SwingBench 2.2
Reference and User Guide

Author: Dominic Giles
**Document Control**

**Authors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominic Giles</td>
<td>Oracle</td>
</tr>
</tbody>
</table>

**Change Record**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft 1</td>
<td>2nd January 2002</td>
<td>Initial Version</td>
</tr>
<tr>
<td>Draft 2</td>
<td>8th April 2003</td>
<td>Update to reflect code improvements</td>
</tr>
<tr>
<td>Draft 3</td>
<td>8th July 2003</td>
<td>New functionality detailed</td>
</tr>
<tr>
<td>Draft 4</td>
<td>3rd November 2003</td>
<td>New functionality detailed</td>
</tr>
<tr>
<td>Draft 5</td>
<td>1st December 2003</td>
<td>Information on default benchmarks and wizards included.</td>
</tr>
<tr>
<td>Draft 6</td>
<td>5th January 2004</td>
<td>Updated to reflect new functionality for 2.1f</td>
</tr>
<tr>
<td>Draft 7</td>
<td>8th August 2005</td>
<td>Update to reflect 2.2</td>
</tr>
</tbody>
</table>

**Reviewers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Nangle</td>
<td>Oracle</td>
</tr>
</tbody>
</table>

---

*SwingBench Reference and User Guide*
Purpose Of Document
The purpose of this document is to detail the functionality and operation of the *swingbench* test harness.

Introduction
The UK based Oracle Database Solutions group have developed “*swingbench*”, a Java based test harness, to demonstrate the stress testing of Oracle Databases. *Swingbench* enables developers to define their own classes by implementing a simple interface. These classes are then loaded and run by *swingbench* according to the parameters defined by the user.

This tool is not intended in anyway as a replacement for commercial load generators, it should be viewed as a free alternative to demo certain features of the Oracle database. The code is continually under development and certain features described in this document are likely to change.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENT CONTROL</td>
<td>2</td>
</tr>
<tr>
<td>PURPOSE OF DOCUMENT</td>
<td>3</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>4</td>
</tr>
<tr>
<td>SWINGBENCH</td>
<td>4</td>
</tr>
<tr>
<td>OVERVIEW</td>
<td>6</td>
</tr>
<tr>
<td>NEW IN 2.2</td>
<td>7</td>
</tr>
<tr>
<td>DIRECTORY STRUCTURE</td>
<td>8</td>
</tr>
<tr>
<td>INSTALLATION</td>
<td>9</td>
</tr>
<tr>
<td>RUNNING SWINGBENCH</td>
<td>10</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>12</td>
</tr>
<tr>
<td>User Interface (Swingbench only)</td>
<td>12</td>
</tr>
<tr>
<td>Configuration Tab</td>
<td>12</td>
</tr>
<tr>
<td>Output Tab</td>
<td>13</td>
</tr>
<tr>
<td>Events Tab</td>
<td>14</td>
</tr>
<tr>
<td>Transaction Panel</td>
<td>14</td>
</tr>
<tr>
<td>Graphing Panel</td>
<td>15</td>
</tr>
<tr>
<td>XML Configuration File</td>
<td>17</td>
</tr>
<tr>
<td>ConnectionInformation</td>
<td>18</td>
</tr>
<tr>
<td>TransactionList</td>
<td>18</td>
</tr>
<tr>
<td>CoordinatorInformation</td>
<td>19</td>
</tr>
<tr>
<td>Charts</td>
<td>19</td>
</tr>
<tr>
<td>ConnectionInitializationCommands</td>
<td>19</td>
</tr>
<tr>
<td>AllowedErrorCodes</td>
<td>19</td>
</tr>
<tr>
<td>Statistics</td>
<td>19</td>
</tr>
<tr>
<td>EnvironmentVariables</td>
<td>20</td>
</tr>
<tr>
<td>Command Line Options</td>
<td>21</td>
</tr>
<tr>
<td>Examples</td>
<td>22</td>
</tr>
<tr>
<td>DEVELOPING TRANSACTIONS</td>
<td>23</td>
</tr>
<tr>
<td>PL/SQL stubs</td>
<td>23</td>
</tr>
<tr>
<td>Developing Java Transactions</td>
<td>25</td>
</tr>
<tr>
<td>SUPPLIED BENCHMARKS</td>
<td>28</td>
</tr>
<tr>
<td>ANTERA BENCHMARK WALK THROUGH</td>
<td>31</td>
</tr>
<tr>
<td>Step 1 (Set Up the Environment)</td>
<td>31</td>
</tr>
<tr>
<td>Step 2 (Create the Calling Circle Schema)</td>
<td>31</td>
</tr>
<tr>
<td>Step 3 (Generate Data for a Benchmark Run)</td>
<td>33</td>
</tr>
<tr>
<td>Optional Step (Start CPU Monitor on target system)</td>
<td>34</td>
</tr>
<tr>
<td>Alternative Step 4a (Run Swinbench with command line options)</td>
<td>35</td>
</tr>
<tr>
<td>Alternative Step 4b (Modify the swingconfig.xml File)</td>
<td>36</td>
</tr>
<tr>
<td>Step 5 (Run the Swingbench Load Generator)</td>
<td>36</td>
</tr>
</tbody>
</table>
SwingBench

Overview

Swingbench is designed to stress test a database by simulating a workload using user-defined transactions. The user can control the number of users (threads) that attach to a database and the amount and type of work they perform. Users can dynamically monitor the response times and load which is displayed in a series of graphs.

Swingbench Load Generator
**New in 2.2**
The latest release of swingbench 2.2 introduces a number of new features these include.

