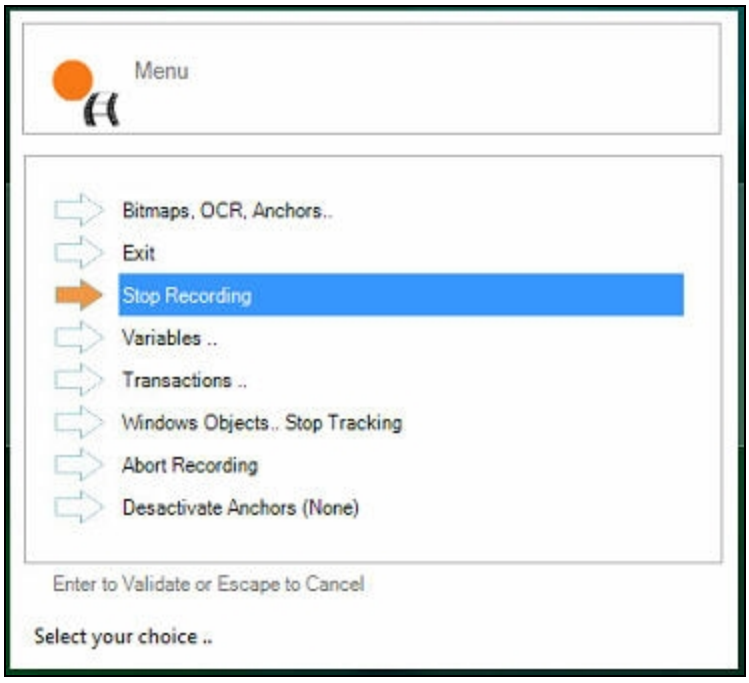

Figure 2.16: Stopping the Recording

Once the recording is stopped, a callout message stating Session Hooked will appear above the icon. A Create Script window will now appear which will start compiling the script automatically. Once the script has been built successfully, a message stating Build is Success will appear (see Figure 2.17) and an executable of the script will be created - in our example this will be **eGTest-1.exe** (see Figure 2.17).The script that is recorded will be stored in the following location:

C:\Documents and Settings\All Users\Application Data\Itexis\Components\Scripts

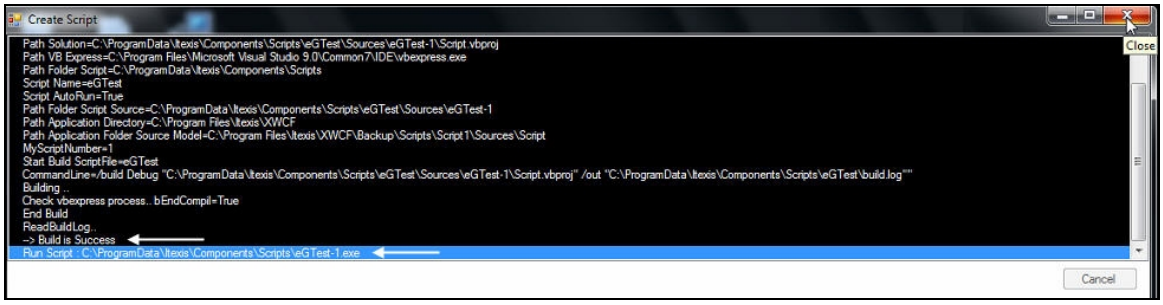

Figure 2.17: The Create Script window

24. Now, the script will load automatically to open its corresponding **Player** window. Close the **Player** window and open the **Recorder** window by clicking the **Show Recorder** option (usually done by right clicking the icon).
25. The **Exe** tree available in the **Components** section will now list the executable that was just created i.e., **eGTest- 1** as shown in 2.2.1.

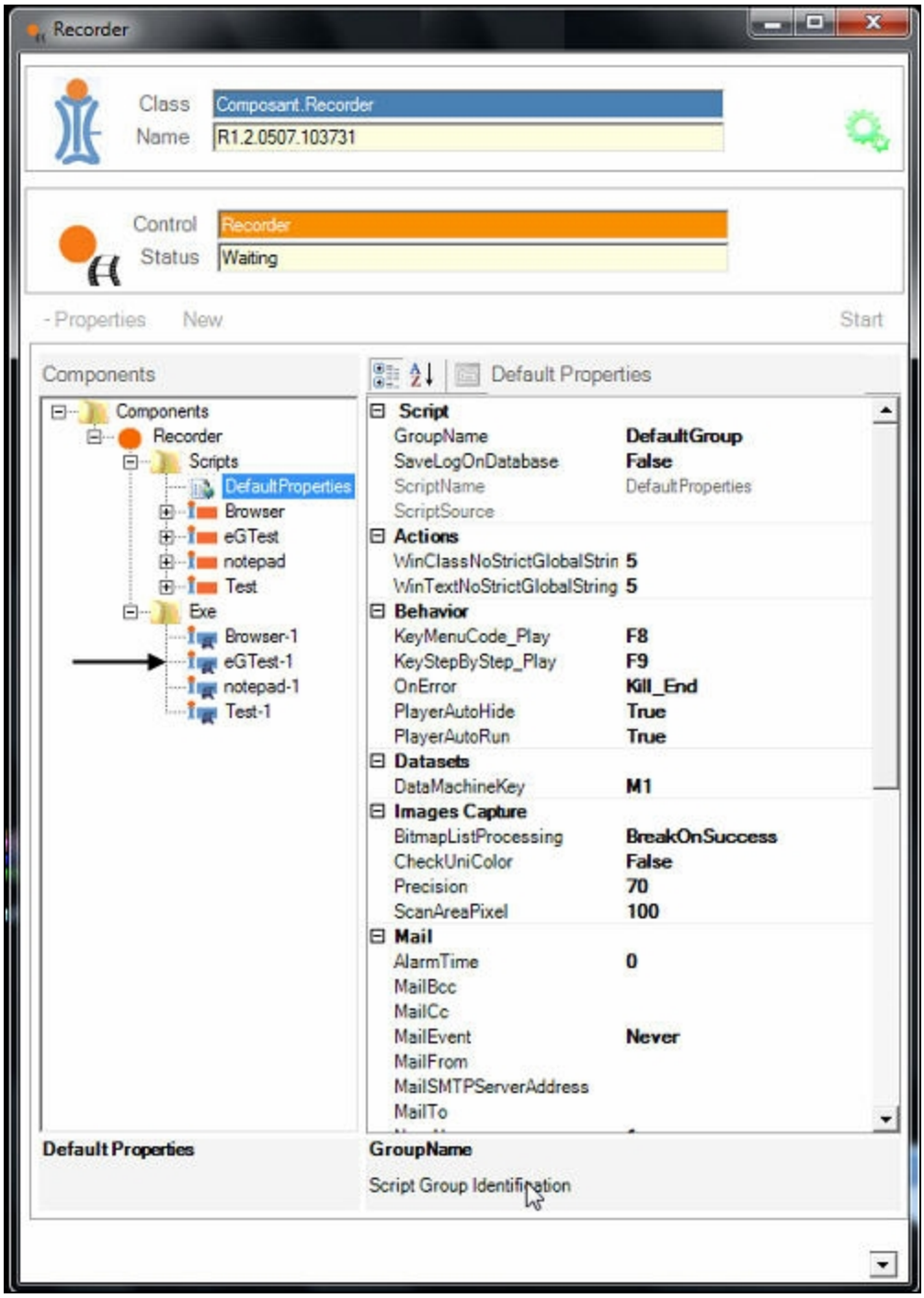

Figure 2.18: Viewing the eGTest-1 in the Exe tree

26. Clicking on the executable **eGTest-1** will load the script and lead you to the **Player** window. Click the **Start** button (see Figure 2.18) to start playing the recorded script.

27. During playback, if any of the transactions could not be completed due to reasons like say for e.g., the image captured is disoriented from its original location or removed, the script will abort and stop immediately. An error message as shown in Figure 2.19 will then appear.

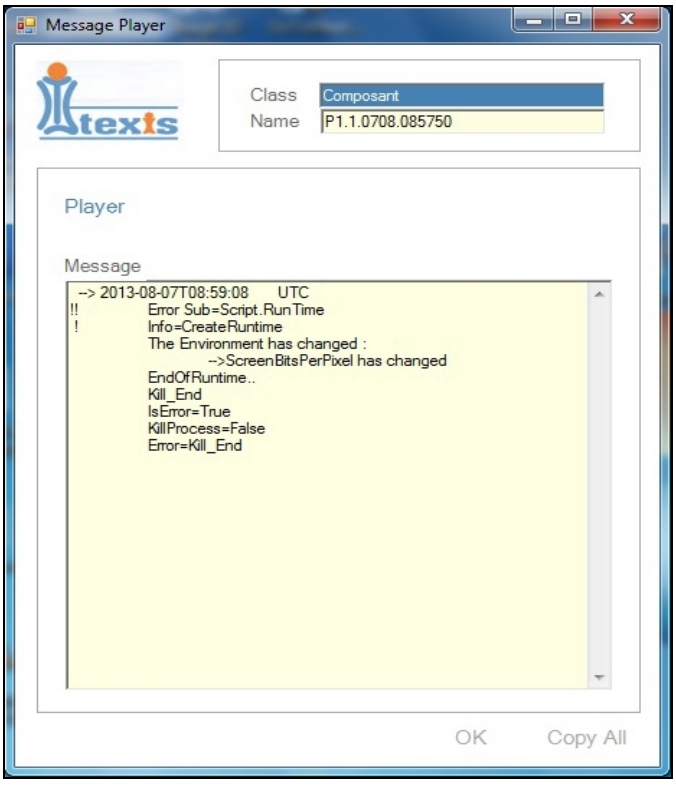

Figure 2.19: The error message displayed in the Player window

28. Once the script is successfully played back, a dialog box displaying the reference time or the total time taken to emulate the transactions will appear. Save the reference time for the script (see Figure 2.20) by clicking the **Yes** button and close the player window.

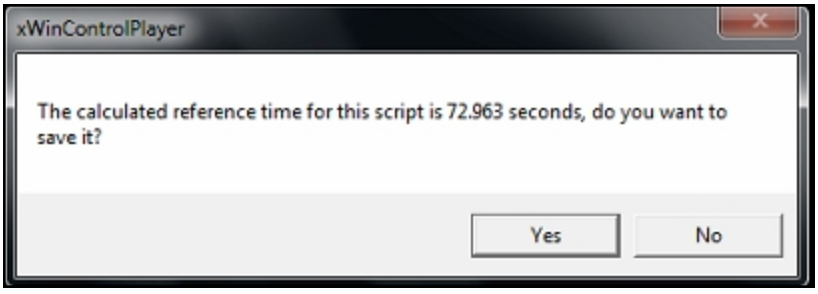

Figure 2.20: The calculated reference time for the script

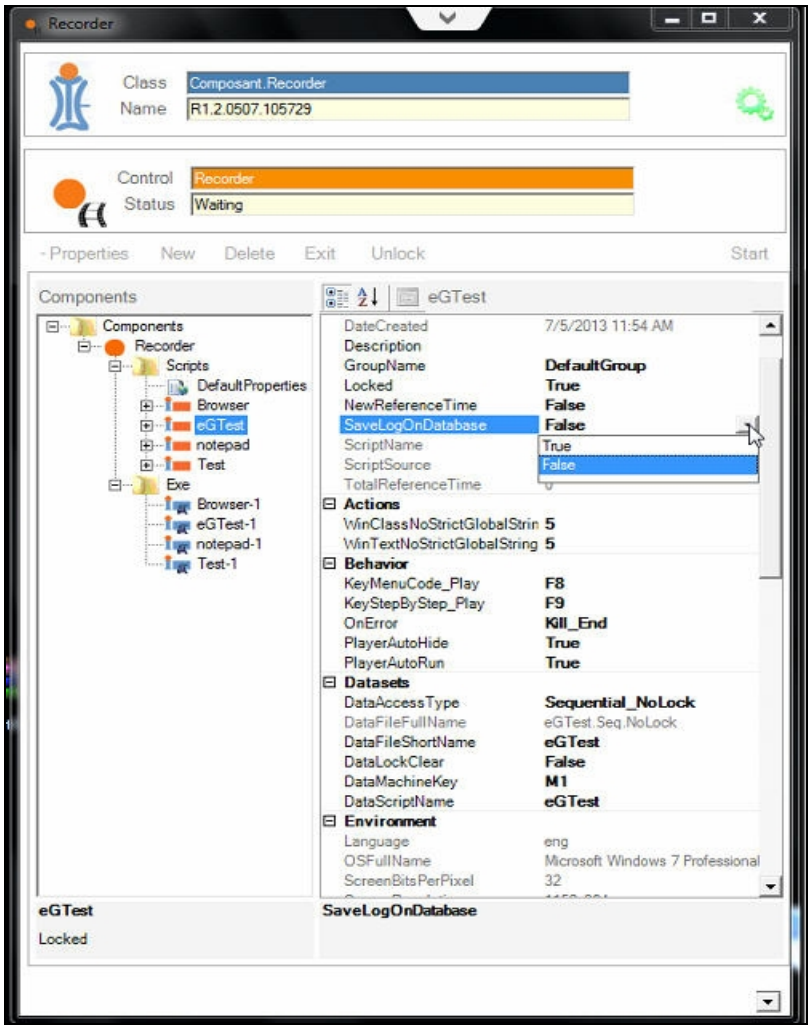

Figure 2.21: Setting the SaveLogOnDatabase field to False

29. By default, the logs created while emulating the transactions will not be stored in the database. To save the logs in a particular location of the database, say for e.g., **XWCF_Logs**, then set the **Savelogondatabase** for the chosen script to **False** as depicted in Figure 2.21. The logs will now be successfully stored in the following location of the database: **C:\Documents and Settings\All Users\Application Data\Itexis\JobQ\M1\XWCF_Logs**

    Note:

    The logs will not be stored in your desired location immediately after the **Savelogondatabase** option is set to False. Instead, the script should be played back once successfully and from the next execution of the script, the logs will be stored in the chosen location which could then be used for client emulation.

30. The script that is generated using the AppsMon for Windows, will be integrated with the eG client emulation tool using the path of the **Log** and **Script** files. The eG client emulation tool checks the response time and availability of the component and reports to the client. The integration of the AppsMon for Windows to work with eG is explained in detail in the following sections.

