

ASIM-Functional User Guide

March 2015

This edition applies to ASIM-Function Version 1.2x

This document created or updated on March 10, 2015

Please send your comments and suggestions to:

GTECH Corporation  
55 Technology Way,  
West Greenwich, RI – 02817 USA.

This item is the property of GTECH Corporation, West Greenwich, Rhode Island, and contains confidential and trade secret information. It cannot be transferred from the custody or control of GTECH except as authorized in writing by an officer of GTECH. Neither this item nor the information it contains can be used, transferred, reproduced, published, or disclosed, in whole or in part, and directly or indirectly, except as expressly authorized by an officer of GTECH, pursuant to written agreement.

**Copyright© 2015 GTECH Corporation. All rights reserved**

Table of contents

1 - Introduction 1

1.1 What is the Automation Simulator? 1

1.2 ASIM Compared to V-Script(\*) 2

2 – Installation & Configuration 3

2.1 System Requirements 3

2.2 Installation & Starting ASIM 3

2.3 Configuration 4

3 – Using ASIM 8

3.1 ASIM Tab Descriptions 8

3.2 Transactions Log File 30

3.3 Other Log Files 31

4 – Known Issues 32

4.1 SDK Issue 32

Preface

## 1 - Introduction

## What is the Automation Simulator?

The Automation Simulator (ASIM) is a tool that enables users to send test transactions generated from **Game Matrix**(\*) into the Host (ESTE). ASIM communicates with ESTE using MXSRV/MX Client, ESConnect protocol, or HTML (through Internet Adapter for internet wager transactions). The transaction messages can be based on Terminal Bincoders, B2B Translets, or OpenAPI (interactive wagering protocol).

ASIM provides information about transactions being processed based on each script that is processed, overall statistics and a detailed description of each transaction.

This version of ASIM is generally known as “ASIM-Functional” since it is primary used for functional testing of Gaming products on ESTE system. Another application, called “ASIM-Performance”, is capable of performing load testing. There is future plan to merge both of these applications into a single application. Until then, this document only covers ASIM-Functional.

*(\*)* **Game Matrix***: GTECH gaming configuration and transaction generator tool.*



Figure 1.0 – ASIM-Functional application window

## ASIM Compared to V-Script(\*)

One of the main advantages of ASIM is that the effort required to configure the tool for a new site is minimal because ASIM uses Terminal Bincoders or B2B Translets for message encoding/decoding.

Like V-Script, ASIM automates draw processing but does it in a less cumbersome way than V-Script. ASIM sends the appropriate RPCs to the Host based on the information included in the XML file that is generated from the Game Matrix.

Users can run ASIM and if there are no unexpected situations, there is no need for manual attention. All information and statuses can be checked after processing is completed.

The other main advantage of ASIM is that Users do not need to use other tools like SIMNET, where VScript needs two instances of SIMNET (one for transactions and one for RPCs) during its operation.

ASIM is developed in Java and being constantly updated with new features and support protocols.

*(\*)* ***VScript****: GTECH older version of automated script controller. As of Jan 2012, VScript is no longer supported by GTECH Test Automation team.*

## 2 – Installation & Configuration

## 2.1 System Requirements

Minimum Run-time system requirements:

* IBM-PC compatible, 2 GB RAM, Microsoft Windows XP SP3 or Windows 7
* Java 1.6 or later Run-time environment

## 2.2 Installation & Starting ASIM

ASIM zipped package will be provided for site installation. This package needs to be unzipped to a directory on the User’s PC. For the purpose of this document, assume the package is unzipped to directory C:\ASIM. A **distribution** directory will automatically be created. In directory **C:\ASIM\distribution** users can find the following files and directory structures:

* **asim.jar** – ASIM application container
* **products.xml** – Site product mapping files
* **conf** – directory containing configuration files
* **log** – directory containing log files
* **jar** – directory containing external jar files (if applicable)
* **jar** – directory containing log files
* A few batch files to support different running options

The application now needs to be started for configuration.

**To start ASIM** the user can do one of the followings:

* Double-click the **asim.jar** icon (if jar file is associated with the Java run-time environment on the User’s PC )

**OR**

* From a Command window (cmd.exe, or “DOS” window), navigate to the directory containing asim.jar (C:\ASIM\distribution) , then type:

**java –jar asim.jar**

**OR**

* Double-click the RunASIM.bat (a Windows Command batch file) in the same distribution directory with asim.jar. The batch file contains the same java command (**java –jar asim.jar**) as of the other starting option, but also captures all console output messages to a timestamp log file (**AsimConsole\_YYYYMMDDhhmm.txt**). This file is useful for resolving ASIM terminal message encoder issues, or to assist support engineer in troubleshooting an ASIM installation.

* *Note:* 
  + *All cases above assume the JAVA run-time environment (1.6 or higher) has been properly installed on the user’s PC.*
  + *Every time ASIM starts, a set of time stamp log files will be generated in the same directory as asim.jar. These log files mirror/capture the contents of various ASIM panels during its session. They can be removed after the session is completed.*
  + *RunASIM.bat expects the Microsoft freeware “tail.exe” (also included with ASIM distribution) to be in the same directory with the batch file or in executable path.*

## 2.3 Configuration

ASIM uses two connections when communicating with the ESTE, one for Terminal transactions and one for RPCs. The MXSRV on ESTE needs to be configured to allow ASIM to communicate with the Host. This is done by editing the **mx\_server.fil** in the ESTE **$MXSRV/files** directory.

* **(ESTE) $MXSRV/files/mx\_server.fil**: Add two applications under two separated CLUSTERED\_APP group, one called **asim** and one called **asim-rpc.** See the left side of **Figure 1.1** for example. Enter the user PC name (“TESTPC01” in this example) under each application section. Note that the user PC name (e.g. TESTPC01) must be a valid entry on ESTE host file. Refer to ESTE manual for more detailed procedure of modifying ESTE Host table and MXSRV configuration file.
* From ASIM, go to the **Configuration Tab** (Figure 1.1) and enter the following information:
  + **Destination Port:** Enter the MXSRV port number found in the **mx\_server.fil**.
  + **Host address**: Enter the Host Name of the ESTE system the User is connecting to.
    - Note: If the User PC cannot resolve this Host name (via “ping” test), an entry for the Host name with its IP can be made to the User’s hosts file

(**C:\WINDOWS\system32\drivers\etc\hosts)**.

* + **Own Address**: Enter the IP Address of the User PC that ASIM is running from..
  + For the **Transformer** field, select either **bincoders** or **translets** from the dropdown list, depending on which transformer being used to test the site’s Host products. ASIM is packaged with both Baseline transformers. However, only the site’s specific transformer should be used for site testing.
  + Select the **Names configuration** **file** by clicking on the **Open** button at the Names Configuration file field and selecting the file (default name=“**nameBincoders.xml**”) from the location stored on the User’s PC This file should be in the **conf** subfolder of the ASIM distribution package.
  + **Ignore Transactions With Errors**: Put a checkmark on this box for ASIM to process the full script to completion without interruption. Any error notification will still be displayed and recorded while ASIM is running.

To stop ASIM at any error notification while the script is processing, leave the check box unchecked.

* + **Ignore Wager StartDraw Setting**: Put a checkmark on this box for ASIM to ignore the starting draw number in Wager transaction messages when sending them to ESTE. This option is useful for quick testing of ASIM on environment that is not starting from a startover. All wager transactions will be assumed the current default draw as starting draw.

Leave the check box unchecked when formally running ASIM with the multi-day test script, since test cases involved advanced draw will contain wager transaction with starting draw number explicitly specified.

* + **Data Input File**: This field specifies the XML script to be processed. In general, this XML file is generated by Game Matrix. However, it could be any XML file that contains test transactions created from any other sources, as long as it follows ASIM input file format convention.

Click the **Open** button to selecting the file from the location it was saved on the User’s PC.

* + **Internet Adapter Addr:** This field specifies the HTTP address of the Internet Adapter used in Internet wager test. ASIM will send the internet transactions (indicated by XML tags) within the test script through this address using HTML protocol.

Leave the field blank for online wager testing.