- A new lightweight graphical load generator called "minibench"
- The coordinator now has a number of command line options (start, stop, status)
- The coordinator can now be run in either graphical or character mode
- The charting engine now uses Oracle's BI-Beans graphing technology
- Better exception handling and error messages in both swingbench and clusteroverview
- Users can now turn off jumping to the events panel in swingbench
- The user chart in clusteroverview now allows users to specify monitored users
- Swingbench can logon/logoff users between transactions (experimental)
- Minor structural changes to swingconfig.xml
- New command line options for coordinator/swingbench/minibench/charbench
- CPU monitor output in charbench
- Simpler configuration for multiple load generators
- Fixes to clusteroverview
- Fixes to wizards
- The order entry benchmark can now be scaled to 100GB
- CPU monitor for database nodes in clusteroverview/swingbench
**Directory Structure**

By default the directory structure for the swingbench framework is as follows.

Windows users should use the winbin directory when running swingbench scripts. The scripts used inside of the bin directory use the bash shell, these will need to be modified if the OS does not support this shell (i.e HP UX).

The top level directory (home directory) is referred to as $SWINGHOME in this document.
Installation

To run swingbench a Java Virtual Machine (JVM) must be installed on the client platform. The author currently recommends using the latest available 1.4 JVM for the platform. An Oracle client must also be available, this can either be in the form of a full blown Oracle database install or the Oracle instant client down loadable from the Oracle technology network


Swingbench is supplied in a single zip file. To uncompress this file issue the command (Unix/Linux)

```
[oracle@dgiles-uk swingbench]$ unzip swingbench
```

On Windows use a tool such as WinZip to perform this operation.

The default installation of swingbench is performed by modifying the values in the $SWINGHOME/swingbench.env file under Linux/Unix and in $SWINGHOME/swingbenchenv.bat file under Windows. The contents of an example swingbench.env are shown below.

```
#!/bin/bash
export ORACLE_HOME=/home/oracle/orabase/product/10g
export JAVAHOME=/usr/java/j2sdk1.4.2_08
export SWINGHOME=/home/oracle/swingbench
export ANTHOME=$SWINGHOME/lib
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$ORACLE_HOME/lib

```

The values shown in red need to be modified to reflect the file structure into which the software has been installed.
Running Swingbench

The core kernel of swinbench has three font ends

- Swingbench: A rich graphical font end ideal for demonstrations.
- Minibench: A lighter weight graphical front end useful for testing over remote connections
- Charbench: A character mode front end useful for data collection (importing into spreadsheets)

All of them are capable of running the same benchmarks and use the same infrastructure.

Swingbench can be invoked on Unix/Linux using the commands

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./swingbench
```

Or on Windows using the commands

```
C:\ cd winbin
C:\ swingbench
```

swingbench, minibench and charbench have a number of command line options, these can be displayed using the “-h” option i.e.
usage: parameters:
-D <variable=value>     use value for given environment variable
-a                      run automatically
-c <filename>           specify config file
-co <coordinator>       specify/override coordinator in configuration file. i.e. "//<hostname>/CoordinatorServer"
-cpuloc <CPUmonitor>    specify/override location of the cpu monitor. Value is in the form "//<hostname>/CPUMonitor"
-cs <connectstring>    override connect string in configuration file
-dt <drivertype>       override driver type in configuration file. Value is either "thin" or "oci"
-h,--help              print this message
-i                      run interactively (default)
-max <milliseconds>     override maximum think time in configuration file
-min <milliseconds>     override minimum think time in configuration file
-p <password>          override password in configuration file
-r <filename>          specify results file
-u <username>          override username in configuration file
-uc <number>           override user count in configuration file.
**Configuration**

Swing Bench is initialized in one of three ways, via an XML configuration file, manually entering new parameters into the user front end or by using command line options at startup. Because of the complexities of building a fully functional user interface the most complete method of initializing the test harness is via the XML file. All three methods will be discussed in the following sections.

**User Interface (Swingbench only)**

The user interface is composed of four sections

- The configuration, results and events panel (left hand side), this is responsible for the entry and updating of parameters such as user populations, connect strings, think times etc.
- The transaction panel (top right hand), maintains a list of the current active transactions
- The graphing panel (bottom right hand), displays results such as transactions per minute and average response time.
- The menu, this provides similar functionality to the toolbar with the addition of more save options

Users can navigate to each of these panels via the mouse. Changes to the current configuration by loading new versions from the menu.

**Configuration Tab**

Users can enter data relating to a benchmark run within the “Configuration” tab

![Configuration Tab](image)

**Configuration Tab**
A description of each of the fields is described in the following table

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>The username to which each of the users (threads) will connect.</td>
</tr>
<tr>
<td>Password</td>
<td>The password for the user.</td>
</tr>
<tr>
<td>Connect String</td>
<td>The jdbc connect string. This will have the following format if the thin drivers are used: &quot;&lt;&lt;Host&gt;&gt;:&lt;&lt;Port&gt;&gt;:&lt;&lt;SID&gt;&gt;&quot; or &quot;//&lt;&lt;Host&gt;&gt;:&lt;&lt;Port&gt;&gt;/&lt;&lt;Service&gt;&gt;&quot;, or a simple TNS names entry if OCI drivers are used. User may also use the new easy connect format when running against an Oracle10g database.</td>
</tr>
<tr>
<td>Driver Type</td>
<td>Used to tell <em>swingbench</em> which driver to use. The OCI driver requires a Oracle client to be installed. The thin driver emulates Oracle Net packets.</td>
</tr>
<tr>
<td>No. of Users</td>
<td>The total number of users that Swing Bench will attempt to connect to the database (can be changed during benchmark run)</td>
</tr>
<tr>
<td>Connection option</td>
<td>This drop down list defines whether users will all have their own connection or use a shared pool of connections (implemented using the Oracle10g datasource implementation)</td>
</tr>
<tr>
<td>Pooled</td>
<td>The number of physical connections used (only valid if pooled connections are used)</td>
</tr>
<tr>
<td>Min. Think Time</td>
<td>The minimum think time (delay) between transactions (ms)</td>
</tr>
<tr>
<td>Max Think Time</td>
<td>The maximum think time (delay) between transactions (ms)</td>
</tr>
<tr>
<td>Max Trans</td>
<td>The maximum number of transactions to be run (approximate)</td>
</tr>
<tr>
<td>Logon Delay</td>
<td>The time between session connecting to the database (ms)</td>
</tr>
<tr>
<td>Wait until users logged on</td>
<td>Indicates whether transactions should be started before all of the user population is logged on.</td>
</tr>
<tr>
<td>Query Timeout</td>
<td>The time before <em>swingbench</em> considers a transaction to have failed.</td>
</tr>
<tr>
<td>Output Option</td>
<td>Indicated whether the results should be send to the output tab, a user defined file or the OS’s standard output.</td>
</tr>
<tr>
<td>File Name</td>
<td>The output file if relevant.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Indicates whether full or minimal statistics are collected. A full collection requires more runtime memory.</td>
</tr>
</tbody>
</table>
Output Tab

The “Output” tab is used to display the results of a completed benchmark run when the user selects the “output” tab in the configuration panel. The results are displayed in simple XML document.