2.2.2 Troubleshooting

This section discusses about how to identify an error that has occurred during script execution?

Sometimes, scripts may abort automatically during execution, if errors are encountered. Let us say for example, the **eGTest** script has encountered such errors. To identify the errors for further debugging, do the following:

1. Click on the **eGTest- 1** script executable from the **Exe** tree of the **Recorder** window. The context sensitive right panel will display the properties associated with this script. By default, the **On_Error** option available under the **Behavior** section will be **Kill_End**. Double clicking the **Kill_End** option will enable a drop down list. Select the **Learning** option from this list and then click on the **Start** button (see Figure 2.22).

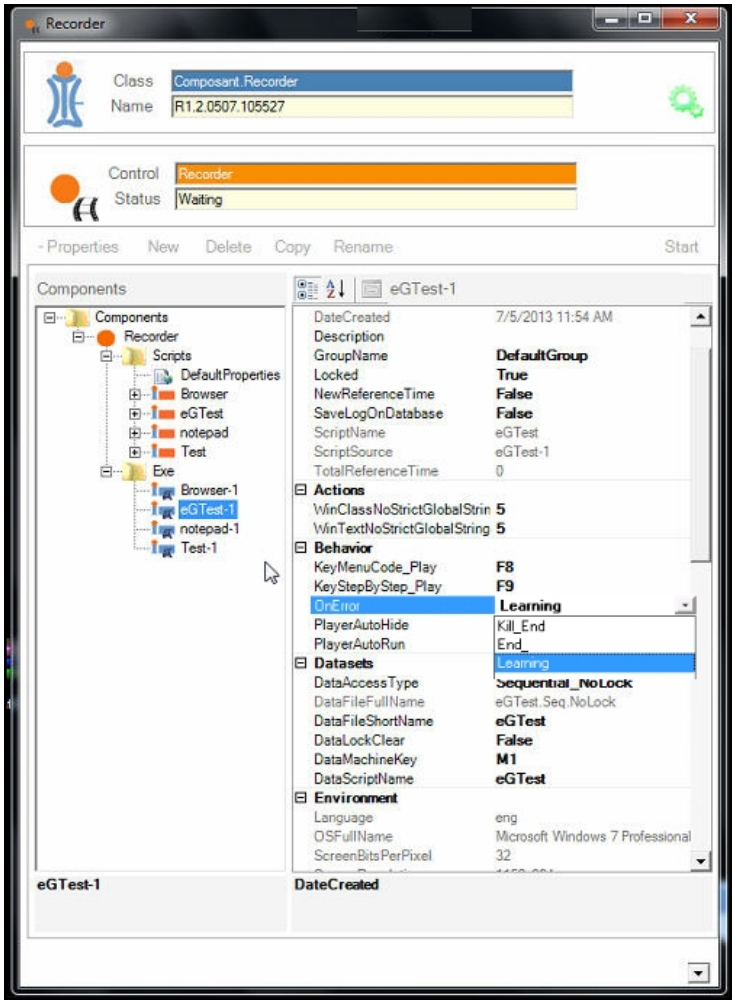

Figure 2.22: Setting the OnError option to Learning through the Recorder window

2. Alternately, you can right click on the **eGTest- 1** in the **Exe** tree and select the **Load** option. The **Player** window will then appear with the **Script Properties**. From here too, you can change the **OnError** option from **Kill_End** to **Learning** and click the **Start** button (see Figure 2.23).

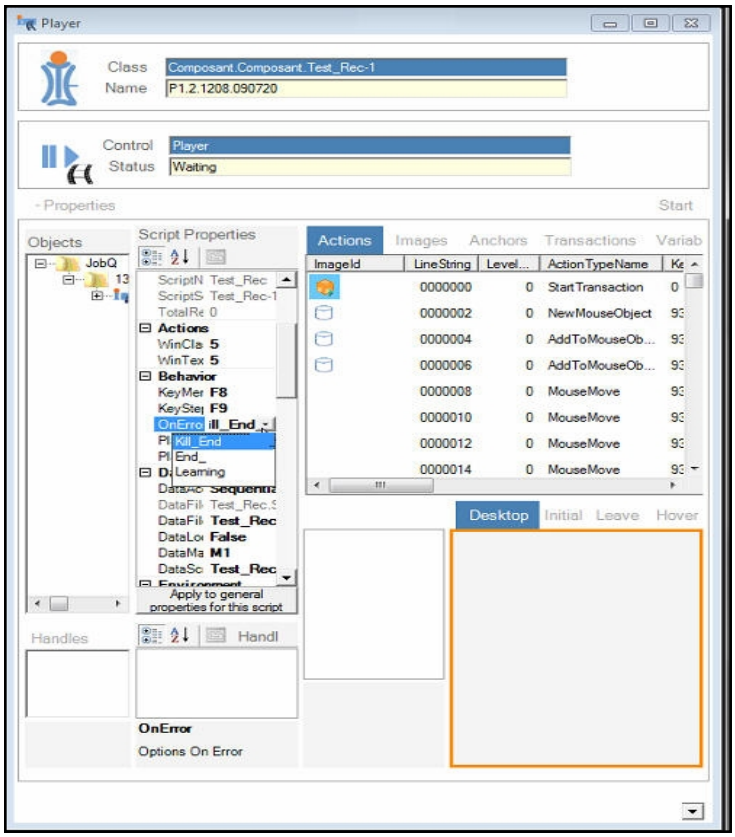

Figure 2.23: Setting the OnError option to Learning through the Player window

3. The script will now be played back and if it encounters any errors, a pop up window will appear displaying the details of the error. The exact location of the error in the VB script i.e., the line at which the error has occurred can be identified from the **Action Line** field of Figure 2.23. Likewise, there may be errors related to the captured images and the actionable events i.e., the associated mouse clicks. In such cases, the pop up window will display the exact error location.

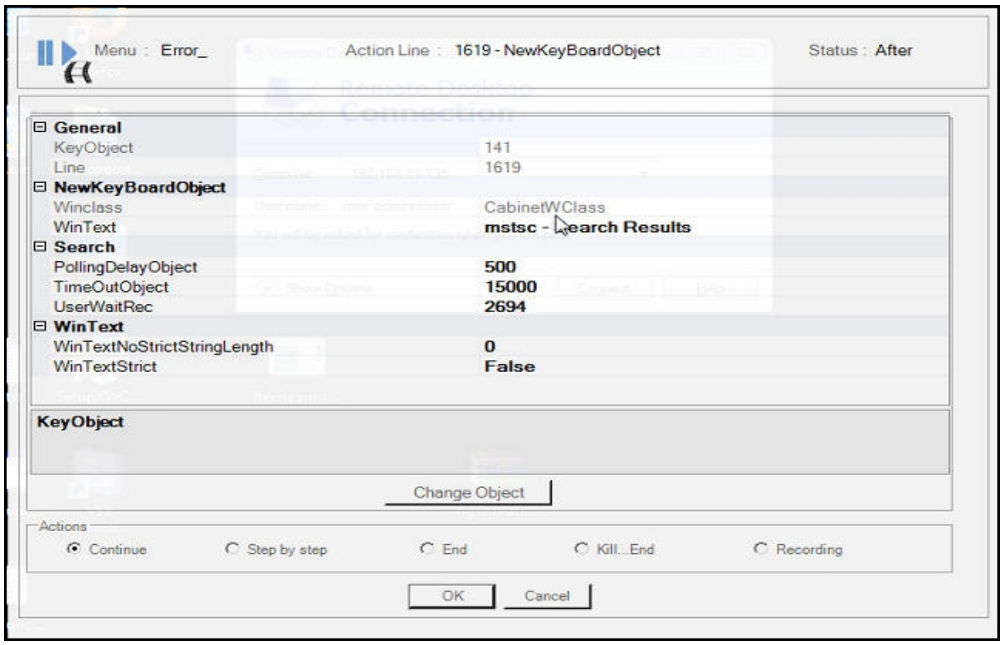

Figure 2.24: The Error pop up window

4. Clicking on the **Ok** button in Figure 2.24 will return you back to the **Player** window, which reports **ExecutionError** in the **Status** text box (see Figure 2.25).

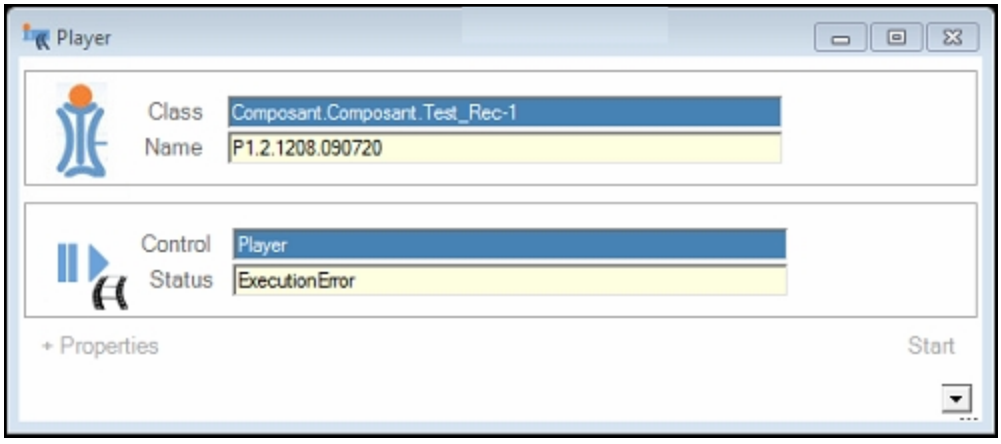

Figure 2.25: The Player window with the error status message

This way, you can identify the errors that were encountered during script execution.

2.2.3 Configuring the eG manager to Work with AppsMon for Windows

The next step is to configure the eG manager to work with AppsMon for Windows. The eG manager supports an Emulated Client component type that has been specifically designed to extract performance metrics from request emulators such as **AppsMon for Windows**.

To configure the eG manager to work with AppsMon for Windows, do the following:

1. Login to the eG administrative interface.
2. Next, add the component to be monitored as an Emulated Client. To do so, follow the menu sequence: Infrastructure -> Components -> Add/Modify. From the **Add/Modify Components** page that appears, select the Emulated Client option from the **Component type** list box (Figure 2.26).

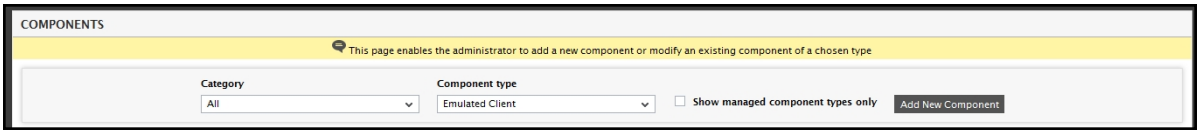

Figure 2.26: Selecting the Component type

3. Then, click on the **Add New Component** button in Figure 2.26 to add a new component of type *Emulated Client*.
4. Provide the IP address and host name of the component in Figure 2.27 that appears. In our example, the component to be monitored is the web server, 192.168.9.194, which hosts the **Eurail** web site. Therefore, provide 192.168.9.194 against the **Host IP/Name** text box and specify a nick name for the chosen component type in the **Nick name** text box. Then, select the external agent that will execute the recorded script, and click the **Add** button therein to register the changes.

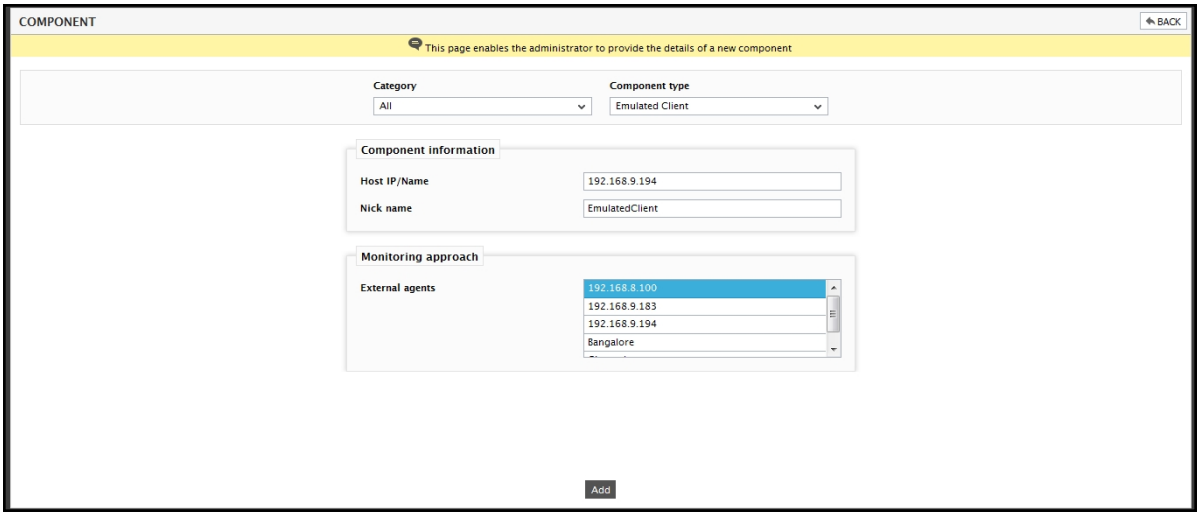

Figure 2.27: Adding a component of type *Emulated Client*

Note:

Only those external agents for which the **CLIENT EMULATION** capability has been enabled will be displayed in the **External agents** list of Figure 2.28. Such agents cannot monitor any other component type.

5. Now, try to sign out of the eG administrative interface by clicking on the signout button at the top right corner of the screen. Upon attempting to sign out, a list of unconfigured tests appears (see Figure 2.28).

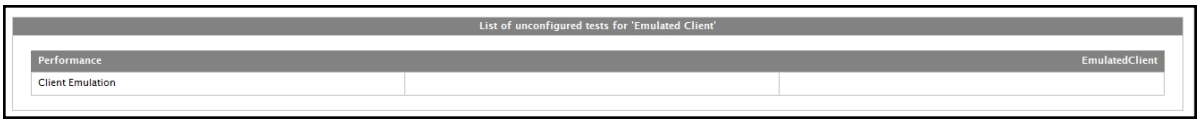

Figure 2.28: List of tests to be configured

6. Figure 2.28 reveals that the **Client Emulation** test associated with the Emulated Client Component type is yet to be configured. This test reports the availability and *response time* of the application being monitored by the *AppsMon for Windows* tool. To configure this test, click on the test name in Figure 2.28. Figure 2.29 will then appear.