* + **Use ODBC** Checkbox: If the site is using OpenAccess (OA) gateway for RPC transactions, put a checkmark on this checkbox, then enter the required ODBC Client information into the popped up text fields (ODBC Database name, User and Password).

Keep the box unchecked for site using MXGATE for RPC transactions.

* + **Use ESC** Checkbox: Check this box if ASIM will be sending transactions through ESConnect. Fill in the ESConnect IP address or host name for the “ESConnect IP” field, and the terminal ROM ID for the “ROM ID” field. Note that there is only one ROM ID entry field, so all test terminals configured for using this feature should be having identical ROM ID.

Keep the box unchecked if ASIM will be sending transactions directly to ESTE using MXClient.

* Click on **Apply** button. ASIM configuration Setup for Site testing is now complete.
* If testing for a site that uses **Terminal Bincoders**, proceed to the checklist and instruction in **Section 2.3.1**. If testing for a site that uses **B2B Translets** for transformer, proceed to the instructions and check list in **Section** **2.3.2**.

|  |  |
| --- | --- |
| **mx\_server.fil (from ESTE)** | **ASIM configuration** |
| [TNI\_CONFIG]  LOCAL\_DOMAIN\_NAME = ibm223  LISTENING\_PORT = 53390  [CLUSTERED\_APP]  APP\_NAME = asim  REMOTE\_DOMAIN\_NAME = TESTPC01  AUTHENTICATION\_REQUIRED = FALSE  [CLUSTERED\_APP]  APP\_NAME=asim-rpc  REMOTE\_DOMAIN\_NAME = TESTPC01  AUTHENTICATION\_REQUIRED = FALSE |  |

**Figure 1.1 – MXSRV (ESTE) and ASIM Configuration**

### 2.3.1 Terminal Bincoders Required Files

If Terminal Bincoders (used for message encoding and decoding) are not already packed within ASIM main jar, there will be a few jar files included with each ASIM distribution:

* ***[sitename]\_*bcode.jar (e.g. "bg\_bcode.jar"):** This file contains the site's Terminal Bincoders. It should be included in the distribution package and located in the **jar** subdirectory (C:\ASIM\distribution\jar)
* ***[sitename]\_*baseline.jar** (e.g. "**bg\_baseline.jar**"): This file contains the site’s Baseline Bincoders that the previous file was extended or built from. This file should be included in the distribution package and located in the **jar** subdirectory (C:\ASIM\distribution\jar).
* **namesBincoders.xml**: This is a configuration file with the appropriate Bincoder field mappings for the specific site. This file should be included in the distribution package and located in the **conf** subdirectory (C:\ASIM\distribution\conf)

**NOTE:** *The number of jar files and their name convention may be different for sites that have multiple jar dependencies in bincoder. In any case, all necessary jars should be included with each ASIM distribution.*

### 2.3.2 B2B Translets Required Files

If B2B Translets are to be used for message encoding/decoding, the following additional information will be needed:

* A **translets.jar** file with the correct translets (should be provided by the site’s Host Engineer – instructions for the developer are included in the ASIM Developer Guide)
  + Put the **translets.jar** file in the jar directory C:\ASIM\distribution\jar
* A names configuration file with the appropriate field mapping (should also be provided from the site’s Host Engineer – instructions also included in the ASIM Developer Guide)
  + Put the names configuration file in the configuration directory C:\ASIM\distribution\conf

In the **Configuration tab** in ASIM:

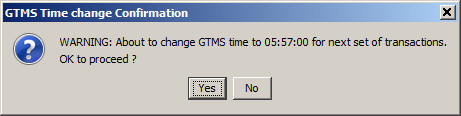
* Type in **translets** in the **transformer** field
* Click on the **Open** button for the **Names Configuration File** field and select the names configuration file provided by the Host Engineer from the location it was saved on the User’s PC

Configuration using B2B Translets is now complete; the User is now ready to begin using ASIM for site testing.

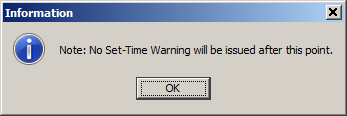
## 3 – Using ASIM

After starting and configuring ASIM (Section 2.2 & 2.3), processing can be started by clicking the button **“Load & Run”** from the top right corner of the **Script** tab.

A dialog will be displayed asking for re-confirmation of ASIM first command of setting GTMS time of the test environment to the first script segment’s time value (usually the system’s day-start time). This dialog provides an option to back out in case the Load&Run button was accidentally pressed.



Processing can only be started with a “**Yes**” confirmation. At which point another dialog is displayed to remind user that ASIM will no longer display any warning for subsequent Set Time commands:



Click **OK** to close the dialog. If properly configured, ASIM will load the XML test script and start sending transactions to the test ESTE. User can monitor the process from various ASIM tabs, transaction and error log files, and from the console window where ASIM was started (via command line method described in Section 2.2)

The following sections describe all the tabs available from the ASIM application and their functions in details.

## 3.1 ASIM Tab Descriptions



### Script Tab

The **Script** tab is the main tab of ASIM operation. From here, a test script’s transaction details can be examined before, during, and after execution.

Transactions are loaded into the Script tab from one of two methods:

* + If the User wants to load the script but not execute it immediately, select the **Load XML** button at the bottom of the Main. All the transactions will be loaded onto tables from upper and lower panel of this tab. This is the preferred method if for some reason, the transactions need to be examined before being sent to the test host, or if the test script needs to be altered in some way (as discussed in the next section)
  + If the User wants to execute the script immediately as soon as it is loaded, press the **Load & Run** button from the top right corner (Figure 1.2). ASIM will attempt to execute the script from the beginning to the end, **as soon as it is loaded into the program**. Note that the **Load & Run** button will be replaced by the **Run** button after a script is loaded (as shown in Figure 1.3)

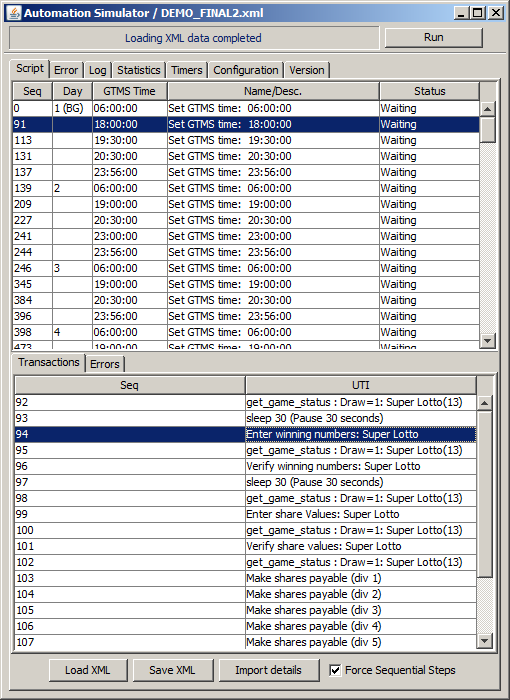
Once data is loaded, the **Script Tab** contains the list of available script elements or “time slices” arranged as sequential table rows. Each row is labeled with test day, time, “set\_gtms\_time”, and the current processing status. When the User selects a row, all the transactions applicable to that row (or time slice) will be displayed at the bottom half of the window (labeled as **Transaction** tab). The next section (ASIM Transaction Tab) will provide detailed descriptions of the data columns and the method of viewing them.

|  |  |
| --- | --- |
| **Load &Run/Run/Stop** – This button is used to stop and start the script  **Load XML** – Used to load selected XML script into ASIM  **Save XML** – Used to save the changes made during execution and exported as XML.  **Import details** – Used to import details from transactions file and continue testing previously used script.  **Force Sequential Steps** – This checkbox makes the script follow the sequence order. |  |

Figure 1.2 ASIM main window script tab before loading input XML file

Additional information on the buttons accessible from the Script tab:

* + When the script is executing, the **Run** button indicates **Stop** to allow the User to use the same button to stop the script if they choose to do so.
  + **Import Details** button at the bottom of the Main Window
    - This would only be necessary if the User had not completed working through all transactions during the previous ASIM session
    - After loading the XML for a script that was run previously, this button is used to import serial number and status information from the log file to use for ticket validations or cancellations.