```
<Configuration>...<Output>...<Results>...
<TotalRunTimeSec>748,0</TotalRunTimeSec>
<TotalLogTimeSec>25</TotalLogTimeSec>
<TotalCompletedTransactions>3273</TotalCompletedTransactions>
<TotalFailedTransactions>0</TotalFailedTransactions>
<TotalAbortedTransactions>0</TotalAbortedTransactions>
</Results>
</Output>
</Configuration>
```

Events Tab

The “Events” tab displays information generated by swingbench indicating the status or errors within a benchmark run. Typical information might include the failure of a session due to connection issues or lack of data. By default swingbench will automatically jump to this screen whenever a new event occurs, this can be disabled by modifying the configuration file (described in a later section).
Transaction Panel

The transaction Panel allows users to enter and update transaction that are to be executed by the simulated user sessions (threads). Users may enter as many transactions as they wish and specify the profile of how often they are likely to be run in comparison to other transactions. Transactions can be added or removed using the two buttons at the top of the table. Currently changes in the profile only take effect at the start of a new benchmark run.

<table>
<thead>
<tr>
<th>Id</th>
<th>Class Name</th>
<th>Type</th>
<th>Load</th>
<th>Activate</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Customer Registration...</td>
<td>com.dom.benchmarking...Transaction</td>
<td></td>
<td>Load (0 to 100) 20</td>
<td></td>
</tr>
<tr>
<td>Browse Products</td>
<td>com.dom.benchmarking...Transaction</td>
<td></td>
<td>Load (0 to 100) 50</td>
<td></td>
</tr>
<tr>
<td>Order Products</td>
<td>com.dom.benchmarking...Transaction</td>
<td></td>
<td>Load (0 to 100) 50</td>
<td></td>
</tr>
</tbody>
</table>

Transaction Panel

The following table describes the columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique identifier for the transaction/process</td>
</tr>
<tr>
<td>Class Name</td>
<td>A fully specified class name, identifying the class to be loaded</td>
</tr>
<tr>
<td>Type</td>
<td>Indicating whether the class is a simple transaction</td>
</tr>
<tr>
<td>Load</td>
<td>Indicates the “weight” of the transaction in comparison to other transactions. A higher weight indicates that it more likely to be run</td>
</tr>
<tr>
<td>Activate</td>
<td>Whether this transaction is to be executed in this particular run.</td>
</tr>
</tbody>
</table>

Graphing Panel

This panel is used to display the current activity of a target system. It also indicates the percentage of the users currently logged on (top right hand of the panel)

TPM Graph  Response Time Graph  CPU/TPM Graph

The following graphs are available in the current release
- Transaction Throughput Graph displays the number of transactions that have completed per minute. These transactions may have been explicitly declared or be implicit in the processes that are running.
- Nested Transaction Throughput Graph displays the number of nested transactions that have completed per minute. Nested Transactions are those that are dynamically registered by a parent transaction.
- Transaction Maximum, Minimum and Average Graph dynamically displays the response times of transactions.
- Nested Transaction Maximum, Minimum and Average Graph displays the response times of nested transactions.
- DML Throughput Graph displays the selects, inserts, updates, deletes, commits and rollbacks.
- CPU History (likely to be deprecated) displays the current CPU load on the target system since the current swingbench session has started.
- CPU and Transaction Overview Graph provides an overview of the target cpuload overlaid against the total transaction load (requires a cpu monitor running on the target system).
XML Configuration File

By default swingbench reads its configuration properties from a file called swingconfig.xml located in the $SWINGHOME/bin/sample directory which can be copied and edited. Users can edit this file to persist changes. There are also sample files located in the $SWINGHOME/bin/sample directory which can be copied and edited. Users can specify different configuration files by using the “-c” option at startup of swingbench/minibench/charbench.