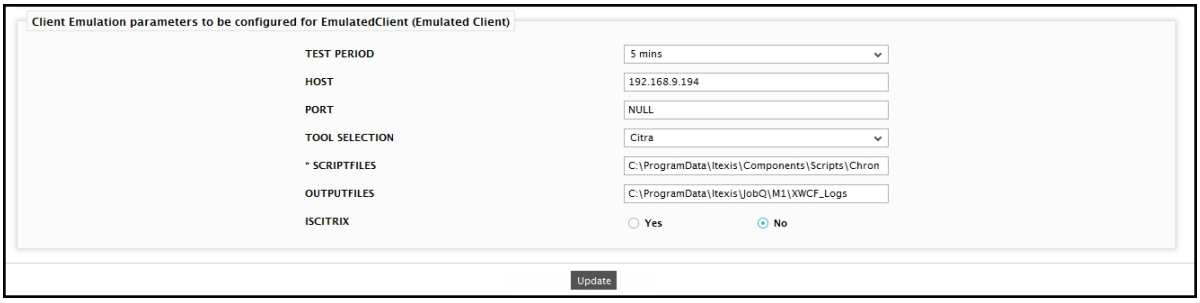

Figure 2.29: Configuring the Client Emulation test

7. Specify the following in Figure 2.29:

   - **TEST PERIOD** – How often should the test be executed
   - **HOST** – The host on which the test will execute. In our example, the test will attempt to extract measures from the host, 192.168.9.194.
   - **PORT** – The port at which the specified **HOST** listens. By default, this is NULL.
   - **TOOL SELECTION** – Select the tool using which the client emulation is to be performed from this text box. By default this is *Citra*. To initiate Itexis - AppsMon for Windows, select **Itexis** option from this list.
   - **SCRIPTFILES** – The full path to the script file that is to be played back for emulating a request to, and extracting metrics from the monitored application. Multiple script files can be provided as a comma-separated list, but all script files should monitor the same application only. In our example, the path to the **eGTest** script has to be specified here.

     Note:

     If the script file resides on another host, then ensure that the location of the script file is mapped to any drive on the measurement host.

   - **OUTPUTFILES** – Enter the full path to the output file that contains the metrics extracted by the specified script file. Here again, multiple output files can be provided as a comma-separated list, but only if multiple script files are also provided.

     Note:

- If **None** is specified here, then the eG system will collect statistics from the default output files associated with each of the specified script files. The default output files will be present in the same location as the respective script files, and will have the same name as the script files. In our example, the value of the **OUTPUTFILES** parameter can remain as **None**.
- While specifying multiple output files, ensure that they are provided in the same order as their corresponding script files in the **SCRIPTFILES** text box.
- If the **SCRIPTFILES** parameter consists of multiple entries and the **OUTPUTFILES** parameter consists of only one, then eG will automatically associate the first script file entry in the **SCRIPTFILES** box with the **OUTPUTFILES** entry. Measures pertaining to the other script files will therefore not be displayed in the eG monitor interface.

➢ **ISCITRIX** – If the specified script emulates a request to a Citrix client then, specify **Yes** here. If not, specify **No**. Our example does not attempt to extract measures from a Citrix client. Therefore, provide **No** here.

8. Next, click on the **Update** button in Figure 2.29 to complete the configuration.
9. Now that **Client Emulation** test has been configured, the eG Enterprise will playback the specified **SCRIPTFILES** according to the chosen **TEST PERIOD**. Whenever the **eGTest** script is played back, the *AppsMon for windows* tool will collect the *Availability* and *Response time* metrics for every timer configured in the script, and will store these details in the corresponding **OUTPUTFILES**. The eG Enterprise will then extract the measures from the **OUTPUTFILES** and display them in the monitor interface.
10. To view the measures in the eG monitor interface, first, **SIGNOUT** of the eG administrative interface.
11. Though the **Client Emulation** test helps you to identify whether slowdowns have occurred while accessing an web site/application, it will not exactly pinpoint you to what has actually contributed to the slowdown - is it the URL components like HTTP, jss etc? or is it the error components? or is it the page load time? In order to know the real reason behind such slowdowns, eG Enterprise provides the **Synthetic Transaction** test. This test is disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : *Agents -> Tests -> Enable/Disable*, pick the desired **Component type**(in our case, *Emulated Client*), set *Performance* as the **Test type**, choose the test from the **DISABLED TESTS** list(see Figure 2.30) .

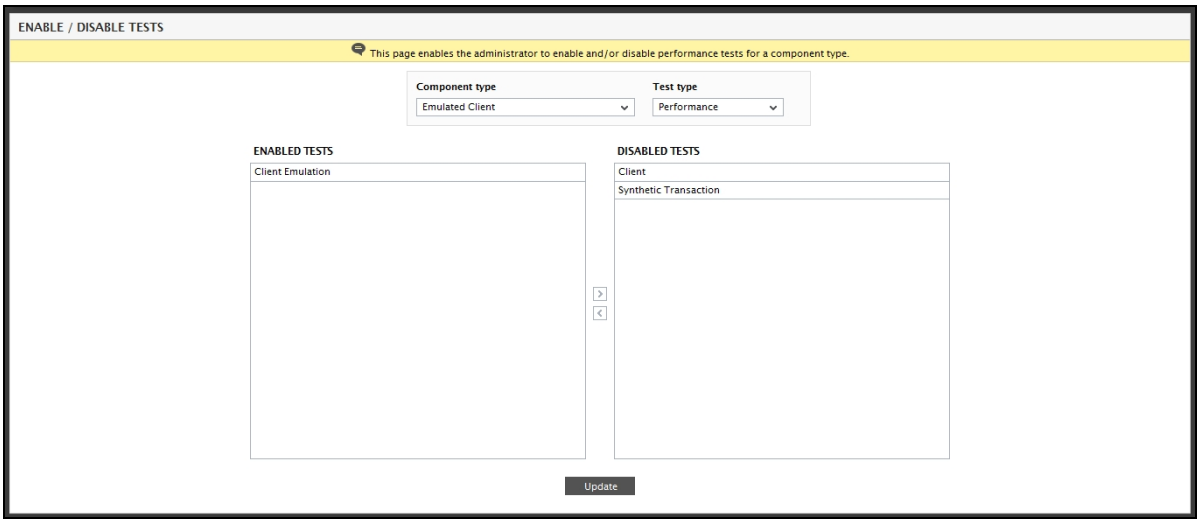

Figure 2.30: Selecting the Synthetic Transaction test from the DISABLED TESTS list

12. Click on the **<** button to move the test to the **ENABLED TESTS** list as shown in Figure 2.31.

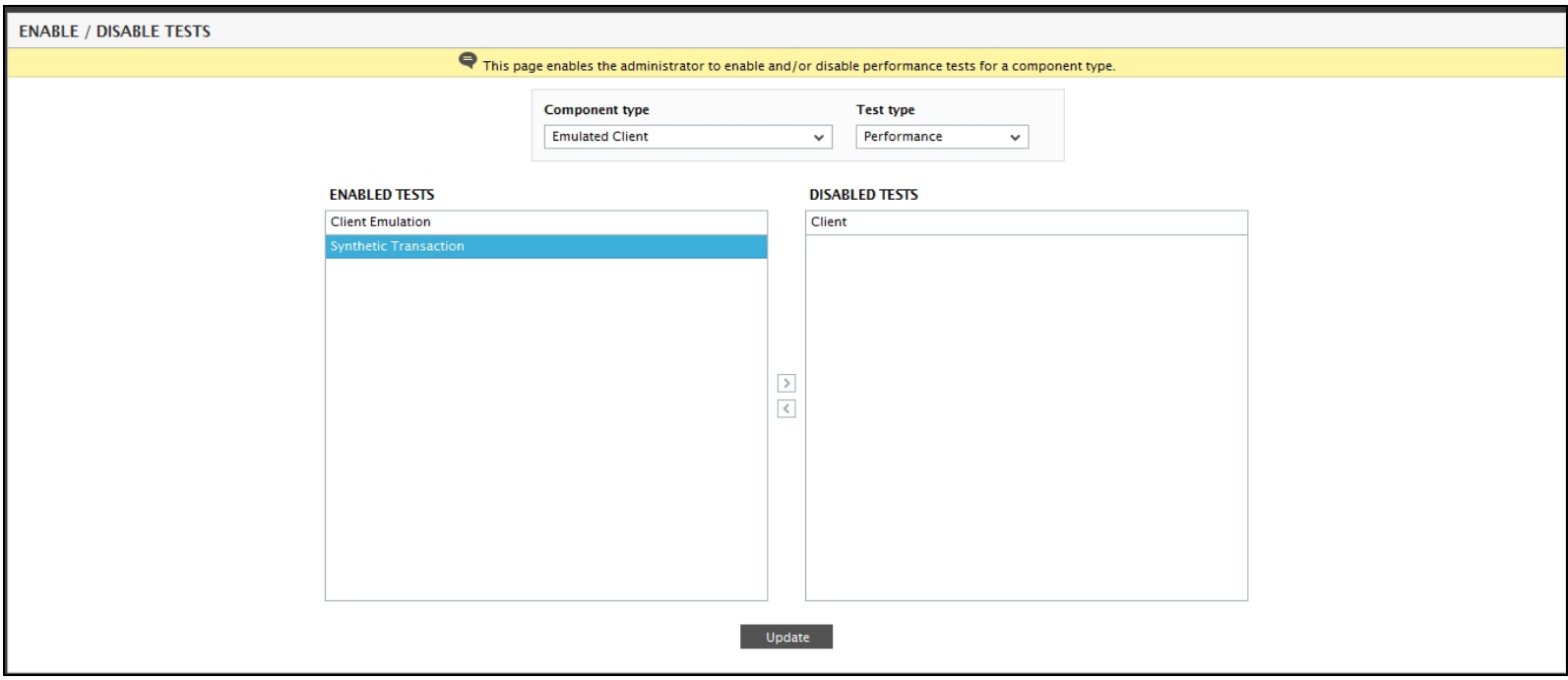

Figure 2.31: Enabling the Synthetic Transaction test

13. Finally, click the **Update** button.
14. Next, proceed to configure the **Synthetic Transaction** test. Figure 2.32 will then appear.

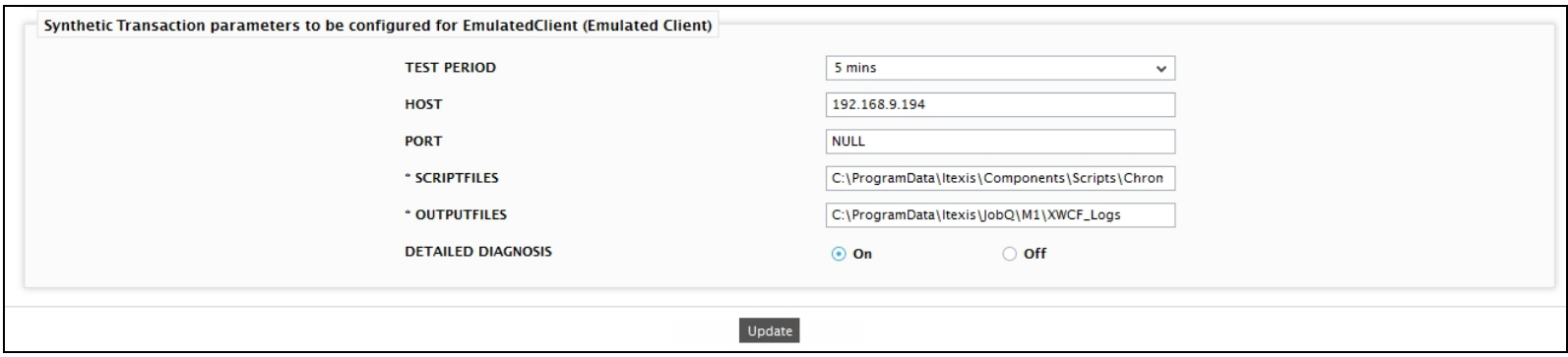

Figure 2.32: Configuring the Synthetic Transaction test

15. Specify the following in Figure 2.32.

- **TEST PERIOD** – How often should the test be executed
- **HOST** – The host on which the test will execute. In our example, the test will attempt to extract measures from the host, 192.168.9.194.
- **PORT** – The port at which the specified **HOST** listens. By default, this is NULL.
- **SCRIPTFILES** – The full path to the script file that is to be played back for emulating a request to, and extracting metrics from the monitored application/website. Multiple script files can be provided as a comma-separated list, but all script files should monitor the same application only. In our example, the path to the **eGTest** script has to be specified here.

  Note:

  If the script file resides on another host, then ensure that the location of the script file is mapped to any drive on the measurement host.

- **OUTPUTFILES** – Enter the full path to the output file that contains the metrics extracted by the specified script file. Here again, multiple output files can be provided as a comma-separated list, but only if multiple script files are also provided.

  Note:

  - If **None** is specified here, then the eG system will collect statistics from the default output files associated with each of the specified script files. The default output files will be present in the same location as the respective script files, and will have the same name as the script files. In our example, the value of the **OUTPUTFILES** parameter can remain as **None**.
  - While specifying multiple output files, ensure that they are provided in the same order as their corresponding script files in the **SCRIPTFILES** text box.
  - If the **SCRIPTFILES** parameter consists of multiple entries and the **OUTPUTFILES** parameter consists of only one, then eG will automatically associate the first script file entry in the **SCRIPTFILES** box with the **OUTPUTFILES** entry. Measures pertaining to the other script files will therefore not be displayed in the eG monitor interface.

- **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be

configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the **DETAILED DIAGNOSIS** capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

16. Next, click on the **Update** button in 2.2.3 to complete the configuration.
17. Now that **Synthetic Transaction** test has been configured, the eG Enterprise will playback the specified **SCRIPTFILES** according to the chosen **TEST PERIOD**. Whenever the script is played back, the *AppsMon for windows* tool will collect all the relevant metrics for every timer configured in the script, and will store these details in the corresponding **OUTPUTFILES**. The eG Enterprise will then extract the measures from the **OUTPUTFILES** and display them in the monitor interface.
18. To view the measures in the eG monitor interface, navigate to the layer model of the *Emulated Client* component in the eG monitor interface.