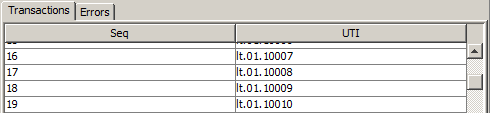


XML script loaded into ASIM

#### Script Sub-Tabs Transactions Tab

The **Transactions Tab** becomes visible when the Script tab is selected. It occupies the lower half portion of ASIM application window. The **Transactions Tab’s** table data is updated whenever a row on the **Script Tab** table is selected (each row on the Script tab table represents a single script element or a “time slice” within the full script). The **Transactions Tab’s** table data represents the list of transactions associated with the selected time slice.

If transactions for a selected script are not yet processed, there will only be two columns with data:



* **Seq (Sequence Number)** – The sequential order of the transaction relative to entire XML test script. This number is assigned when the transaction table is loaded.
* **UTI** – Unique ID that identifies the simulated terminal transaction’s properties. This number was assigned by the test script generator (Game Matrix). It has the following format:

**xx.yy.zzzzz (Example: lt.01.10009)**

with

**xx**: 2-character product identification label as configured in Game Matrix.

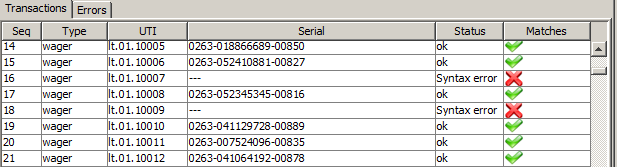
**yy**: Transaction Type code: 01=wager, 02=Cancellation, 03=Validation, 04=Exchange(Other codes possible, depending on sites)

**zzzzz**: 5-digit date indicator plus the sequential order of the transaction within that date. This number begins at 10,000 for day 1 and increases by 10,000 for each day. For example, 10030 would belong to the 30th transaction of the 1st test date, and 30256 would be from a 256th transaction of the 3rd test date.

From the example above**, lt.01.10009** is the 9th transaction from the 1st test date, with Wager (“01”) as the transaction type, and for the product Lotto (“lt”).

For RPC transactions, the UTI column will show the RPC name instead.

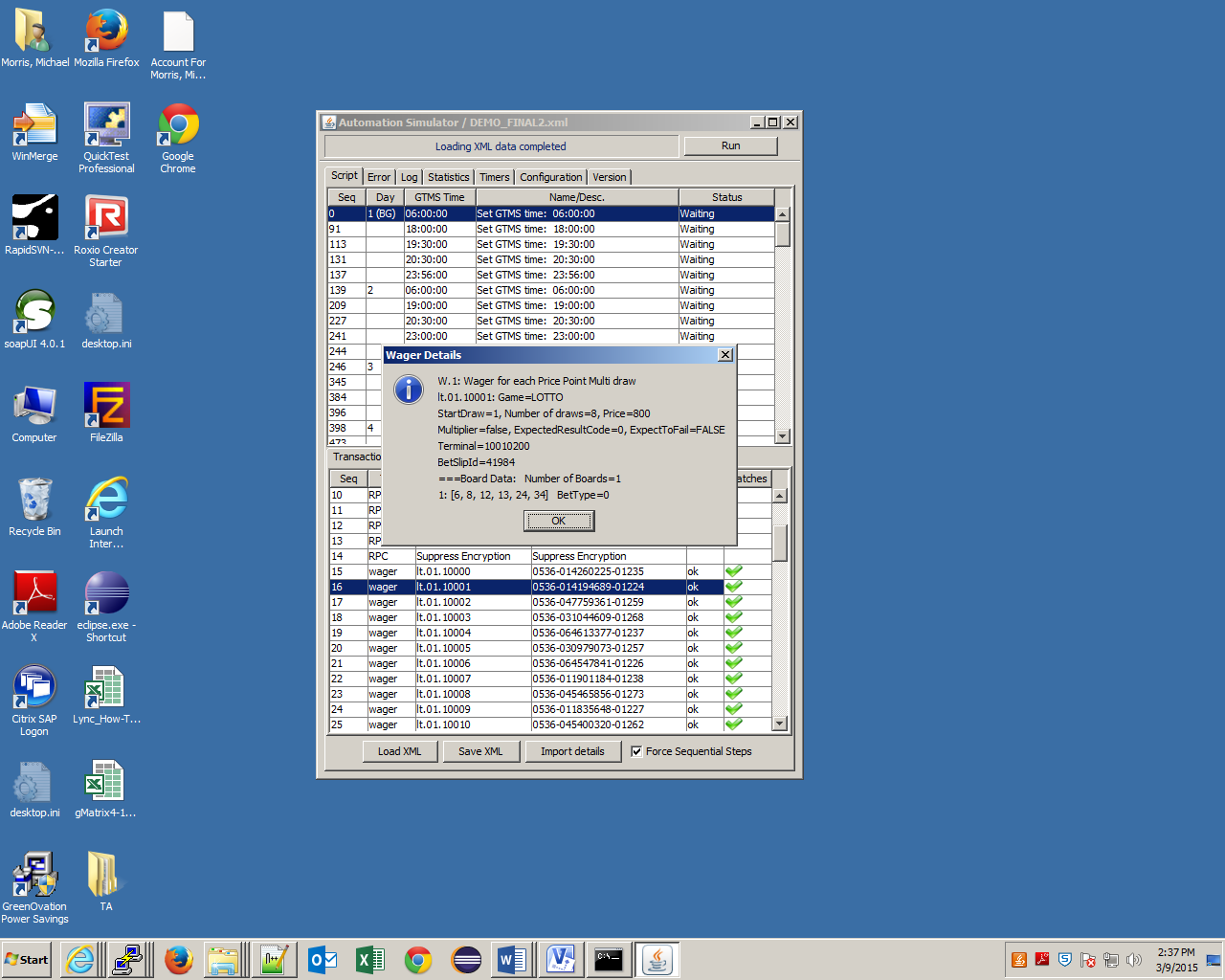
* If transactions for a selected script are already processed, all of the following columns will have data for that transaction:



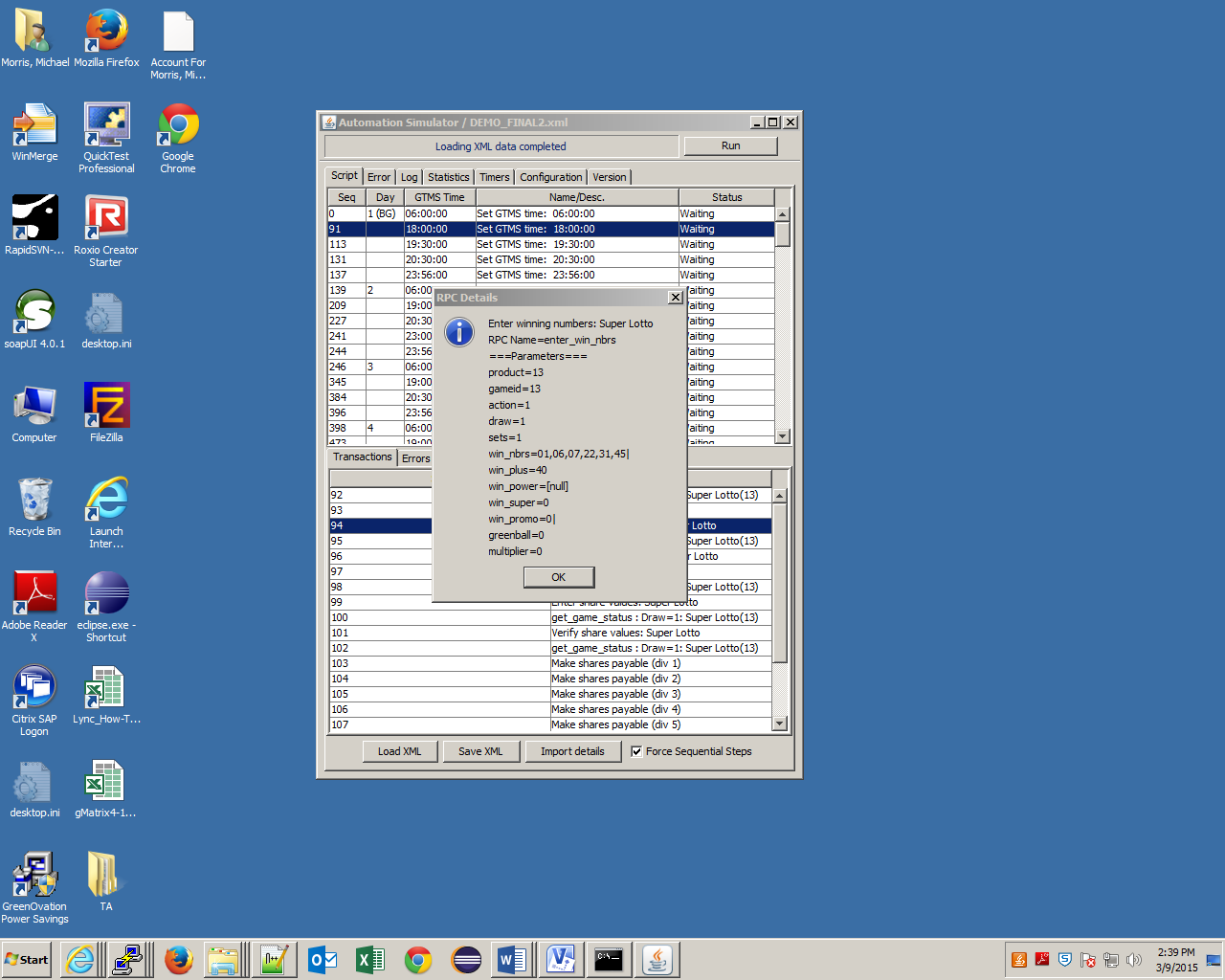
* **Seq. -** Sequence number as assigned when transaction table is loaded
* **Type** – Transaction type, such as “RPC”, “wager”, “cancel”, “validation”, etc.
* **UTI** - Unique transaction ID as assigned by test script (Game Matrix)
* **Serial** – serial number from the Host transaction
* **Status** – status returned from the Host
  + - If the transaction processed successfully, the status will indicate **ok**
    - If the transaction was rejected, the status will either be the Host error return code or a short error text message.
* **Matches** – this is validation of the match between expected and actual results:
  + - If expected result matches actual result for the transaction, a green checkmark  will be displayed in this column.
    - If expected result does not match actual result for the transaction, a red cross will be displayed in this column.

##### Displaying individual transaction detail

Within the script transaction window, user can double-click on a transaction (either terminal or RPC) to show its details in a pop up dialog as in the figures below.

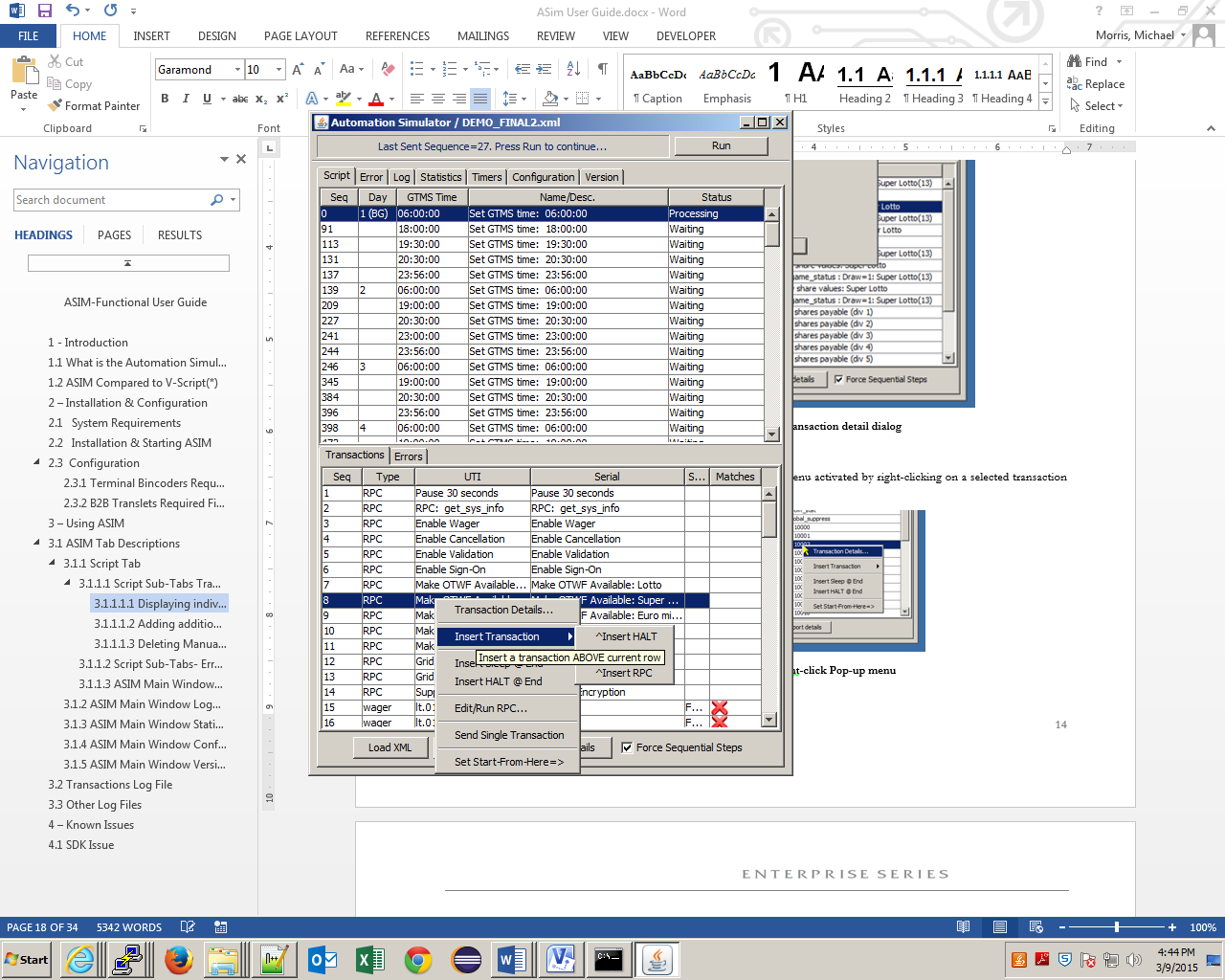


**Figure 1.3.1 Sample wager transaction detail dialog**



**Figure 1.3.2 Sample RPC transaction detail dialog**

Transaction details can also be displayed from the pop up menu activated by right-clicking on a selected transaction then select “Transaction Details…” menu item.



**Figure 1.3.2 Right-click Pop-up menu**

##### 3.1.1.1.2 Adding additional script Controller Steps

The **Script Tab** also provides user with the ability to run some of the transactions in a particular script segment before pausing it, as well as the ability to skip transactions and start execution from a specific point instead of the default location.

These options are useful if there is a need to validate the result of a single or group of transactions before the entire script is completed, or if the user decides not to include certain transactions at the beginning of the script during the running session.