```
<?xml version='1.0'?>
<Swing Bench Configuration Name="Order Entry (PLSQL)" StartMode="Manual"
Output="true" JmpToEvents="false">
  <Connection Information>
    <UserName>soe</UserName>
    <Password>soe</Password>
    <ConnectionString>/localhost:1521/DOM102</ConnectionString>
    <DriverType>thin</DriverType>
    <NumberOfUsers>15</NumberOfUsers>
    <MinNumberOfUsers>1</MinNumberOfUsers>
    <MaxNumberOfUsers>30</MaxNumberOfUsers>
    <Pooled>false</Pooled>
    <LogonDelay>0</LogonDelay>
    <LogOutPostTransaction>false</LogOutPostTransaction>
  </Connection Information>
  <Transaction List>
    <Transaction Id="New Customer Registration" ShortName="NCR"
      SourceFile="com.dom.benchmark.swingbench.plsqltransactions.NewCustomerProcess"
      Weight="20" Enabled="true"/>
    <Transaction Id="Browse Products" ShortName="BF"
      SourceFile="com.dom.benchmark.swingbench.plsqltransactions.BrowseProducts"
      Weight="50" Enabled="true"/>
    <Transaction Id="Order Products" ShortName="OP"
      SourceFile="com.dom.benchmark.swingbench.plsqltransactions.NewOrderProcess"
      Weight="50" Enabled="true"/>
    <Transaction Id="Process Orders" ShortName="PO"
      SourceFile="com.dom.benchmark.swingbench.plsqltransactions.ProcessOrders"
      Weight="10" Enabled="true"/>
    <Transaction Id="Browse Orders" ShortName="BO"
      SourceFile="com.dom.benchmark.swingbench.plsqltransactions.BrowseAndUpdateOrders"
      Weight="50" Enabled="true"/>
  </Transaction List>
  </Pearson Information>
  <CPU Monitor/>
  <Charts>
    <Chart Name="Transactions per Minute" Autoscale="true"
      Maximum Value="-1.0"/>
    <Chart Name="DML Operations per Minute" Autoscale="true"
      Maximum Value="-1.0"/>
    <Chart Name="Transactions Maximum, Minimum and Average" Autoscale="true"
      Maximum Value="-1.0"/>
  </Charts>
  <Connection Initialization Commands>
    <Command Type="Connection Property">BatchUpdates=1</Command>
    <Command Type="Connection Property">FetchSize=1</Command>
    <Command Type="Connection Property">StatementCaching=50</Command>
    <Command Type="SQL Command">alter session set sql_trace = false</Command>
    <Command Type="SQL Command">alter session set optimizer_mode = first_rows</Command>
  </Connection Initialization Commands>
  <Allowed Error Codes/>
  <Environment Variables>
    <Variable Key = "SOEPRODUCTSDATA_LOC" Value = "data/productids.txt"/>
    <Variable Key = "SOENAMESDATA_LOC" Value = "data/names.txt"/>
    <Variable Key = "SOENLSDATA_LOC" Value = "data/nls.txt"/>
  </Environment Variables>
  <Statistics Collection Type="Minimal"/>
</Swing Bench Configuration>
```
The use of an XML file offers more functionality to the user. It has eight main sections:

- ConnectionInformation
- TransactionList
- CoordinatorInformation
- Charts
- ConnectionInitializationCommands
- AllowedErrorCodes
- Statistics
- EnvironmentVariables

The following sections describe their use and attributes.

**ConnectionInformation**

```xml
<ConnectionInformation>
  <UserName>soe</UserName>
  <Password>soe</Password>
  <ConnectionString>/\Localhost:1521/DOM102</ConnectionString>
  <DriverType>thin</DriverType>
  <NumberOfUsers>15</NumberOfUsers>
  <MinNumberOfUsers>1</MinNumberOfUsers>
  <MaxNumberOfUsers>30</MaxNumberOfUsers>
  <Pool>-1</Pool>
  <LogonDelay>0</LogonDelay>
  <LogOutPostTransaction>true</LogOutPostTransaction>
</ConnectionInformation>
```

As with the user interface, the “ConnectionInformation” node enables users to define the connection and nature of the connections to the database. All of the node’s attributes and child nodes should be considered mandatory. The “ConnectionString” element can be either be a valid oci connectstring or a java thin connectstring of the form `<hostname>::<port>::<SID>`. Users can also use Oracle10g's easy connect of the form `//<<hostname>>/<<service>>`. The “LogonDelay” element, specified in milliseconds, determines how long swingbench should wait between logging users on to the system. A value of –1 for MaxTransactions and Pooled indicates that this functionality is not to be used.

**TransactionList**

```xml
<TransactionList WaitTillAllLogon="true" MinDelay="250" MaxDelay="750"
MaxTransactions="-1" QueryTimeout="60">
  <Transaction Id="New Customer Registration" ShortName="NCR"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.NewCustomerProcess"
Weight="20" Enabled="true/>
  <Transaction Id="Browse Products" ShortName="BP"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.BrowseProducts"
Weight="50" Enabled="true"/>
  <Transaction Id="Order Products" ShortName="OP"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.NewOrderProcess"
Weight="50" Enabled="true"/>
  <Transaction Id="Process Orders" ShortName="PO"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.ProcessOrders"
Weight="10" Enabled="true"/>
  <Transaction Id="Browse Orders" ShortName="BO"
SourceFile="com.dom.benchmarking.swingbench.plsqltransactions.BrowseAndUpdateOrders"
Weight="50" Enabled="true"/>
</TransactionList>
```

The “TransactionList” describes which transactions are to be executed, where they are located and how often they are to be run. A transaction may be based on the same source file as other processes in the “TransactionList” but must have a unique “Id”. As with the user Interface the “Weight” should be between 0 and 100. The “ShortName” attribute of “Transaction” is used by charbench when displaying the transactions in a table like format.
CoordinatorInformation

The CoordinatorInformation node describes the location of a Java RMI process responsible for the coordination of multiple swingbench load generators. This functionality allows a much larger load to be run against a single database or clustered databases.

Charts

The Charts node describes the way data is charted in the various graphs. Users can set maximum values and disable autoscaling if they know the profile of the benchmark they are running. This can make it easier for side by side comparisons.

ConnectionInitializationCommands

The ConnectionInitializationCommands node allows users to initialize a connection before any transactions are executed against it. This is useful for the debugging of performance problems and to take advantage of vendor specific optimizations. Two ConnectionInitializationCommands child node types exist, Connection Property and SQL Command. The first sets properties specific to the connection itself and is usually vendor specific, only two properties are available in this release BatchUpdates and FetchSize. The latter enables users to set dynamic database properties such as sql_trace, the full SQL command should be included.

AllowedErrorCodes

The AllowedErrorCodes node simply allows users to indicate that certain database errors should be ignored and not displayed the console. This is useful if the transactions are allowed to insert bogus values into the database to simulate operator error.

Statistics

The Statistics node simply allows users to indicate what type of statistics are collected.
The Statistics node specifies the type of timings that are recorded. It can currently have two values “Minimal” or “Full”, when “Minimal” is specified only minimum, maximum and averages are collected for each type of transaction or process.

EnvironmentVariables