2.2.4 Starting the External Agent

Refer to the *eG Installation Guide* for an elaborate procedure on starting the eG agent on Windows environments. In our example, the external agent, 192.168.8.20, which is associated with the EmulatedClient component (192.168.9.194), will have to be started.

Before starting the external agent assigned to an Emulatedclient, ensure that the eGurkhaAgent service is allowed to interact with the desktop. To do this, do the following:

1. Open the **Services** window by following the menu sequence: Start -> Settings -> Control Panel -> Component Services.
2. Right-click on the *eGurkhaAgent* service therein, and select the **Properties** option.

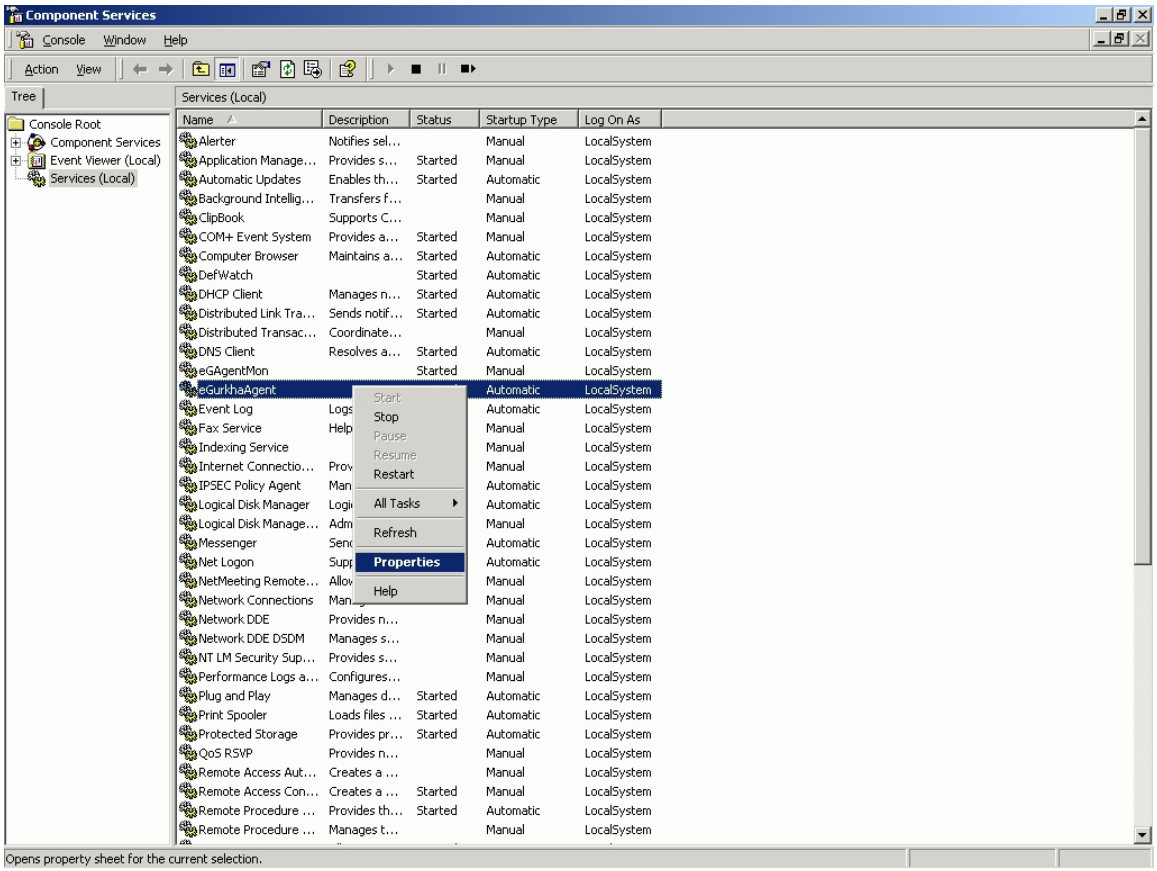

Figure 2.33: Selecting the Properties option

3. Click on the **Log On tab** in the **Properties** dialog box (see Figure 2.33), and select the **Allow service to interact with desktop** check box.

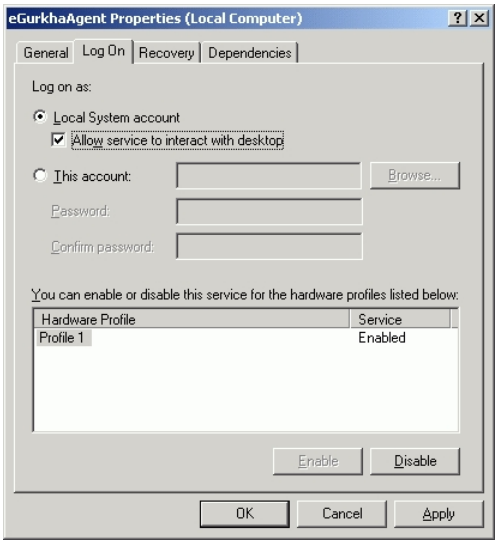

Figure 2.34: Allowing the service to interact with the desktop

4. Finally, click on the **Apply** button, and then the **OK** button.

2.2.5 Viewing the Measures

To view the measures reported by the tests associated with the *Emulated Client* component, do the following:

1. Login to the eG monitor interface.
2. From the **Hosts/Applications** tile, select the **Components** option. Once the **COMPONENTS** page appears, select the *Emulate Client* option from the **Type** list therein, so as to view the current state of all components of type *Emulated Client*. Then, click the **Submit** button.

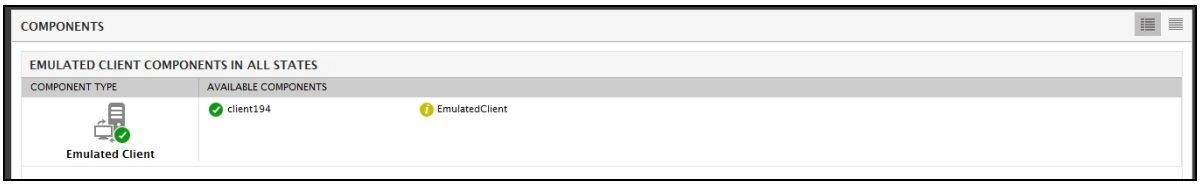

Figure 2.35: The Independent Components page

3. The Emulated Client component that we had configured previously will then be listed (see Figure 2.35). Click on the component to view its layer model, tests, and measurements (see Figure 2.36).

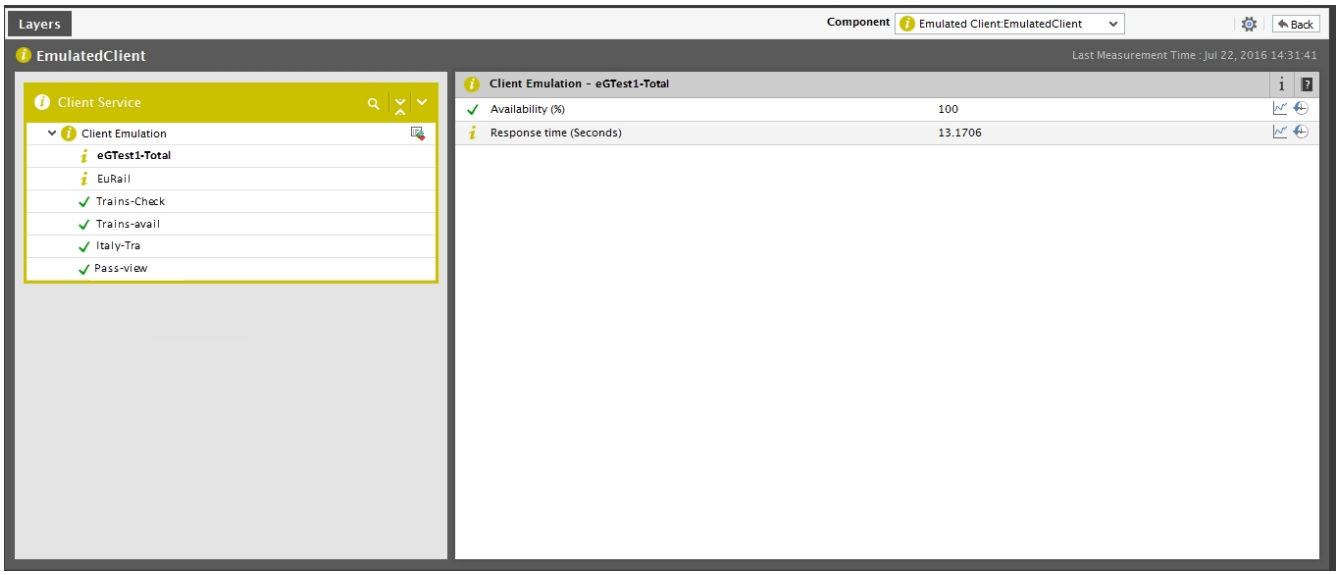

Figure 2.36: Viewing the layer model, tests, and measurements of the Emulated Client

4. Figure 2.36 will list the tests associated with the Emulated Client component and the measures associated with the Client Emulation test. From Figure 2.36, it is evident that all the transactions that were recorded for the **eGTest** script will appear as descriptors of the Client Emulation test.
5. Besides the transactions that were recorded for the **eGTest** script, Figure 2.36 also displays an additional descriptor *eGTest-1 Total*. This transaction tracks the time taken by every transaction of the script, calculates the sum of the duration of all the transactions, and provides this sum as the response time of the monitored application. Figure 2.36 reveals the response time returned by the *eGTest-1 Total*

descriptor. This is the total time taken to access the **Eurail website**, request for **High-speed train** information, check for the trains available and receive a response from the web server.

The table below briefly explains the *Availability* and *Response time* measures returned by the **Client Emulation** test.

Measure Name	Description
Availability (%)	This relates to the availability of the page, i.e., whether a page has downloaded successfully or not. In Windows Applications, this relates to whether the step has executed properly or not. It is expressed in terms of percentage (%) – 100% if the download is a success and 0% if not. A variation of this is for the availability of all the steps in total - if even 1 step has failed, the total availability is 0 %.
Response time (Secs)	This relates to the time taken for the page to download or the step to execute. If the total availability is 0 % then the response time is set as 'unknown', else it is expressed in **secs**.

6. When you click on a descriptor of the **Synthetic Transaction** test, Figure 2.37 appears. Using the measures reported by this test, administrators can easily identify where exactly has the slowdown occurred while accessing the specific web site/application.

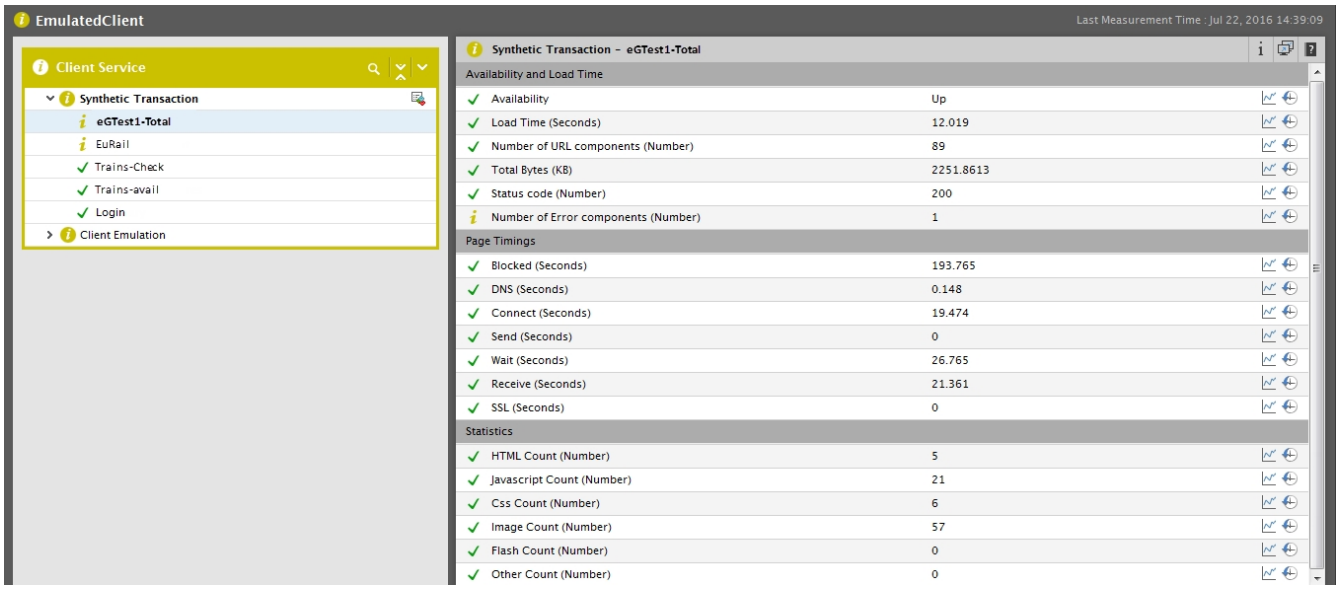

Figure 2.37: Viewing the measures reported by the Synthetic Transaction test

The table below briefly explains the measures returned by the **Synthetic Transaction** test.

Measure Name	Description
Availability (%)	This relates to the availability of the page, i.e., whether a page has downloaded successfully or not. In Windows Applications, this

Measure Name	Description
	relates to whether the step has executed properly or not. It is expressed as *Up* and *Down* – *Up* if the download is a success and *Down* if not. A variation of this is for the availability of all the steps against the *Summary* descriptor - if even 1 step has failed, the *Availability* is termed as *Down*.
Load time (Seconds)	This relates to the time taken for the page to download or the step to execute. A low value is desired for this measure.
Number of URL Components (Number)	Indicates the number of URL components available in this page .
Total Bytes (KB)	Indicates the number of bytes used for accessing this page.
Status code (Number)	Indicates the current status of this page.
Number of Error components (Number)	Indicates the number of error components detected in this page.
Blocked (Seconds)	Indicates the time spent by this page in a queue waiting for a network connection. If the time taken by this measure is too long, then, administrators should check the network connectivity of their infrastructure.
DNS (Seconds)	Indicates the amount of time spent for a DNS lookup on this page.
Connect (Seconds)	Indicates the time taken to establish a TCP connection. A high value is an indication of network congestion.
Send (Seconds)	Indicates the time required to send the HTTP request to the server of this web page. The value returned by this measure mostly depends on the size of the data sent, network congestion and proximity of the client ot the server.
Wait (Seconds)	Indicates the amount of time that is spent while waiting for an initial response upon a request.
Receive (Seconds)	Indicates the amount of time required to read the entire response from the server. The value returned by this measure mostly depends on the size of the data received, network congestion and proximity of the client ot the server.
SSL (Seconds)	Indicates the time taken to establish an SSL handshake to this web page.
HTML Count (Number)	Indicates the number of HTML components in this web page.
Javascript Count (Number)	Indicates the number of javascript components in this web page.
Css Count (Number)	Indicates the number of CSS components in this web page.
Image Count (Number)	Indicates the number of images in this web page.
Flash Count (Number)	Indicates the number of flash messages in this web page.
Other Count (Number)	Indicates the number of other components in this web page.