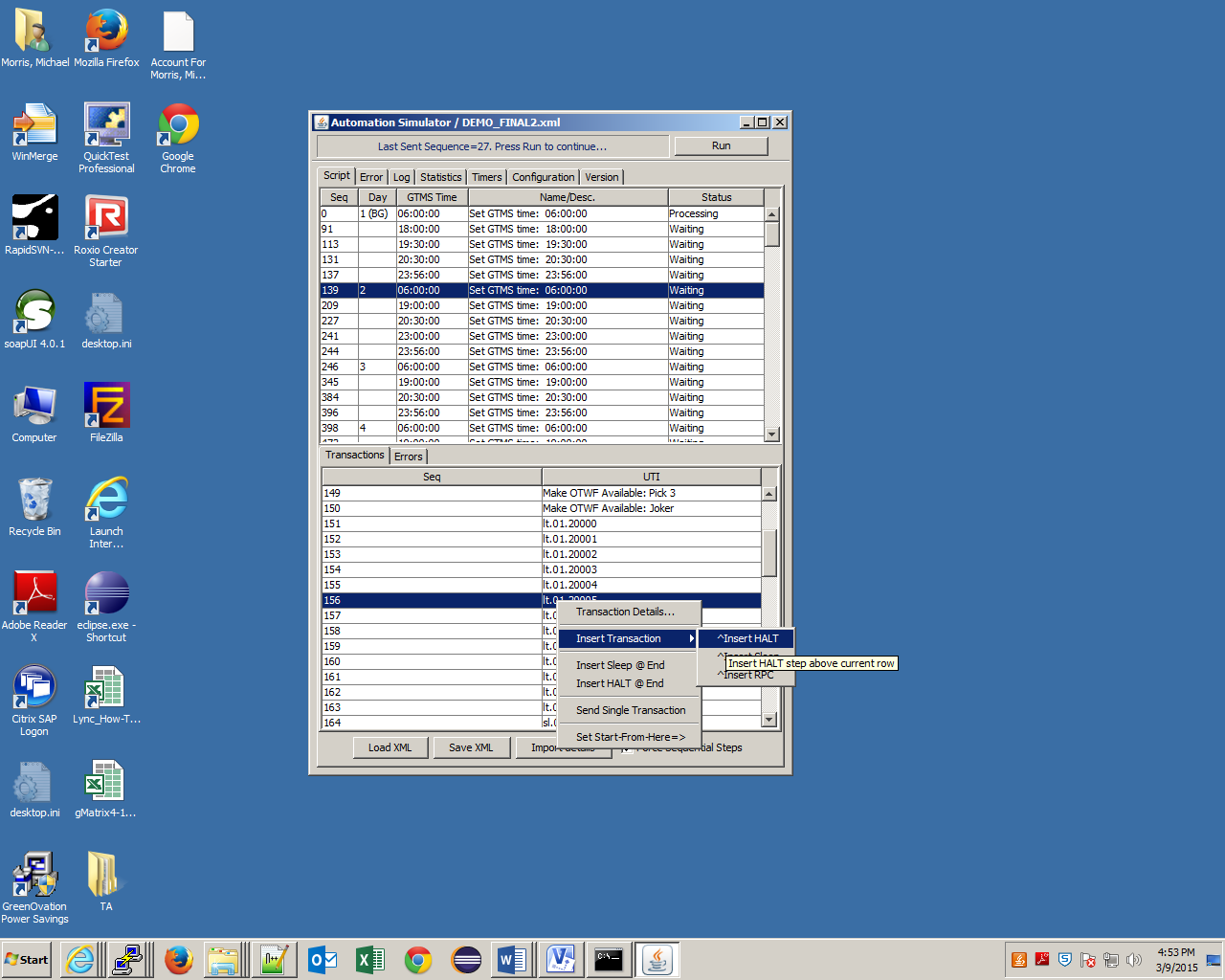
Regardless of the reason for the manual intervention, the process for script execution is different in this case:

* + If the User intends to intervene with the script at some points during its execution, the **Load and Run** button (see Figure 1.2) should not be used for script execution. Instead, use the **Load XML** button at the bottom of the Main Window to load the script without running it immediately. When the script transactions are loaded , the **Load and Run** button is replaced by a **Run** button (as in Figure 1.3)
  + Before executing the script with the **Run** button, the User has the option of manually inserting one or more script controller steps. Available options are:
    1. **HALT** transaction step: ASIM will stop processing at this step until the Run button is pressed.
    2. **SLEEP** transaction step. ASIM will delay at this step for an amount of seconds (adjustable value, default is 60 seconds), then continues processing when the time period expired.
    3. Product **RPC** command. Any (ESTE) product RPC identified by name and parameters can be insert at the specified location.
  + User also has the option of starting the script at certain location instead of the default location (which is the beginning of the script or at the last stopping point) by creating a START-FROM-HERE point.

The options are selectable from the context menu activated by right-clicking on a transaction in the transactions window (lower panel).

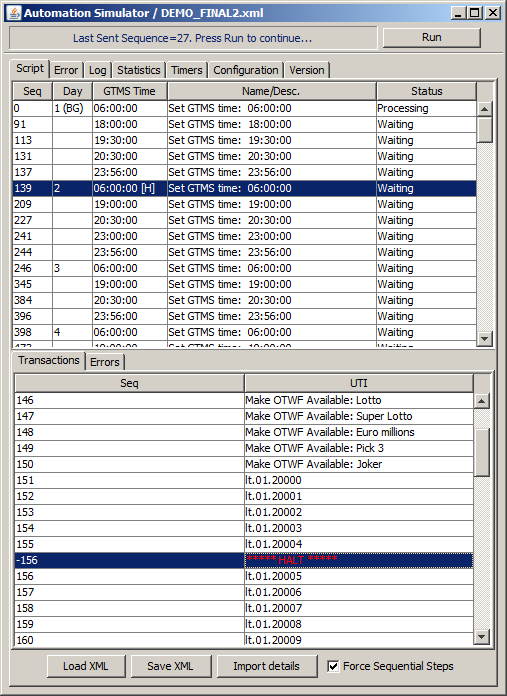
***NOTE***: *The controller steps CANNOT also be inserted into a script while ASIM has been running. However, a controller step added at this stage will function properly only on the script section (or time slice) that has not been executing at all. On partially-executed time slice, depending on its memory state, ASIM processor may skip over the newly added steps.*

* + **To add HALT transaction step**:
    1. Right-click on the transaction at which the script should stop execution; a drop-down menu will be displayed as shown in the figure below.

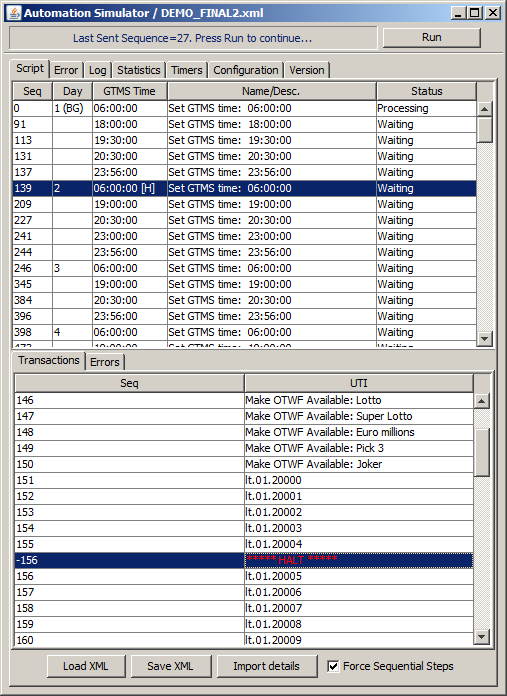


**Figure 1.3.3 Insert HALT menu selection**

* + 1. Select “^**Insert HALT**” from the sub menu that pops up when the mouse pointer is over “Insert Transaction” item.
    2. When  **^Insert HALT** is selected, there is a duplicate of the transaction created in the line above the original transaction; it will have the same Sequence Number as the original transaction but the number is preceded by a minus sign to indicate it was manually added. The UTI field for this manually added line will indicate “\*\*\* **HALT** \*\*\*”.



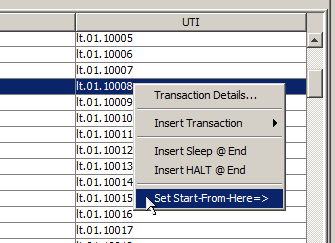
* + 1. **Insert HALT @ End**: This option from the main popup menu is for adding a HALT point at the end of the current transaction table, since **^Insert HALT** always inserts the HALT point **above** the selected transaction (which makes it impossible to use this option to add a HALT point at the very end of the transaction list)
    2. When the script is executing it will stop when it gets to the transaction where the HALT point was added



* + 1. Any time slice script that contains at least a HALT point will have an **“[H]”** indicator on its upper panel as shown in figure below. This signifies ASIM will eventually stop processing somewhere within the time slice, and will require manual intervention to continue.
    2. User can add as many HALT points as needed. However, either one HALT point or SLEEP point (next section) can be added as the last transaction on the current transaction table.
    3. While the script is running, the **Run** button will indicate **Stop** to allow the User to stop script execution if they choose to do so before getting to the manually added Halt point
    4. Once the script gets to the transaction at which the Halt point was added and it stops, the **Stop** button goes back to indicating **Run** and the User can perform whatever checks are needed.
  + **To add SLEEP transaction step**:
    1. Similar to HALT, a SLEEP point is added from the pop-up menu activated by right-clicking on a transaction that the SLEEP point is to be added directly above it.
    2. The default SLEEP period is 60 seconds. Double-click on the Sleep transaction to change this value.
    3. **Insert Sleep @ End**: This option from the main popup menu is for adding a SLEEP point at the end of the current transaction table, since **^Insert Sleep** always inserts the SLEEP point **above** the selected transaction (which makes it impossible to use this option to add a SLEEP point at the very end of the transaction list)
    4. User can add as many SLEEP points as needed. However, either one SLEEP point or HALT point (previous section) can be added as the last transaction on the current transaction table.
  + **To add RPC transaction step** (Advanced Feature)
    1. RPC transaction is added from the same pop-up menu for adding HALT and SLEEP transaction
    2. User is asked to enter the exact name of the RPC and its parameters values in additional dialog boxes.