```
<EnvironmentVariables>
  <Variable Key="SOE_PRODUCTSDATA_LOC" Value="data/productids.txt"/>
  <Variable Key="SOE_NAMESDATA_LOC" Value="data/names.txt"/>
  <Variable Key="SOE_NLSDATA_LOC" Value="data/nls.txt"/>
</EnvironmentVariables>
```

The EnvironmentVariables section supports the runtime specification of variables used by a given benchmark. Each variable comprises of a key (lookup) and a value. Typically these contain locations of seed data or configuration files. Users who have defined their own transactions may specify their own environmental variables here.

The following environmental variables used by swingbench for the two supplied benchmarks are described here.

<table>
<thead>
<tr>
<th>Environmental Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOE_PRODUCTSDATA_LOC</td>
<td>Location of the product data for the soe benchmark</td>
</tr>
<tr>
<td>SOE_NAMESDATA_LOC</td>
<td>Location of the names data for the soe benchmark</td>
</tr>
<tr>
<td>SOE_NLSDATA_LOC</td>
<td>Location of the nls data for the soe benchmark</td>
</tr>
<tr>
<td>CC_NEWPROCESS_FILE_LOC</td>
<td>Location of the new customer data for the cc benchmark</td>
</tr>
<tr>
<td>CC_QUERYPROCESS_FILE_LOC</td>
<td>Location of the query data for the cc benchmark</td>
</tr>
<tr>
<td>CC_UPDATEPROCESS_FILE_LOC</td>
<td>Location of the updater data for cc benchmark</td>
</tr>
<tr>
<td>CC_DATA_DIR_LOC</td>
<td>Location of the directory containing all of the data files for the cc benchmark (can be used instead of the three individual variables above)</td>
</tr>
</tbody>
</table>

SwingBench Reference and User Guide
**Command Line Options**

Most of the parameters inside of swingbench can be overwritten from the command line. Command line options can be listed by invoking swingbench/minibench/charbench with the “-h” option.

```
[oracle@dgiles-uk bin]$ ./swingbench -h
usage: parameters:
-D <variable=value> use value for given environment variable
-a run automatically
-c <filename> specify config file
-co <coordinator> specify/override coordinator in configuration file. i.e. "//<hostname>/CoordinatorServer"
-cpuloc <CPUMonitor > specify/override location of the cpu monitor. Value is in the form "/<hostname>/CPUMonitor"
-cs <connectstring> override connect string in configuration file
-dt <drivertype> override driver type in configuration file. Value is either "thin" or "oci"
-h,--help print this message
-i run interactively (default)
-max <milliseconds> override maximum think time in configuration file
-min <milliseconds> override minimum think time in configuration file
-p <password> override password in configuration file
-r <filename> specify results file
-u <username> override username in configuration file
-uc <number> override user count in configuration file.
```

The character version of swingbench has a few additional parameters

```
usage: parameters:
-D <variable=value> use value for given environment variable
-a run automatically
-c <filename> specify config file
-co <coordinator> specify/override coordinator in configuration file. i.e. "//<hostname>/CoordinatorServer"
-cpuloc <CPUMonitor > specify/override location of the cpu monitor. Value is in the form "/<hostname>/CPUMonitor"
-cs <connectstring> override connect string in configuration file
-d <seconds> delay between transaction samples in seconds
-dt <drivertype> override driver type in configuration file. Value is either "thin" or "oci"
-h,--help print this message
-i run interactively (default)
-max <milliseconds> override maximum think time in configuration file
-min <milliseconds> override minimum think time in configuration file
-p <password> override password in configuration file
-r <filename> specify results file
-s run silent
-u <username> override username in configuration file
-uc <number> override user count in configuration file.
-v display run statistics (vmstat/sar like output)
-vc display run statistics including cpu load
   (requires cpu monitor running on server)
-vd display run statistics including DML values
   (vmstat/sar like output)
-vt display run statistics including transaction values (vmstat/sar like output)
```
Examples

The following example launches swingbench changing its connect string to use the easy connect string 
“//kgy11034/db10g2”. The “-dt” option instructs swingbench to use the oci driver to establish the 
connections. The “-min” and “-max” options tell swingbench to use a minimum think time of 200 
milliseconds and a maximum of 5000 milliseconds. The “-cpuloc” tells swingbench the location of a cpu 
monitor. The “-c” option tells swingbench to use the sample config file “soeconfig.xml”.

```
[oracle@dgiles-uk:bin] $ ./swingbench  -cs //kgy11034/db10g2 -dt oci -min 200 
-max 5000 -uc 200 -cpuloc //kgy11034/CPUMonitor -c sample/soeconfig.xml
```

The next example launches charbench with the options discussed in the previous example with the following 
addition of the “-r” option which tells swingbench to put the results of the run in a file called “doms.txt”. The 
“-a” option starts charbench running automatically without any user intervention. The “-vc” option displays 
both the transaction profile as well as the cpu load.