2.3 Client Emulation for a Citrix Application

The second example involves a **Citrix Receiver** client using which you can access XenDesktops or XenApp servers in the target environment. The script to be built should access the server/desktop, and should try to open the notepad application published on it. In addition, the script should key in a few words in the notepad and close the application without saving the changes.

2.3.1 Building a Script File

Let us now begin script building.

1. Create a new script named **Citrix** using steps 1-7 explained in Section **2.2.1**.
2. Next, begin capturing the images, clicks, and key strokes that form part of the user transactions that are to be emulated. Since the **Citrix Receiver** client has to be accessed through the web, open a web browser and enter the URL of the web interface i.e., the **Citrix StoreFront**. In our case, the URL is http://192.168.11.151/Citrix/StoreWeb.
3. The **Citrix** Receiver client login page will then appear as shown in Figure 2.38. Proceed to capture the image of the **Citrix Receiver** in Figure 2.38 as your first transaction.

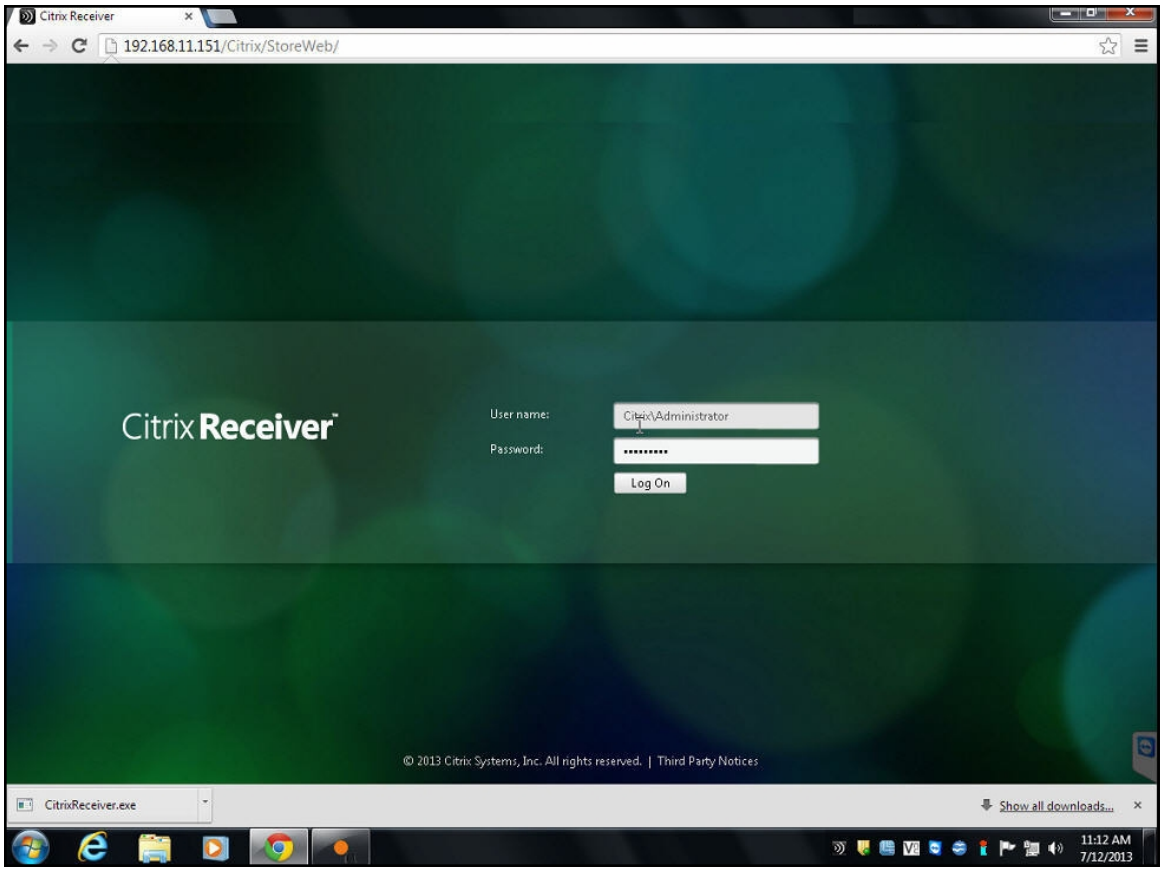

Figure 2.38: The Citrix Receiver client login page

4. Place the cursor on the **Citrix Receiver** and enable the image capturing by pressing the F8 key. The **Menu** pop up window will then appear as shown in Figure 2.9.
5. By default, the **Bitmaps**, **OCR**, **Anchors** option will be selected in the **Menu** pop up window (see Figure 2.9). Pressing the **Enter** key will lead you to the **Capture Bitmap/OCR/Transactions** window as shown in Figure 2.39.
6. This window allows you to save the captured **Citrix Receiver** image as a transaction. By default, the image that is captured will appear in the **Capture Bitmap** section. This capture will be associated with the **Shape** option of the **Capture Type** section and the **MouseMoveImage** option of the **Mouse Action** section, by default. Typically, when you select the **Shape** option, the AppsMon for Windows recognizes the capture as an image and saves the capture in **BMP** format. Likewise, you can select any of the option from the **Mouse Action** field based on your requirement.
7. Alternately, if you select the **OCR** option from the **Capture Type** section, the AppsMon for Windows recognizes the capture as a text and automatically displays the captured text in the **Value** text box of the **OCR** section. Suppose if the capture is not successful, then instead of the captured text, **Empty** will be displayed in the **Value** text box. This way, you can identify that your capture was not successful and you can either capture the text once again or adjust the coordinates of the capture – i.e., in our case the **Citrix Receiver** by moving the '+' or '-' keys in the **Capture Bitmap** section (see Figure 2.39).
8. To save the captured **Citrix Receiver** as a transaction, enter the name of the transaction in the **New Transaction Name** text box of the **Transactions section** and press the **Enter key**.

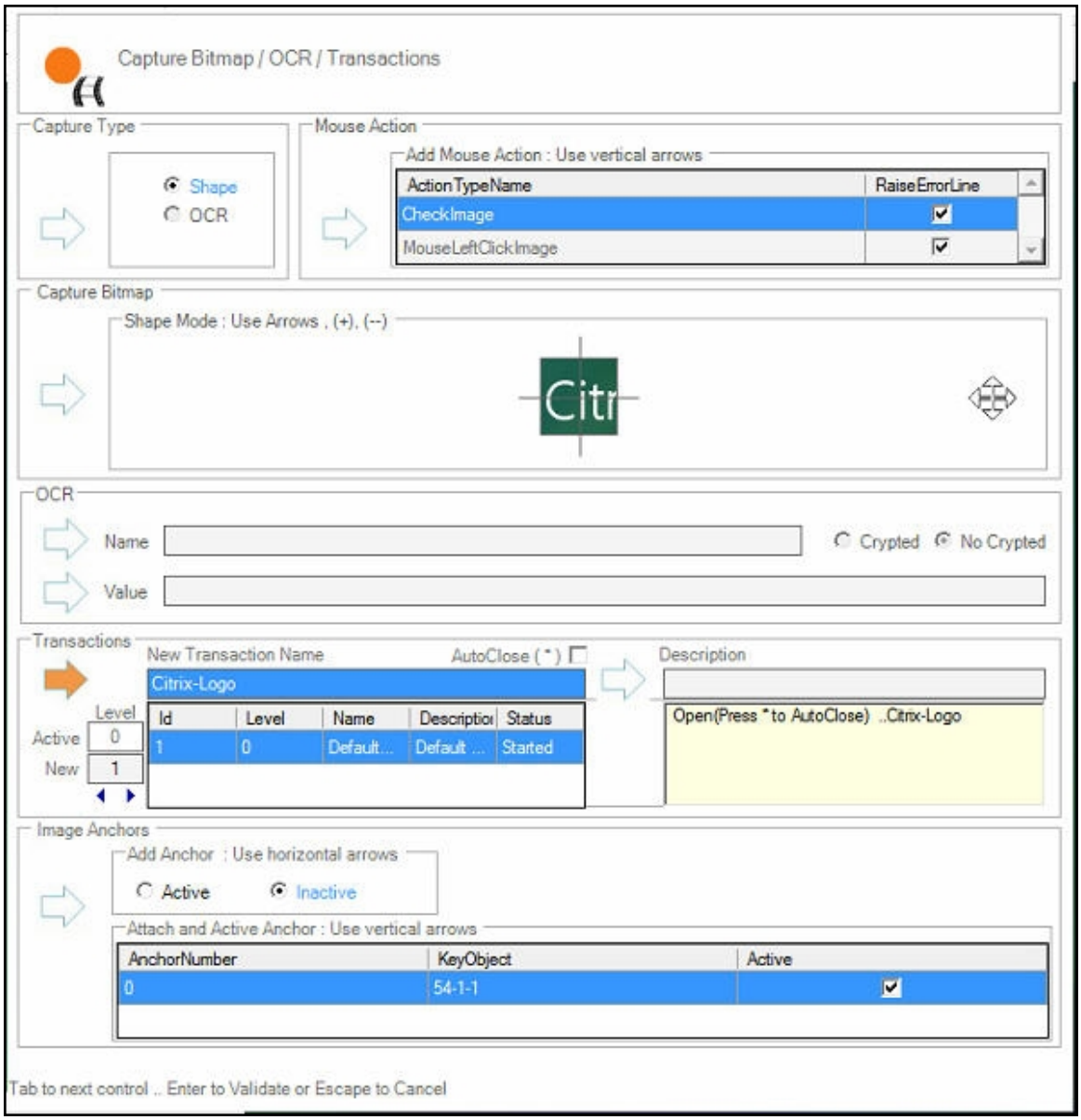

Figure 2.39: Capturing the image of the Citrix Receiver

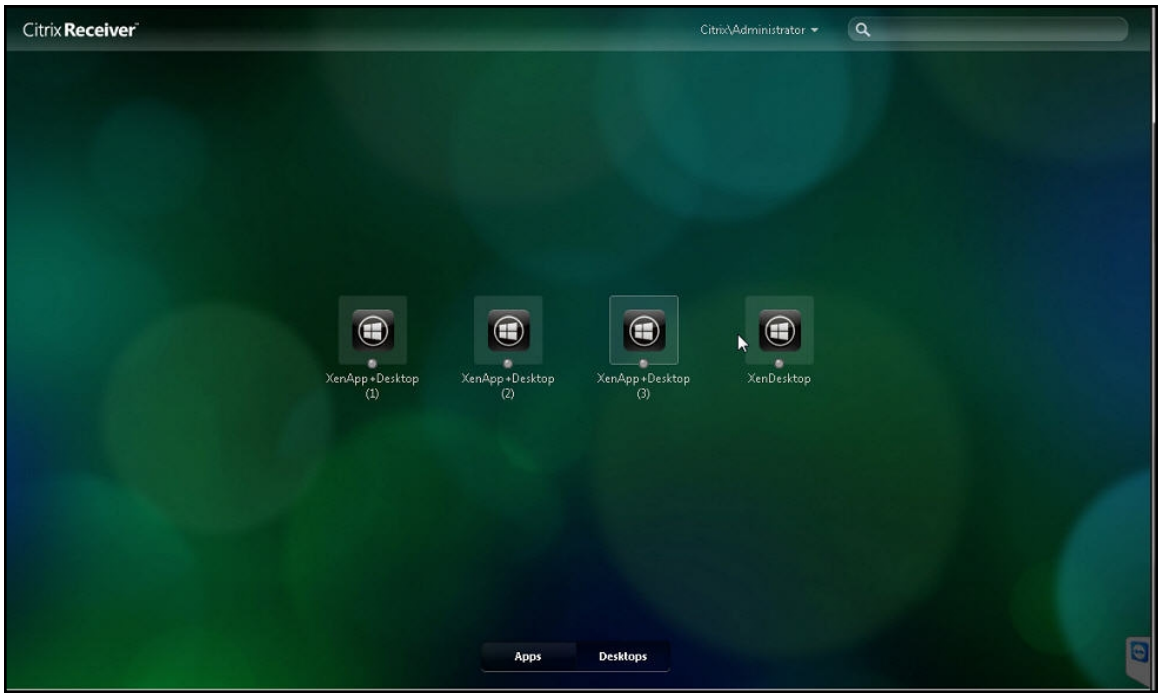

Figure 2.40: The Citrix Receiver client home page

9. Now, to record your second transaction, login to the **Citrix Receiver** client with valid **User Name** and **Password** credentials. In Figure 2.40 that appears, capture the image above the **XenDesktop** by following the steps mentioned above.