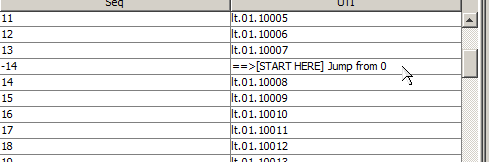
***NOTE****: This feature is more for script developer during unit testing. End user should not have to use this feature since the final test script should already contain all the necessary RPCs when loaded in ASIM.*

* + **To set a Start-From-Here point:**
    1. Right click on the transaction at which the script should start execution; the pop up menu for manually adding script controller steps will be displayed.



**Figure 1.3.4 Set Start-From-Here menu selection**

* + - 1. Select the item “Set Start-From-Here=>”
    1. When **Start-From-Here** is selected, there is a duplicate of the transaction created in the line above the original transaction; it will have the same Sequence Number as the original transaction but the number is preceded by a minus sign to indicate it was manually added

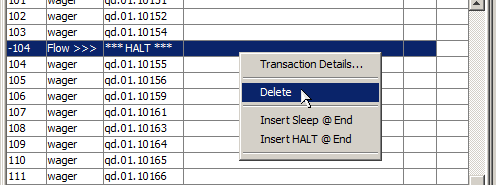


**Figure 1.3.5 Start-From-Here step indicator**

* + 1. The User can then select the **Run** button; the script will begin execution at the transaction where **Start from here** was selected and will skip all transactions prior to that one
    2. While the script is running, the **Run** button will indicate **Stop** to allow the User to stop script execution if they choose to do so before getting through the entire run (or to a halt point if one was manually added)
    3. Only one Start-From-Here point can be added. ASIM will attempt to remove a previously added Start-From-Here point when a new one is added

##### Deleting Manually Added Transaction Step

* + A manually-added transaction step can be removed from the current script. To delete:
    1. Right click on the manually added line for the transaction (the transaction with the Sequence Number preceded by the minus sign)
    2. The drop-down menu will now include a **Delete** option – this will only be available if the transaction selected is the manually added one. Note that all insert options won’t be available from the pop up menu when a manually added transaction is selected.

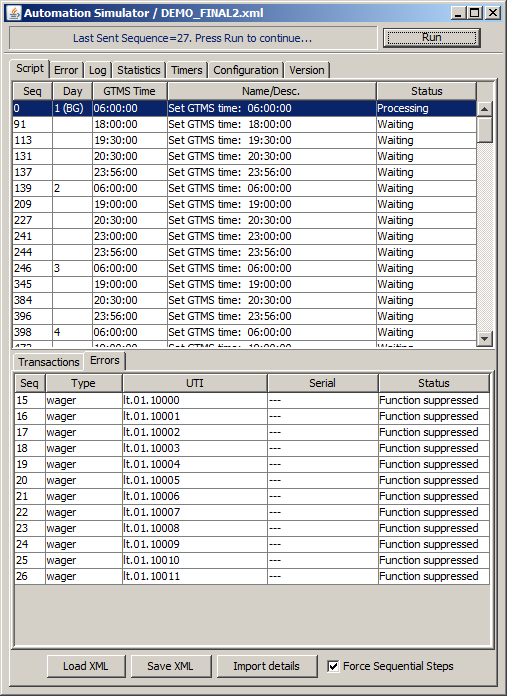


**Figure 1.3.6 Delete controller step menu item**

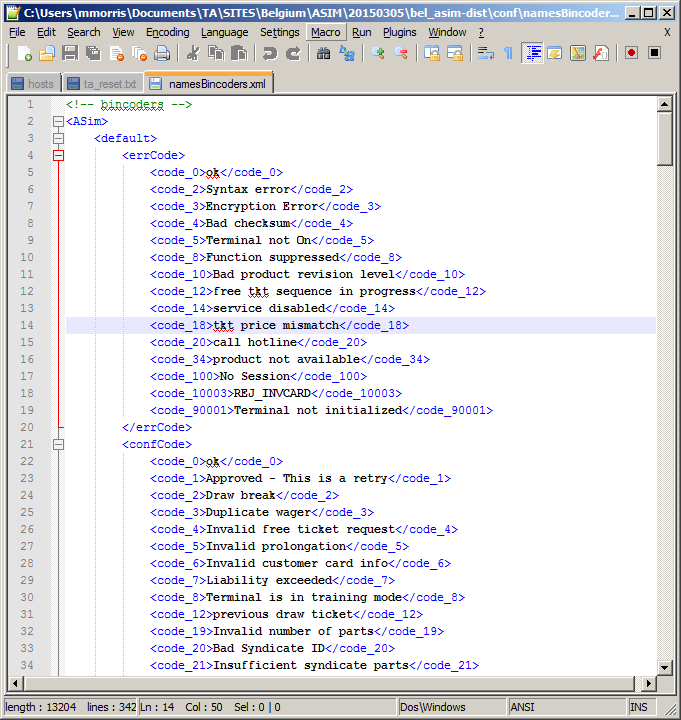
* + 1. Select **Delete** – the manually-added transaction will be removed

#### Script Sub-Tabs- Error Tab

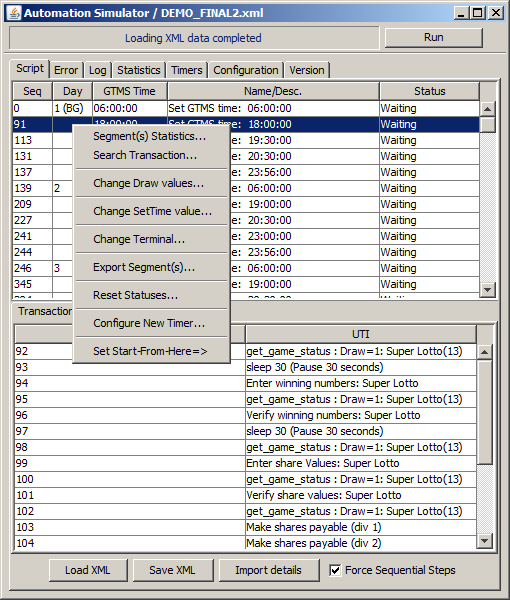
The Script Error tab displays the transactions which have received unexpected results. The **‘Status’** column displays a description of the problem.



The description is the result of the Host Return Code (a code number returned from the Host to signify the results of transaction results being sent) being referenced within the **‘NamesBincoders.xml’** file with the error codes listed.



#### Script Management



There are also some features to assist in managing the test script. As shown above these options affect the time segment which has been selected. The following options available shown above are:

**Segment Statistics** – Displays the makeup of the transactions to be executed in the segment. This can also be displayed via double clicking the time segment. The

**Search for Transaction** – Searches the script for various transactions based on input.

**Change Draw Values** – Update the RPC’s in the time segment to handle a new draw number or enter ‘?’ to have ASIM query the system and enter the current draw.

**Change SetTime Value** – This option enables the tester to make changes to the scheduled time and edit the description for the ‘**Name/Desc**.’ Column.

**Change Terminal** – Terminals within the segment can be changed via this option. ALL Terminals within this segment will be updated to the entered terminal value.

**Export Segment(s)** – The segment will be exported in XML format. The Save window will enable you to name the segment(s) selected for export.