```
$ >./charbench -c sample/soeconfig.xml -cs //kgy11034/db10g2 -dt oci -min 200 
-max 5000 -uc 200 -cpuloc //kgy11034/CPUMonitor -r doms.txt -a -vc
```

Results will be written to doms.txt.

<table>
<thead>
<tr>
<th>Time</th>
<th>Users</th>
<th>TPM</th>
<th>Nested TPM</th>
<th>User</th>
<th>System</th>
<th>Wait</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:23:46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10:23:48</td>
<td>44</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>19</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>10:23:50</td>
<td>134</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10:23:52</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>24</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>10:23:54</td>
<td>200</td>
<td>7</td>
<td>0</td>
<td>15</td>
<td>8</td>
<td>5</td>
<td>71</td>
</tr>
<tr>
<td>10:23:56</td>
<td>200</td>
<td>27</td>
<td>0</td>
<td>32</td>
<td>13</td>
<td>10</td>
<td>45</td>
</tr>
<tr>
<td>10:23:58</td>
<td>200</td>
<td>41</td>
<td>0</td>
<td>25</td>
<td>7</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>10:24:00</td>
<td>200</td>
<td>67</td>
<td>0</td>
<td>44</td>
<td>9</td>
<td>2</td>
<td>44</td>
</tr>
<tr>
<td>10:24:03</td>
<td>200</td>
<td>84</td>
<td>0</td>
<td>46</td>
<td>6</td>
<td>2</td>
<td>46</td>
</tr>
<tr>
<td>10:24:05</td>
<td>200</td>
<td>99</td>
<td>0</td>
<td>38</td>
<td>3</td>
<td>4</td>
<td>55</td>
</tr>
<tr>
<td>10:24:07</td>
<td>200</td>
<td>121</td>
<td>0</td>
<td>51</td>
<td>4</td>
<td>4</td>
<td>42</td>
</tr>
<tr>
<td>10:24:09</td>
<td>200</td>
<td>149</td>
<td>0</td>
<td>38</td>
<td>3</td>
<td>2</td>
<td>57</td>
</tr>
<tr>
<td>10:24:11</td>
<td>200</td>
<td>166</td>
<td>0</td>
<td>23</td>
<td>4</td>
<td>7</td>
<td>67</td>
</tr>
<tr>
<td>10:24:13</td>
<td>200</td>
<td>187</td>
<td>0</td>
<td>23</td>
<td>3</td>
<td>6</td>
<td>68</td>
</tr>
<tr>
<td>10:24:15</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td>10:24:17</td>
<td>200</td>
<td>224</td>
<td>0</td>
<td>22</td>
<td>3</td>
<td>5</td>
<td>71</td>
</tr>
<tr>
<td>10:24:19</td>
<td>200</td>
<td>244</td>
<td>0</td>
<td>13</td>
<td>6</td>
<td>5</td>
<td>77</td>
</tr>
<tr>
<td>10:24:21</td>
<td>200</td>
<td>261</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>83</td>
</tr>
<tr>
<td>10:24:23</td>
<td>200</td>
<td>277</td>
<td>0</td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>62</td>
</tr>
<tr>
<td>10:24:25</td>
<td>200</td>
<td>296</td>
<td>0</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>78</td>
</tr>
<tr>
<td>10:24:27</td>
<td>200</td>
<td>312</td>
<td>0</td>
<td>15</td>
<td>3</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>10:24:29</td>
<td>200</td>
<td>340</td>
<td>0</td>
<td>16</td>
<td>5</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td>10:24:31</td>
<td>200</td>
<td>361</td>
<td>0</td>
<td>21</td>
<td>3</td>
<td>8</td>
<td>68</td>
</tr>
<tr>
<td>10:24:33</td>
<td>200</td>
<td>389</td>
<td>0</td>
<td>27</td>
<td>4</td>
<td>3</td>
<td>66</td>
</tr>
<tr>
<td>10:24:35</td>
<td>200</td>
<td>414</td>
<td>0</td>
<td>26</td>
<td>4</td>
<td>4</td>
<td>67</td>
</tr>
<tr>
<td>10:24:37</td>
<td>200</td>
<td>436</td>
<td>0</td>
<td>30</td>
<td>5</td>
<td>3</td>
<td>62</td>
</tr>
</tbody>
</table>
Developing Transactions

There are two approaches to developing transactions with the swingbench framework, developers can either use a series of PL/SQL stubs to add their own code or create java transactions from scratch. Whilst the first approach requires some knowledge of PL/SQL it is relatively simply to build a fully functional benchmark. The second approach is to modify or add code to the existing transactions under the source directory. Both approaches are described in the following sections.

PL/SQL stubs

To install the PL/SQL stubs run the script `storedprocedures.sql` in the sql directory using sqlplus against a schema of your choice i.e.

```bash
[oracle@dgiles-uk sql]$ sqlplus dom/dom
SQL*Plus: Release 9.2.0.4.0 - Production on Tue Nov 11 12:48:06 2003
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production
SQL> start storedprocedures.sql
Type created.

Package created.

Package body created.

SQL>
```

This will create a type called `integer_return_array` and a new package called `swingbench`. Users can edit the package body of `swingbench` to add transactions that implement their own functionality. By default users can use six predefined functions to create new transactions, this can be extended my modifying the java code in the java package `com.dom.benchmarking.swingbench.storedprocedures`.

The six PL/SQL functions can be edited using tools such as Oracle Enterprise Manager, Oracle Jdeveloper, TOAD etc.
There are six main functions that the developer is free to modify are storedprocedure1 through storedprocedure6, as seen below:

```sql
function storedprocedure1(min_sleep integer, max_sleep integer) return
integer_return_array is
    begin
        init_dml_array();
        sleep(min_sleep, max_sleep);
        return dml_array;
    end storedprocedure1;
```

The code that ships is simply a stub. It can be trivially modified to include the users own set of SQL operations or calls to other procedures/functions. For example to simple select the number of customers the code could be modified as shown below:

```sql
function storedprocedure1(min_sleep integer, max_sleep integer) return
integer_return_array is
    number_of_customers integer := 0;
    begin
        init_dml_array();
        select count(1) into number_of_customers
        from customers;
        increment_selects(1);
        sleep(min_sleep, max_sleep);
        return dml_array;
    end storedprocedure1;
```

The `init_dml()` call simply resets the integer array that is returned from the function. It is used to record the total number/type of DML operations that are performed in your code. The `increment_selects()` call updates the number of select operations that you've performed in this case one. The package comes with a function appropriate for each type of standard database operation, select, insert, update, delete, commit, rollback. Whilst its not necessary to update the array it is necessary to return a integer array (`dml_array`) at the end of each function.
Developing Java Transactions

As described earlier in this document user defined transactions can be of two types either simple atomic operations or more complex processes that may have many atomic transactions contained within them. To create a user defined transaction developers must first implement a interface.