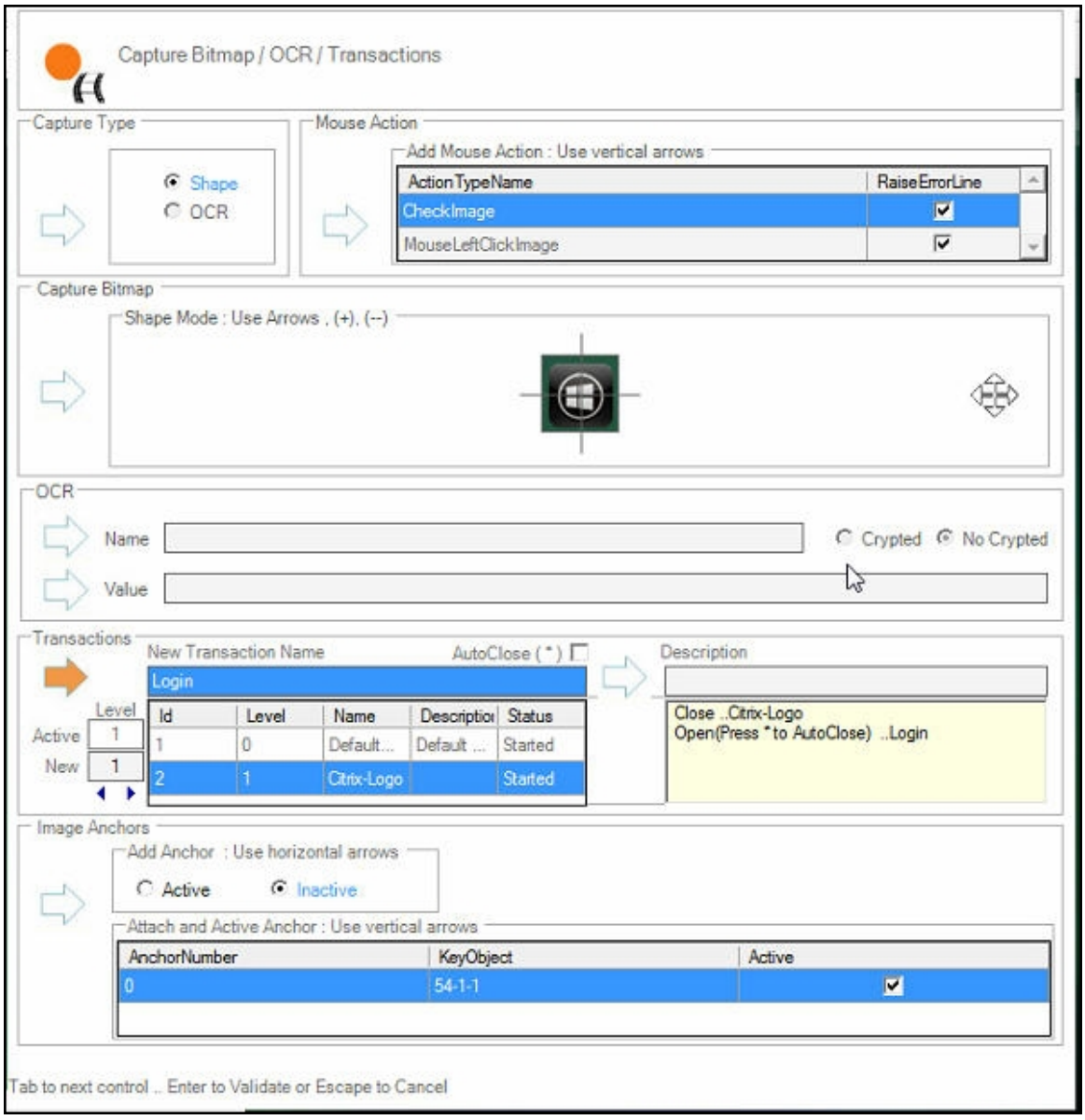

Figure 2.41: Capturing the image of the XenDesktop

10. Clicking on the image above the **XenDesktop** will lead you to Figure 2.42.

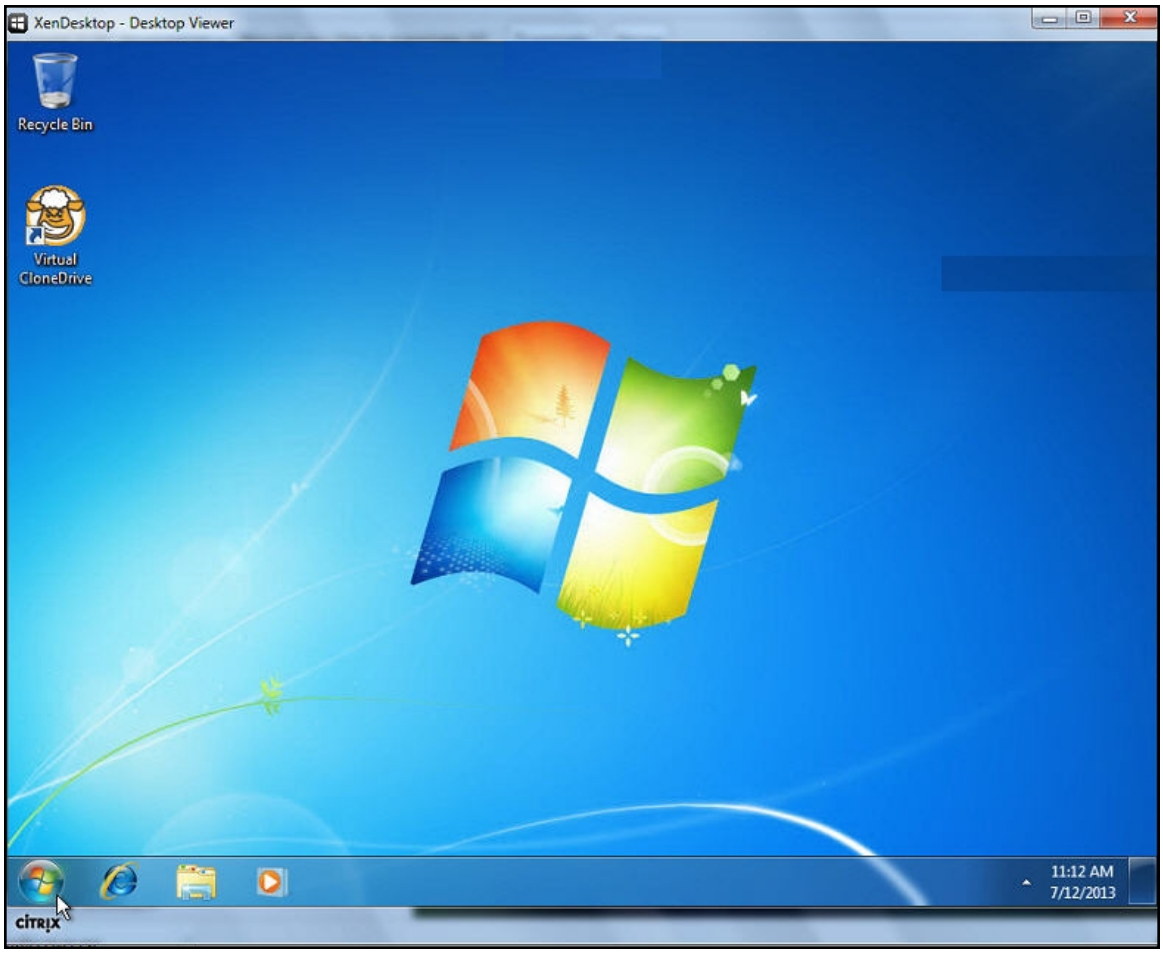

Figure 2.42: The XenDesktop - Desktop Viewer page

11. Try capturing the **Start** button in Figure 2.42 as your next transaction by following the steps 4-8 described above (see Figure 2.43).

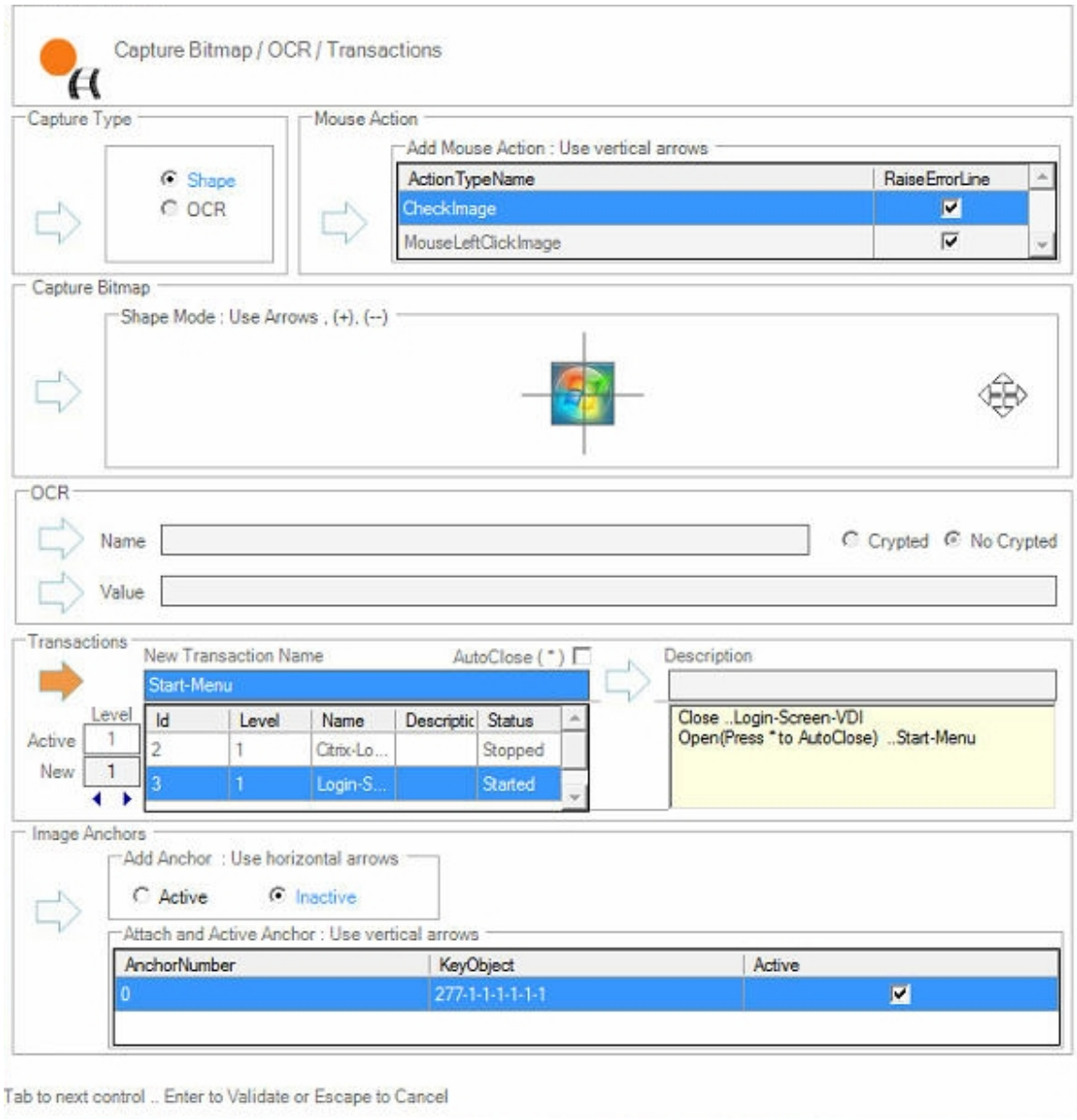

Figure 2.43: Capturing the image of the Start icon

12. Now, open a **Notepad** application published on the XenDesktop by following the menu sequence: *Start->All Programs->Accessories->Notepad* (see Figure 2.42).
13. To capture the **Notepad** icon as your next transaction, follow the steps 4- 8 described above (see Figure 2.44).

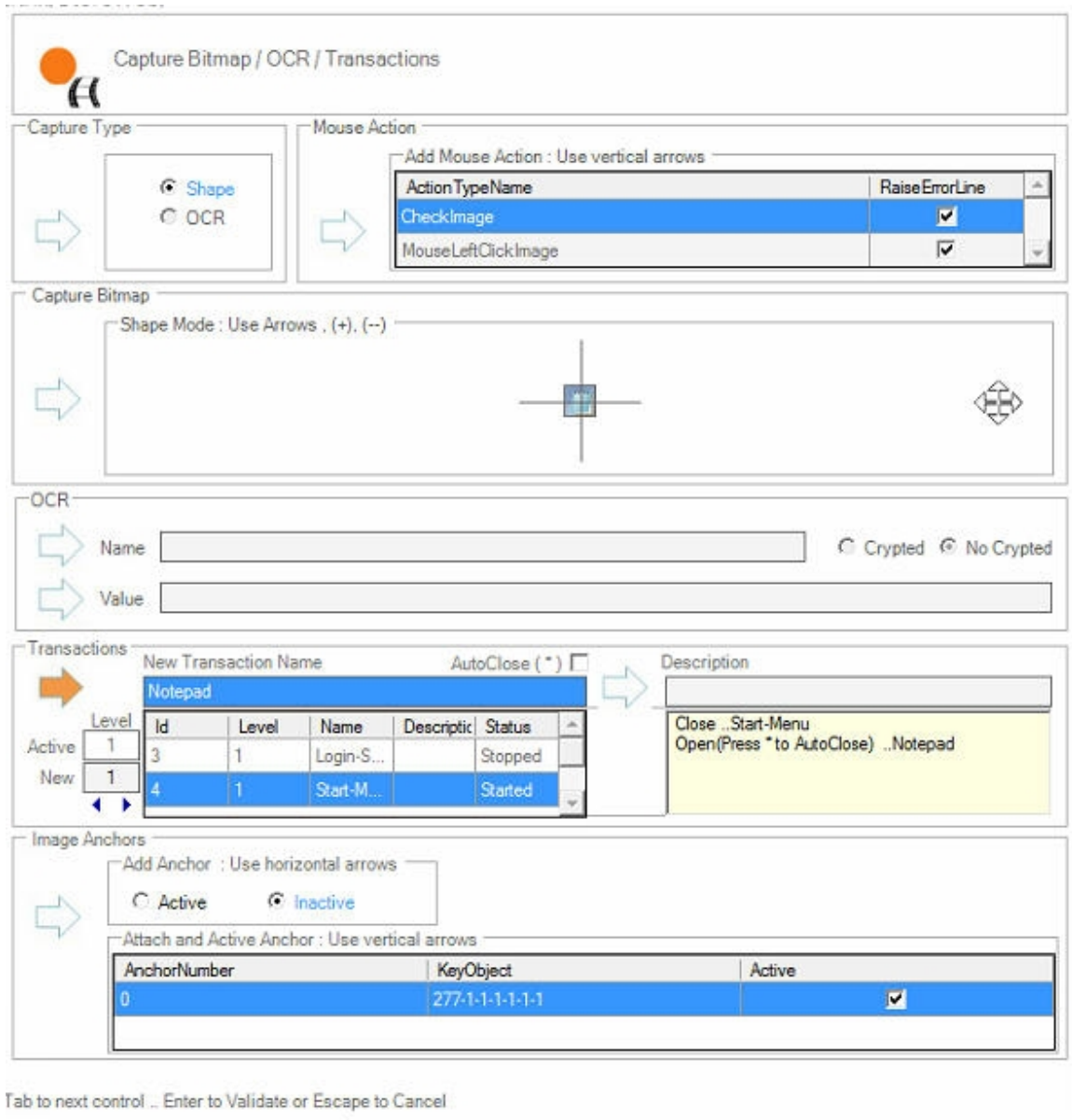

Figure 2.44: Capturing the Notepad icon

14. Now key in a few words in the **Notepad** and close the application without saving the message. Then, logging out of the **XenDesktop** will lead you back to the home page of the **Citrix Receiver** client.
15. To exit the **Citrix Receiver** client, select the **Log Off** option that appears when you click on the down arrow available near the **Citrix\Administrator** in the home page of the **Citrix Receiver** client (see Figure 2.45). To capture this as your next transaction, follow the steps 4- 8 described above.