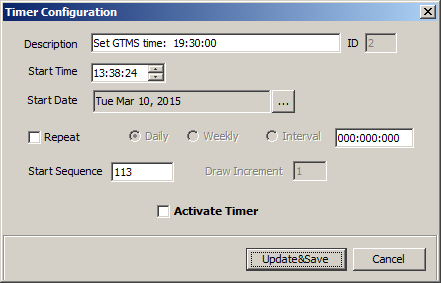
**Reset Statuses... -** The current segment will be reset to the state it was before being run. This will enable the tester to run a particular segment as frequently as they wish.

**Configure New Timer** – Control the test script through setting the launch time to the local machine time begin executing the script from the set starting point.

**Set Start-From-Here=>** - The option to set the starting point at one of the selected time segments. This is similar to the set start from here for the ‘Transactions Tab’. The feature in this section will change the time first then begin executing the selected transactions in the segment.

##### Configuring New Timer

The option to configure a new timer is quite a useful feature to run scripts at their expected times.



Check to Activate Timer

Set the expected time, according to local system time

Set the Sequence number that you wish to start from

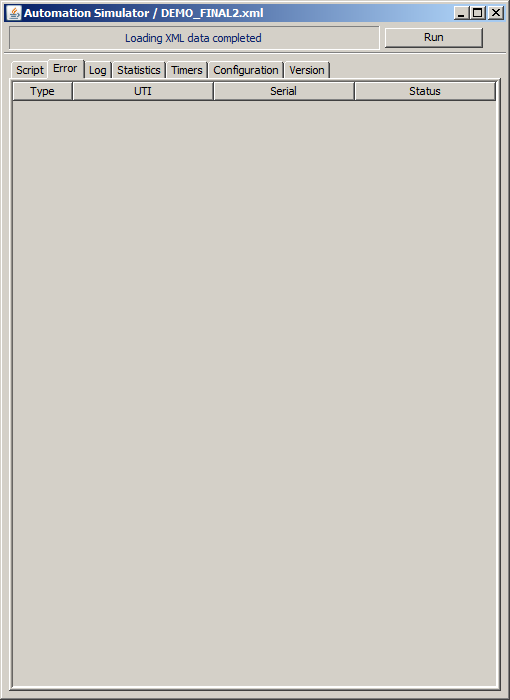
Set Intervals for restarting the Timer

Description for Timer

This window is generated from selecting the option to create a new timer. The options to configure the timer are as listed.

### ASIM Main Window Error Tab

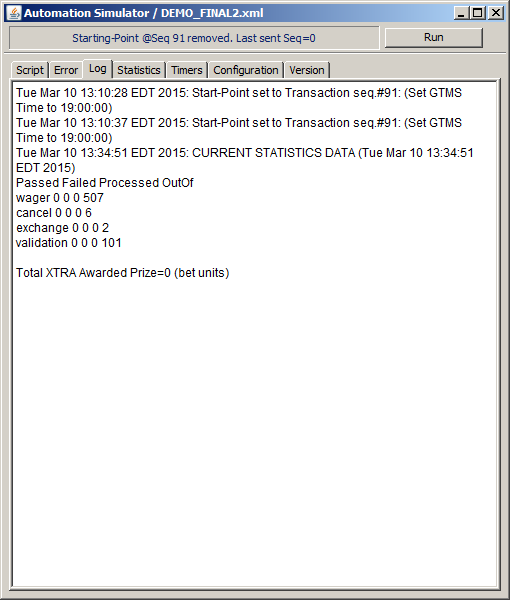
The **Error Tab** contains a list of all failed transactions from all processed scripts



**Figure 1.5 – Error Tab**

### ASIM Main Window Log Tab

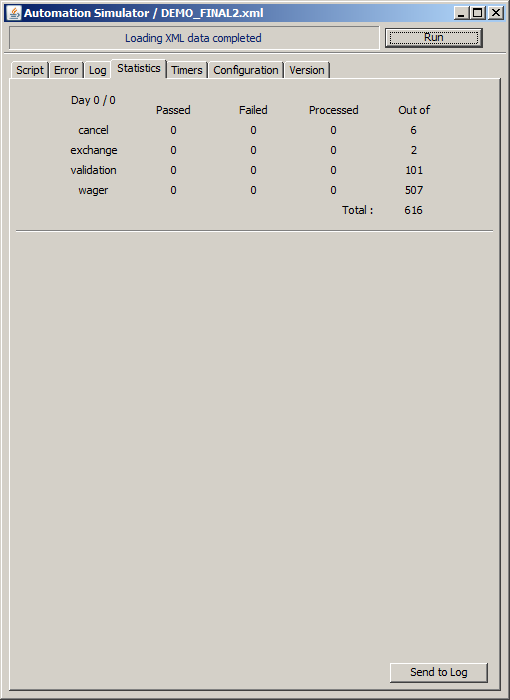
The **Log Tab** contains useful information and error descriptions



**Figure 1.6 – Log Tab**

### ASIM Main Window Statistics Tab

The **Statistics Tab** contains statistics for the processed transactions. The total number of transactions of each type from the loaded test script are shown under the “**Out of**” (last) column. As ASIM processing the transactions, data under the other 3 columns (“**Passed**”, “**Failed**”, and “**Processed**”) will be updated dynamically.

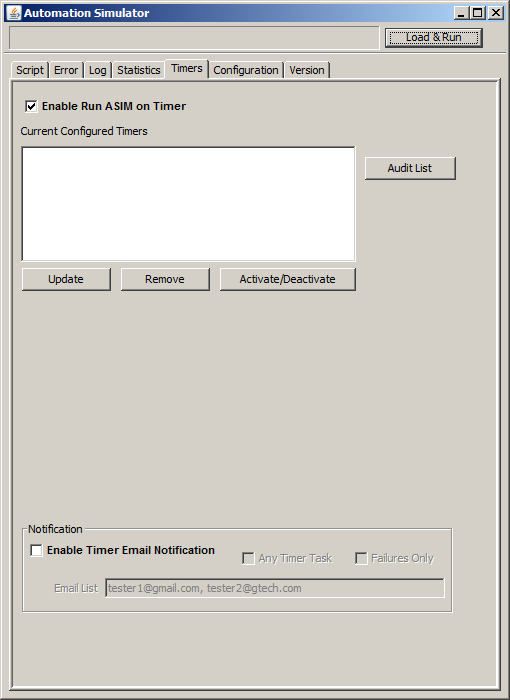


**Figure 1.7 – Statistics Tab**

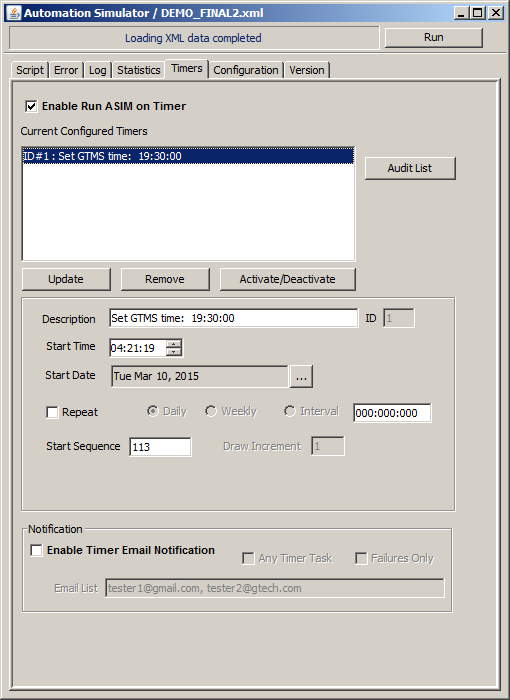
The ‘**Send to Log**’ Button will export the statistics information to the log file. This enables the transactions executed during the course of the testing to record the current progress.

### Timer Configuration Tab

The Timer configuration Tab is made to manage the Timers that have been scheduled in the script. This tab also helps to manage the notifications of any alerts from the tool. The various setting available and functions are explained below:



**Enable Run ASIM on Timer** – This allows the tester to use the Timer Tab’s functions and run scripts as well as receive emails if configured and allowed by network.