```java
public interface Task {
    public static final String JDBC_CONNECTION = "jdbcConnection";
    public static final String QUERY_TIMEOUT = "queryTimeout";

    public void init(Map param) throws SwingBenchException;
    public void execute(Map param) throws SwingBenchException;
    public void close();
    public void addTaskListener(TaskListener transListener);
    public void removeTaskListener(TaskListener transListener);
    public void processTransactionEvent(boolean transactionSuccessful,
                                        long transactionPeriod, String id);
    public String getId();
    public void setId(String newProcessName);
    public void setThinkSleepTime(long newMinSleepTime, long newMaxSleepTime);
}
```

To simplify transaction creation an abstract class JdbcTaskImpl implements all but the init, execute and close methods. This makes the creation of Tasks a fairly simple operation by allowing the developer to simply extend its functionality.

The following section illustrates a simple transaction that calls a stored procedure orderentry.browseandupdateorders.
package com.dom.benchmarking.swingbench.plsqltransactions;
import com.dom.benchmarking.swingbench.event.JdbcTaskEvent;
import com.dom.benchmarking.swingbench.kernel.SwingBenchException;
import com.dom.benchmarking.swingbench.kernel.Task;
import com.dom.benchmarking.swingbench.utilities.RandomGenerator;
import com.protomatter.syslog.Syslog;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Map;
import oracle.jdbc.OracleTypes;
import oracle.sql.ARRAY;

public class BrowseAndUpdateOrders extends OrderEntryProcess {
    public BrowseAndUpdateOrders() {}
    public void close() {
    }
    public void init(Map params) {
        Connection connection = (Connection)params.get(Task.JDBC_CONNECTION);
        try {
            this.getMaxandMinCustID(connection);
        } catch (SQLException se) {
            Syslog.error(this, se);
        }
    }
    public void execute(Map params) throws SwingBenchException {
        Connection connection = (Connection)params.get(Task.JDBC_CONNECTION);
        int queryTimeOut = 60;
        if (params.get(Task.QUERY_TIMEOUT) != null)
            queryTimeOut = ((Integer)(params.get(Task.QUERY_TIMEOUT))).intValue();
        long executeStart = System.currentTimeMillis();
        int[] dmlArray = null;
        try {
            long start = System.currentTimeMillis();
            try {
                CallableStatement cs = connection.prepareCall("{? = call orderentry.browseandupdateorders(?,?,?)}");
                cs.registerOutParameter(1, OracleTypes.ARRAY, "INTEGER_RETURN_ARRAY");
                cs.setInt(2, RandomGenerator.randomInteger(MIN_CUSTID, MAX_CUSTID));
                cs.setInt(3, (int) this.getMinSleepTime());
                cs.setInt(4, (int) this.getMaxSleepTime());
                cs.setQueryTimeout(queryTimeOut);
                cs.executeUpdate();
                dmlArray = (((ARRAY) cs.getArray(1)).getIntArray());
                cs.close();
            } catch (SQLException se) {
                throw new SwingBenchException(se.getMessage());
            } catch (SwingBenchException ex) {
                processTransactionEvent(new JdbcTaskEvent(this, getId(),
                    (System.currentTimeMillis() - executeStart), false,
                    dmlArray));
                throw new SwingBenchException(ex);
            }
        } catch (SwingBenchException ex) {
            processTransactionEvent(new JdbcTaskEvent(this, getId(),
                (System.currentTimeMillis() - executeStart), true,
                dmlArray));
            throw new SwingBenchException(ex);
        }
    }
}

The code first initializes its values in the init() routine which is called at the first loading of the class. This gives the developer the chance to read in values (seed data) from the file system or database. The execute
method is responsible for executing the jdbc operations and telling the swingbench framework how long it took to process. The code notifies the framework of a successful or failed transaction by calling processTransactionEvent and passing it a JdbcTaskEvent.

The execute method is passed a hash map containing parameters for its execution. For jdbc transactions it will always contain two key pairs, the jdbc connection (JDBC_CONNECTION) and the query timeout (QUERY_TIMEOUT). Further values can also be passed by including them in the EnvironmentVariables element in the swingconfig.xml. (described earlier)

The default swingbench environment ships with the source code for 4 benchmarks

- CallingCircle
- OrderEntry (PL/SQL)
- OrderEntry (jdbc)
- PL/SQL stubs

The java source is located in the $SWINGBENCHHOME/source directory along with a script (ant) to compile all of the code. The following shows its compilation of all of the supplied source.

```
[oracle@dgiles-uk source]$ ./antbuild
Buildfile: /home/oracle/java/SwingBench/swingbench/source/build.xml
init:
    [mkdir] Created dir: /home/oracle/java/SwingBench/swingbench/classes
compile:
    [javac] Compiling 26 source files to /home/oracle/java/SwingBench/swingbench/classes
dist:
    [jar] Building jar: /home/oracle/java/SwingBench/swingbench/lib/mytransactions.jar
BUILD SUCCESSFUL
Total time: 4 seconds
```

The script will compile all of the java under the source directory and create a file called mytransaction.jar which is placed in the $SWINGHOME/lib directory. This file contains the transactions in compiled form (class). The default configuration file for swingbench (swingbench.env, swingbenchenv.bat) will use the transactions specified in mytransactions before using the default shipped code.

To use your own java transactions simply include the attribute “SourceFile” of the “Transaction” element in the swingconfig.xml file.