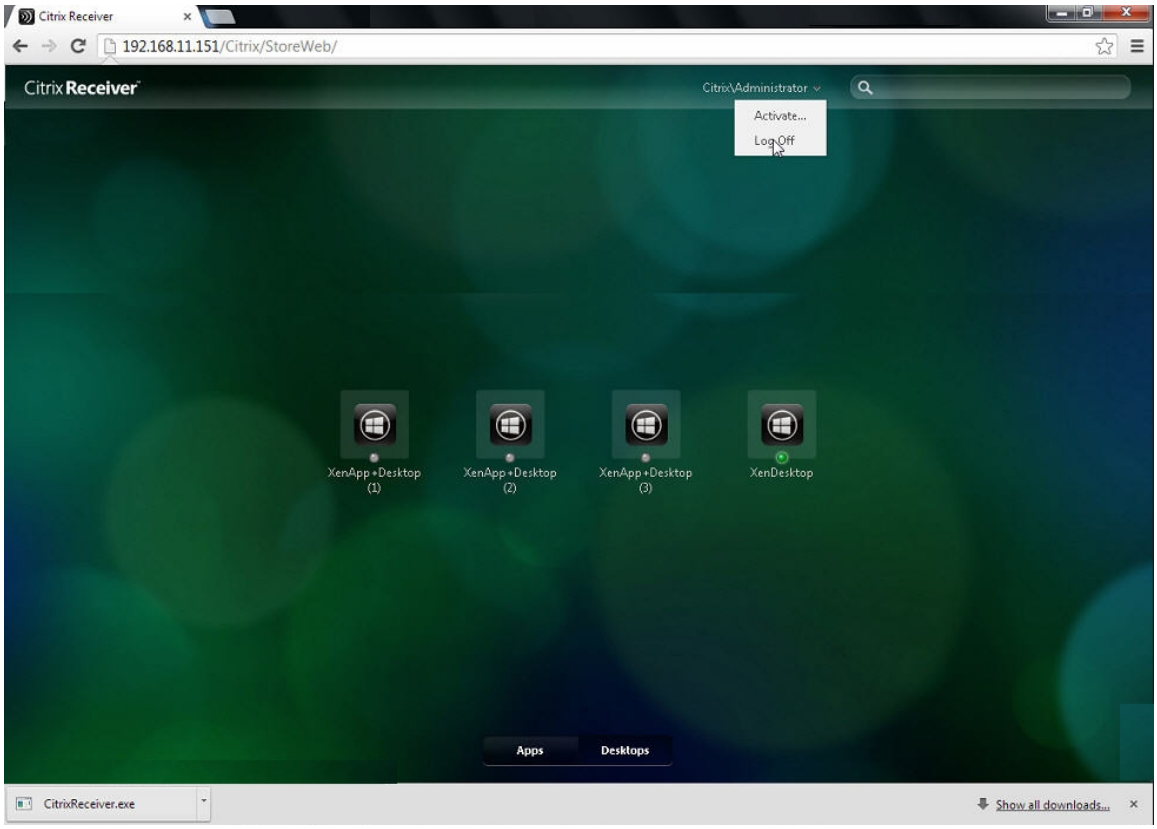

Figure 2.45: The Citrix Receiver administrator page

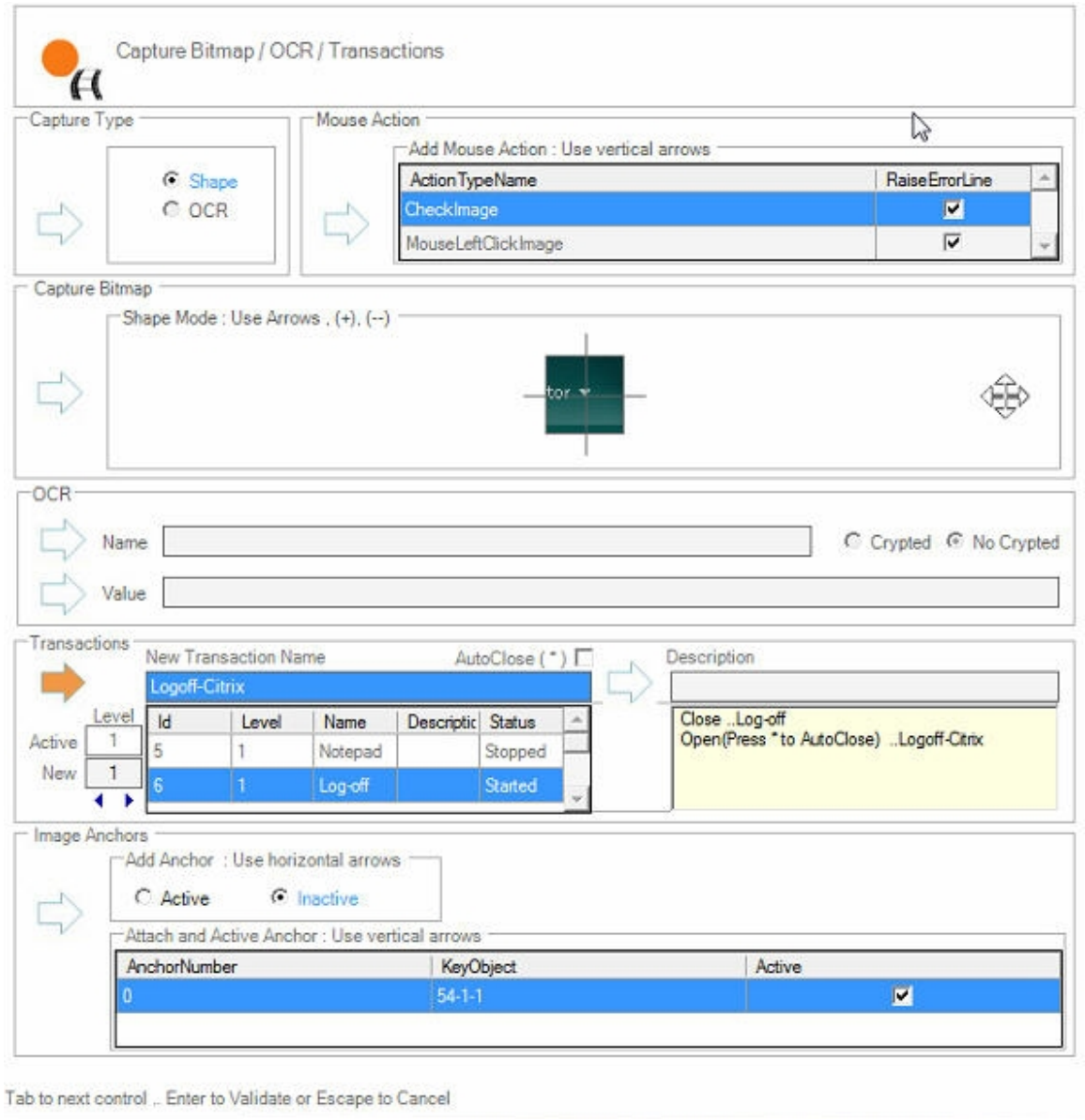

Figure 2.46: Capturing the image of the drop down icon

16. Once you have saved all the required transactions, you can stop the recording by selecting the **Stop Recording** option from the **Menu** pop up window that appears when you press the **F8** key (see Figure 2.47).

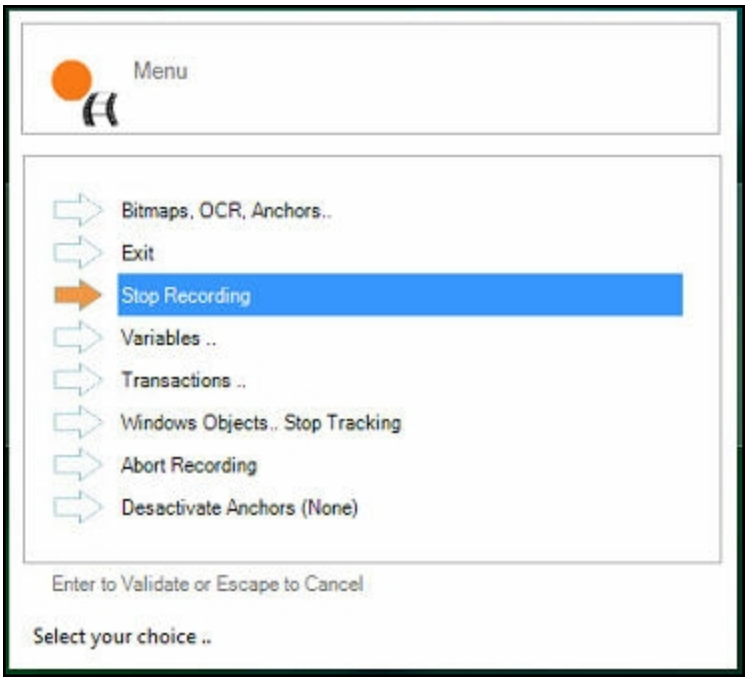

Figure 2.47: Stopping the Recording

Once the recording is stopped, a callout message stating **Session Hooked** will appear above the icon. A **Create Script** window will now appear which will start compiling the script automatically. Once the script has been built successfully, a message stating **Build is Success** will appear in the **Create Script** window and an executable of the script will be created – in our example this will be Citrix-1.exe (see Figure 2.48). The script that is recorded will be stored in the following location:

C:\Documents and Settings\All Users\Application Data\Itexis\Components\Scripts

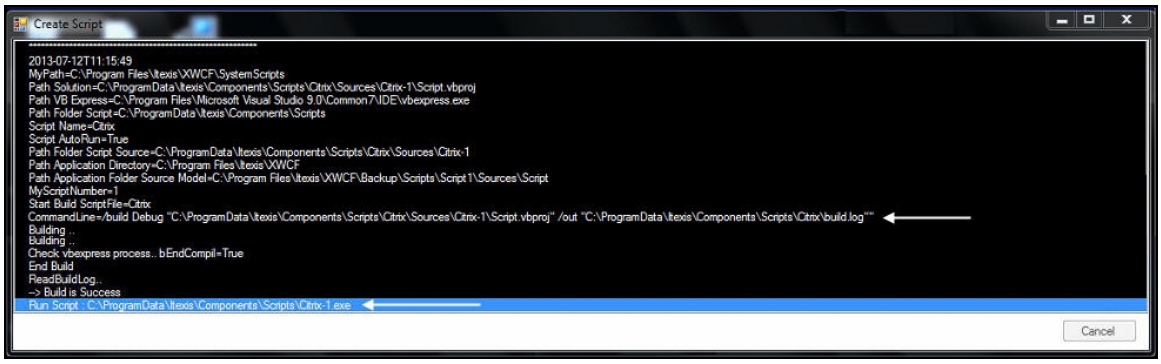

Figure 2.48: The Create Script window showing that the script was built successfully

17. Now, the script will load automatically to open its corresponding **Player** window. Close the **Player** window and open the **Recorder** window by clicking the **Show Recorder** option (usually done by right clicking the icon).

18. The **Exe** tree available in the **Components** section will now host the executable that was just created i.e.,

Citrix- 1 (see Figure 2.49).