**Audit List** – This examines the selected timer and checks for possible errors.

**Update** – This button will save changes to the Timer immediately for execution.

**Remove** – Removes the selected Timer and erases it.

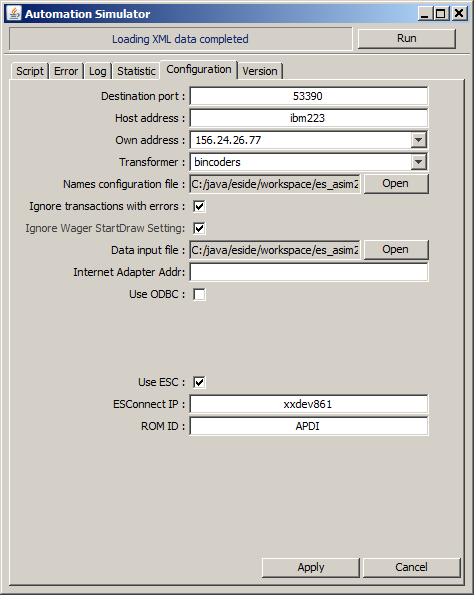
**Activate/Deactivate** – enables or disables the configured Timer to either execute the script or cancel.

**Enable Timer Notification Email** – This checkbox will allow the tester to email notifications for the current script executions

### ASIM Main Window Configuration Tab

The **Configuration Tab** is used when the User is setting up ASIM for use in testing. The procedures for configuring ASIM were previously outlined in section 2.2 of this document. The following fields are available in the **Configuration Tab** (see screen print on the following page):

1. **Destination Port** – Used to enter the MS Server Port Number of the Host system that ASIM will connect to
2. **Host address –**  Used to enter the Host name of the Host system that ASIM will connect to
3. **Own Address** - The User own PC IP Address that ASIM is running from.
4. **Transformer** – Type **translets** or **bincoders**. This specifies the type of transformer being used by ASIM for sending and receiving transaction to/from ESTE
5. **Names Configuration File** – This field is used for selection of the names configuration file provided by the site’s Host or Terminal Engineer (based on whether or not B2B Translets or Terminal Bincoders are being used). Click on **Open** and then select the file from the location it was saved on the User’s PC.
6. **Ignore Transactions With Errors** – Used to indicate whether or not ASIM will stop execution and prompt the User about failed transactions. If the User wants the run to be prompted about failed transactions as the run is progressing, click the checkbox to select this feature
7. **Ignore Wager StartDraw Setting** – This option is useful for quick testing of ASIM on ESTE environment that is not starting from a project startover. All wager transactions will be assumed the current default draw as starting draw. Leave the check box unchecked when formally running the multi-day test script, since test cases involved advanced draw will contain wagers with starting and ending draw number explicitly specified.
8. **Data Input File** – This field is used for selection of the XML file that was exported from the Game Matrix; click on **Open** and then select the file from the location it was saved on the User’s PC
9. **Use ODBC**: Used to indicate whether OpenAccess (OA) gateway is used on the Test Host System for RPC transactions.
10. **Use ESC**: Used to indicate ASIM will be sending transactions through the ESConnect with the assigned IP address or host name. All terminals configured for ESC testing should have the same ROM ID assigned on the field.



**Figure 1.8 – Configuration Tab**

**Create update the**

### ASIM Main Window Version Tab

The **Version Tab** contains product information about ASIM version number, site, and built date.



**Figure 1.9 – Version Tab**

## 3.2 Transactions Log File

All transactions that are executed are saved into a **transactions\_YYYYMMDDhhmm.log** file. This file is automatically created in the same location where the **asim.jar** file is saved (refer to the configuration section at the beginning of this document for the location of the **asim.jar** file). Figure 1.10 on the following page is an example of the transactions.log file.

Transaction status

31 wager P4.01.10051 991-34915845-01400 ok PASSED

35 cancel P4.02.10050 991-10853889-01400 ok PASSED

36 cancel P4.02.10051 991-01416709-01400 ok PASSED

39 --Draw-- 12:20:00

40 --Draw-- 12:20:00

43 wager KN.01.10057 991-56420869-02200 ok PASSED

42 wager KN.01.10056 991-65858049-02200 ok PASSED

41 validation NB.03.10017 991-58583812-00900 ok PASSED

41 Exchange NB.04.10017 991-30192384-00900 NB.03.10017 PASSED

44 --Draw-- 18:00:00

47 wager NB.01.10014 991-58518276-00900 ok PASSED

54 wager NB.01.10020 991-09281280-00900 ok PASSED

45 wager NB.01.10012 991-42845952-00900 ok PASSED

Response code\*

Sequence number

Transaction UTI

Transaction serial

Transaction type

**Figure 1.10 – Transactions Log File**

**\* For Exchange transactions, the response code is the original validation UTI.**

## 3.3 Other Log Files

In addition to the transaction log file (Section 3.2), ASIM also generates as set of time-stamped log files with .**txt** extension for every running session:

* **ErrorPanel**\_YYYYMMDDhhmm.txt
* **LogPanel**\_YYYYMMDDhhmm.txt
* **AsimConsole**\_YYYYMMDDhhmm.txt: only generated when RunASIM.bat was used to launch ASIM (Section 2.2).

The **ErrorPanel** and **LogPanel** file contain the exact text outputs of ASIM tab panels of the similar name. The **AsimConsole** file captures everything from the command (DOS) window that ASIM is running. These files are useful as diagnostic tool during development, or to assist supported engineer in troubleshooting an ASIM installation. If not needed, they can be removed from ASIM log directory.

## 4 – Known Issues

## 4.1 SDK Issue

There is an RPC issue that needs to be fixed via the ESSDK that will sometimes result in a User having to restart ASIM after processing is initially started. This occurs only at the start of processing and mostly after a project startover is done on the Host.

There will be an error generated when this occurs (see Figure 1.11 below). If this error occurs, the User will simply need to restart ASIM and will then be able to proceed with script execution. The fix in the SDK will be addressed at a later time.

86951 [te - RPC Connection[156.24.63.82] # 1: ] INFO tni.TniContentHandler - te - RPC Connection[156.24.63.82] # 1: received PDU\_RPC\_SERVER\_DATA: CC 23 0 4E B 6 1 1 11 23 66 62 30 66 32 39 62 35 31 37 32 2E 33 31 2E 31 30 30 2E 34 31 33 34 32 61 36 37 63 38 35 38 65 37 35 62 A 1 1 B 1 1B 10 0 1B 1 1 D 75 70 64 5F 6F 74 77 66 5F 73 74 61 74 0 0 0 0 1 0 0 0

86951 [te - RPC Connection[156.24.63.82] # 1: ] INFO tni.TniContentHandler - te - RPC Connection[156.24.63.82] # 1: Received an RPC application message for rpcRequestTag[fb0f29b5172.31.100.41342a67c858e75b]

java.lang.reflect.InvocationTargetException

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

at java.lang.reflect.Method.invoke(Method.java:597)

at com.gtech.translets.TransletParser.transformRequest(TransletParser.java:75)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

at java.lang.reflect.Method.invoke(Method.java:597)

at com.gtech.connector.transformer.Transformer.transformRequest(Transformer.java:145)

at com.gtech.connector.HostManagedConnection.transformRequest(HostManagedConnection.java:823)

at com.gtech.connector.HostManagedConnection.sendRecord(HostManagedConnection.java:655)

at com.gtech.connector.HostConnection.sendRecord(HostConnection.java:259)

at com.gtech.connector.HostInteraction.execute(HostInteraction.java:183)

at main.communication.TransletConnector.execute(TransletConnector.java:29)

at main.communication.TransletTerminal.processTransaction(TransletTerminal.java:62)

at main.communication.TransletTerminal.run(TransletTerminal.java:160)

at java.lang.Thread.run(Thread.java:619)

Caused by: java.lang.NullPointerException

at com.gtech.translets.runtime.Convert.getData(Convert.java:611)

at com.gtech.translets.runtime.Convert.getNumber(Convert.java:652)

at com.gtech.translets.runtime.Convert.getAppDetails(Convert.java:689)

at te.gtech.translets.request.Request.encode(Request.java:98)

... 18 more

**Figure 1.11 – Console Output of RPC Error**