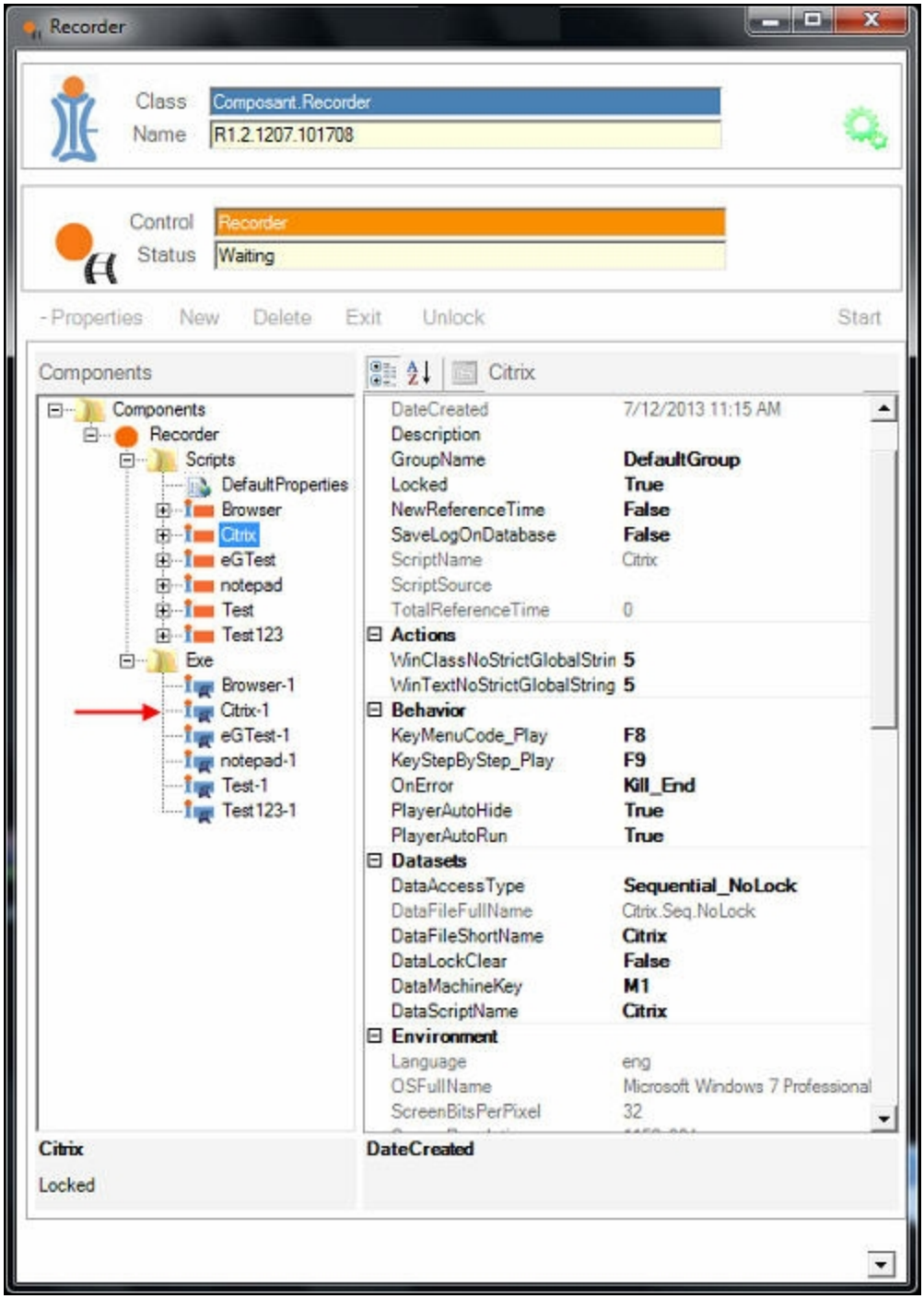

Figure 2.49: Viewing the Citrix-1 in the Exe tree

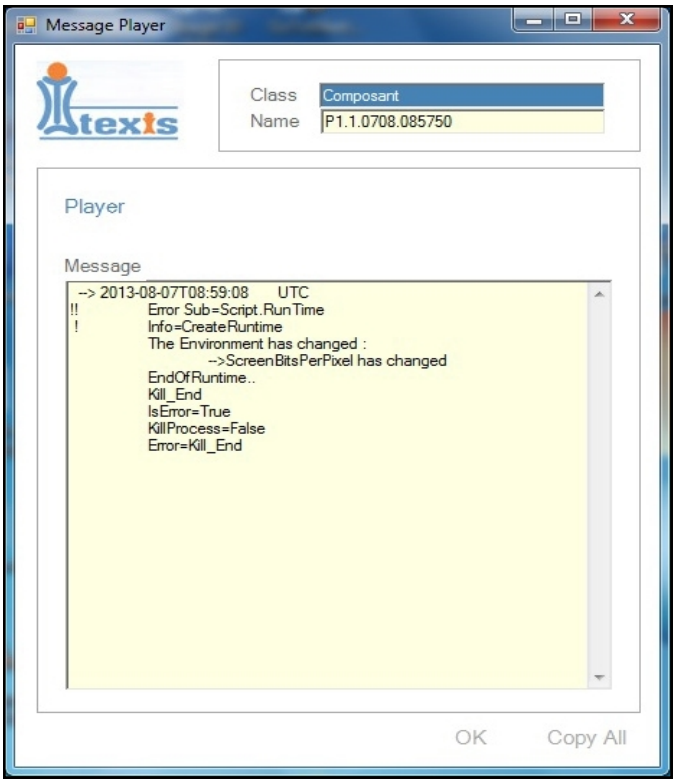

Figure 2.50: The error message that is displayed in the Player window

19. Clicking on the executable **Citrix-1** will load the script and lead you to the **Player** window. Click the **Start** button (see Figure 2.18) to start playing the recorded script.
20. During playback, if any of the transactions could not be completed due to reasons like say for e.g., the image captured is disoriented from its original location or removed, the script will abort and stop immediately. An error message as shown in 2.3.1 will then appear.
21. Once the script is successfully played back, a dialog box displaying the reference time or the total time taken to emulate the transactions will appear. Save the reference time for the script (see Figure 2.51) by clicking the **Yes** button and close the player window.

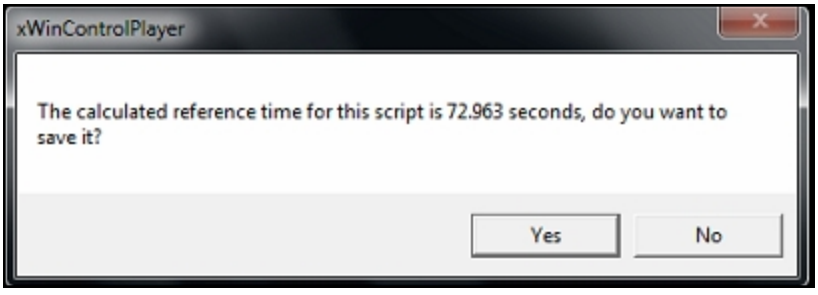

Figure 2.51: The calculated reference time for the script

22. By default, the logs created while emulating the transactions will not be stored in the database. To save the logs in a particular location of the database, say for e.g., **XWCF_Logs**, then set the

Savelogondatabase for the chosen script to **False** as depicted in Figure 2.21. The logs will now be successfully stored in the following location of the database: **C:\Documents and Settings\All Users\Application Data\Itexis\JobQ\M1\XWCF_Logs**

Note:

The logs will not be stored in your desired location immediately after the **Savelogondatabase** option is set to False. Instead, the script should be played back once successfully and from the next execution of the script, the logs will be stored in the chosen location which could then be used for client emulation.

23. The script that is generated using the AppsMon for Windows, will be integrated with the eG client emulation tool using the path of the Log and Script files. The eG client emulation tool checks the *Response time* and *Availability* of the component and reports to the client. The integration of the AppsMon for Windows to work with eG is explained in detail in the following sections.

2.3.2 Configuring the eG manager to Work with AppsMon for Windows

Since the component to be monitored in our second example is the citrix server, add this server as an Emulated Client. The procedure for adding an Emulated Client has already been dealt with in Example 1. Therefore, let us proceed to configure the Client Emulation test for the citrix server.

To achieve this, do the following:

1. After adding the Emulated Client, attempt to sign out of the eG administrative interface.
2. From the list of unconfigured tests that appears, click on **Client Emulation** test.
3. In the test configuration page that appears, specify the following (see Figure 2.52):

Figure 2.52: Configuring the Client Emulation test

- **TEST PERIOD** – How often should the test be executed
- **HOST** – The host on which the test will run. In our example, the test will attempt to extract measures from the host, 192.168.8.201.
- **PORT** – The port at which the specified host listens. In our example, this is Null.
- **TOOL SELECTION** – Select the tool using which you wish to perform client emulation from this list. By default this is *Citra*. To initiate AppsMon for Windows, select **Itexis** option from the drop down list.
- **SCRIPTFILES** – Specify the full path to the script file that is to be played back for emulating a request to, and extracting metrics from the monitored application. Multiple script files can be provided as a comma-separated list, but all script files should monitor the same application only. In our example, the path to the Citrix script has to be specified here.

  Note:

  If the script file resides on another host, then ensure that the location of the script file is mapped to any drive on the measurement host.

- **OUTPUTFILES** – Enter the full path to the output file that contains the metrics extracted by the specified script file. Here again, multiple output files can be provided as a comma-separated list, but only if multiple script files are also provided.

  Note:

  - If **None** is specified here, then the eG system will collect statistics from the default output files associated with each of the specified script files. The default output files will be present in the same location as the respective script files, and will have the same name as the script files. In our example, the value of the outputfiles parameter can remain as **None**.
  - While specifying multiple output files, ensure that they are provided in the same order as their corresponding script files in the scriptfiles text box.
  - If the scriptfiles parameter consists of multiple entries and the outputfiles parameter consists of only one, then eG will automatically associate the first script file entry in the scriptfiles box with the outputfiles entry. Measures pertaining to the other script files will therefore not be displayed in the eG monitor interface.

- **ISCITRIX** – If the specified script emulates a request using Itexis - AppsMon for Windows, then this parameter is not relevant.

4. Finally, click on the **Update** button in Figure 2.52, and then, signout of the administrative interface.

2.3.3 Starting the External Agent

Refer to the Section **2.2.4** for details on what is to be done before starting the external agent associated with an Emulated Client component.

2.3.4 Viewing the Measures

To view the measures reported by the **Client Emulation** test, do the following:

1. Login to the eG monitor interface.
2. From the **Components** menu, select the **Servers** option. Once the component list page appears, select the Emulated Client option from the **Type** list therein, so as to view the current state of all components of type Emulated Client. Then, click the **Submit** button (see Figure 2.53).

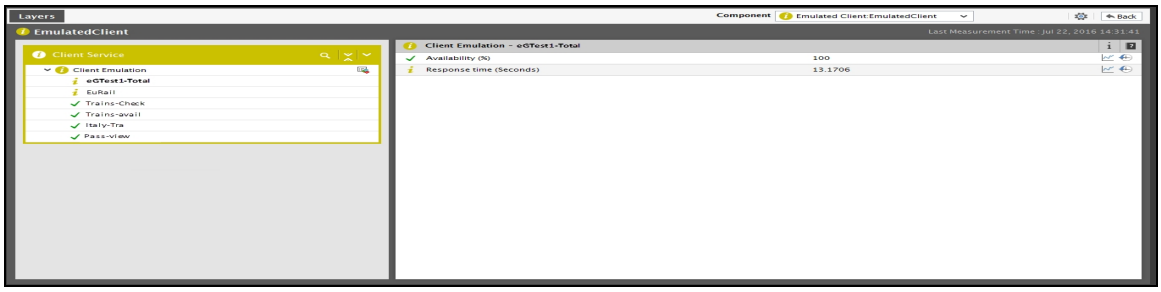

Figure 2.53: The Component List page

3. The Emulated Client component that we had configured previously will then be listed (see Figure 2.53). Click on the component to view its layer model, tests, and measurements (see Figure 2.54).

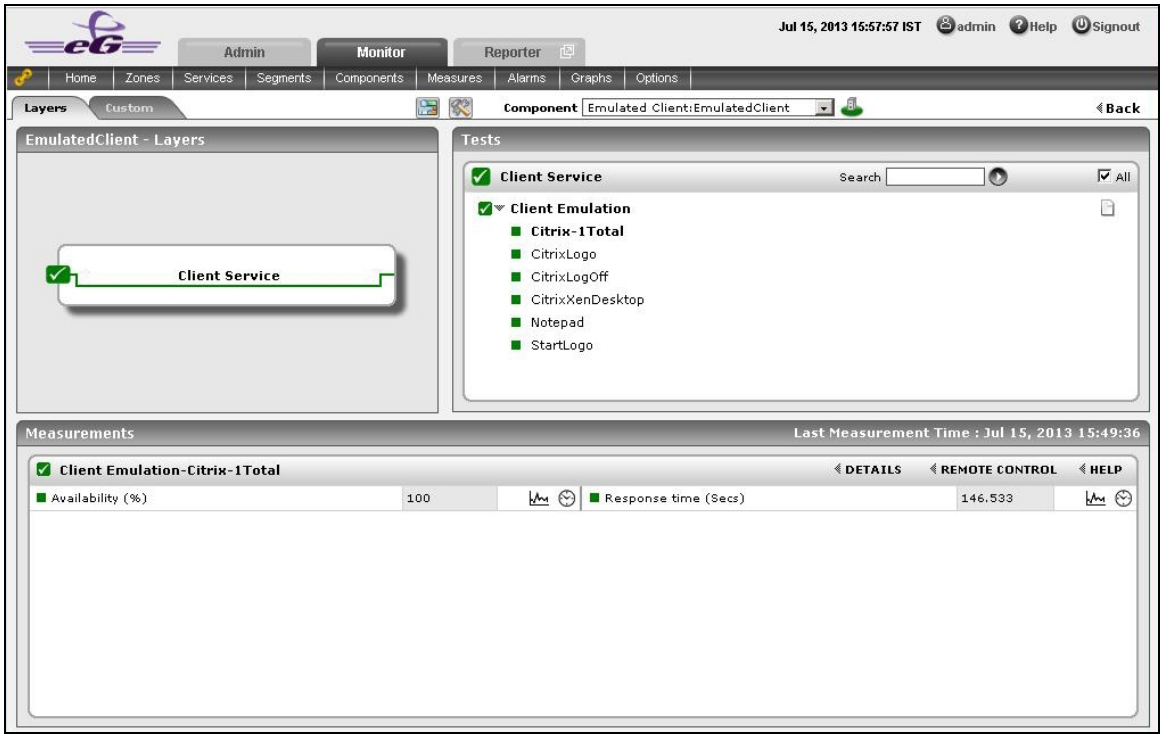

Figure 2.54: Viewing the layer model, tests, and measurements of the Emulated Client

4. Figure 2.54 displays the *Availability* and *Response time* of the **Citrix** script. This step calculates the time taken by a citrix user to login to the Citrix application, and reports this time as the *Response time*. Also, note that all the transactions that have been included in the **Citrix** script appear as descriptors of the test (see Figure 2.54).

5. Besides the transactions that we had explicitly recorded for this script, Figure 2.54 also displays an additional descriptor named, *Citrix-1 Total*. This descriptor - internally generated by the eG Enterprise Suite, reports the total time taken for script execution as the *Response time*. Figure 2.54 reveals the response time returned by the *Citrix-1 Total* descriptor. This is the total time taken to login to the Citrix XenDesktop, open the Notepad application published on it, add a line of text to the Notepad, and close the application without saving the changes.

3

Conclusion

The eG suite of products has been specially designed keeping in mind the unique requirements of IT infrastructure operators. For more information on the eG family of products, please visit our web site at www.eginnovations.com.

This document has described the purpose, benefits, and procedures involved in the usage of the eG Client Emulator.

For more details regarding the eG architecture and the details of the metrics collected by the eG agents, please refer to the following documents:

- A Virtual, Private Monitoring Solution for Multi-Domain IT Infrastructures
- The eG Installation Guide
- Administering the eG Enterprise Suite
- Monitoring the eG Enterprise Suite
- The eG Measurements Manual
- The eG Customization Manual

We recognize that the success of any product depends on its ability to address real customer needs, and are eager to hear from you regarding requests for enhancements to the products, suggestions for modifications to the product, and feedback regarding what works and what does not. Please provide all your inputs as well as any bug reports via email to support@eginnovations.com.