```xml
<TransactionList WaitTillAllLogon="true" MinDelay="250" MaxDelay="750">
    <Transaction Id="HR Transaction : Add Employee" ShortName="AE" SourceFile="com.daves.transaction.HR.addemployee" Weight="100" Enabled="true" />
    <Transaction Id="HR Transaction : Update Employee" ShortName="AE" SourceFile="com.daves.transaction.HR.updemployee" Weight="100" Enabled="true" />
</TransactionList>
```
**Supplied Benchmarks**

Swingbench ships with three functional benchmarks and one template PL/SQL benchmark. These are described below

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Description</th>
<th>Profile</th>
</tr>
</thead>
</table>
| CallingCircle              | The Calling Circle application represents a self-service OLTP application. The application models the customers of a telecommunications company registering, updating and inquiring on a calling circle of their most frequently called numbers in order to receive discounted call pricing. It uses three files: newccprocess.txt, qryccprocess.txt, updccprocess.txt. These contain seed data for each type of transaction in the benchmark. They must be recreated for each run using the ccwizard tool. The location of each file is specified in the swingconfig.xml file in the EnvironmentVariable section. | Large amounts of dynamic PL/SQL. Heavy CPU utilization  
  • Select 83%  
  • Insert 7%  
  • Update 10%  
  • Delete 0%                                                           |
| Order Entry (PL/SQL)       | Models the classic order entry stress test. It has a similar profile to the TPC-C benchmark. This version models an online order entry system with users being required to log-on before purchasing goods. The benchmark uses three files: names.txt, nls.txt, productids.txt. These contain sample data used by each of the transactions. The location of each file is specified in the swingconfig.xml file in the EnvironmentVariable section. | Static PL/SQL with a small table (INVENTORY) that is heavily updated.  
  • Select 50%  
  • Insert 30%  
  • Update 20%  
  • Delete 0% |
| Order Entry (jdbc)         | As Above                                                                                        | Large amounts of jdbc calls. Network/Client intensive  
  • Select 50%  
  • Insert 30%  
  • Update 20%  
  • Delete 0%                                                                 |
| PL/SQL stubs               | Blank PL/SQL stubs provided for users own benchmarks/extensions                                   | PL/SQL based.                                                                                   |

**NOTE:** The only supported way to create the schema's and data for these benchmarks is using the wizards described in the following section.
Wizards

To assist in the creation of the default benchmarks, callingcircle and order entry, swingbench ships with two wizards which step the user through the process of creating, dropping and in the case of the callingcircle benchmark generating data for each run. The wizards can be launched from the $SWINGHOME/bin directory using the following commands (for the callingcircle benchmark).

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./ccwizard
```

And (for the order entry benchmark)

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./oewizard
```

As with the swingbench load generator the the wizards use xml configuration files to maintain a persistent record of a users/database settings for a benchmark. The author recommends that users modify these files to reflect their own configuration. These files are held in the held in the $SWINGHOME/bin directory. The default configuration file for the callingcircle benchmark is displayed below.
Users will typically change values such as their connect string to point to their own instance. From version 2.1f of swingbench users can run the wizards in character mode. This is achieved by changing the “Mode” attribute from “Interactive” to “LightsOut”. In character mode the wizard is driven entirely of the configuration file. The users can change the “operation” attribute to one of three values “create”, “drop” and “generate” to respectively either create a benchmarks schema, drop a existing schema or in the case of the callingcircle benchmark generate data for a new benchmark run.
Appendix A (Simple Benchmark Walk Through)

This appendix describes the process of creating and running a simple benchmark using the calling circle schema. This example assumes the user is using the default install on Unix/Linux.

NOTE: Windows users should use the “.bat” files instead of the unix shell scripts.

Step 1 (Set Up the Environment)

To run the swingbench framework using the default installation you must first edit the swingbench.env file in the $SWINGHOME directory to specify the location of your Java Virtual machine and Oracle Home. An example configuration is shown below

```
#!/bin/bash
export ORACLE_HOME=/home/oracle/orabase/product/10g
export JAVAHOME=/usr/java/j2sdk1.4.2_08
export SWINGHOME=/home/oracle/swingbench
export ANTHOME=$SWINGHOME/lib
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$ORACLE_HOME/lib
export CLASSPATH=$JAVAHOME/lib/rt.jar:$JAVAHOME/lib/tools.jar:
$ORACLE_HOME/jdbc/lib/ojdbc14.jar:$SWINGHOME/lib/mytransactions.jar:$
{SWINGHOME}/lib/swingbench.jar:$ANTHOME/ant.jar
```

Ensure the values in red are set correctly

Step 2 (Create the Calling Circle Schema)

The swingbench framework provides two wizards for the installation of the calling circle and order entry benchmarks. These can be found in the “bin” directory of the default install. The calling circle wizard is launched using the following commands

```
[oracle@dgiles-uk swingbench]$ cd bin
[oracle@dgiles-uk bin]$ ./ccwizard
```

This will launch the following dialogue

![Calling Circle Wizard Dialogue](image)

Welcome to the Calling Circle Benchmark Wizard

This wizard will walk you through the steps to install and maintain a schema for the Calling Circle benchmark. You will need a login with DBA privileges to create the needed table space, users, tables etc.

[Java Port]: Dominic Giles
[Benchmark Author]: Mike Hallas
[Email]: dominic.giles@oracle.com or mike.hallas@oracle.com
Press next and select the “Create the Calling Circle Schema”. Press next again and enter the details of the database in which you wish to create the benchmark schema. Specify a user with DBA privileges.

Press next. Enter the details of the schema, tablespaces and datafiles that will be used to hold the tables.

**NOTE**: If you are using Oracle ASM you only need to specify the name of the storage group for the datafile i.e. “+DATA”.

Press next again. The following dialogue allows the user to specify the size of the benchmark. A larger schema allows more runs before the benchmark must be rebuilt. Users can increase or decrease the customer value by moving the slider (increases logarithmically).
Press next and next again and the benchmark creation will begin. This may take some time depending on the performance of your machine.

**Step 3 (Generate Data for a Benchmark Run)**

The Calling Circle benchmark requires a new set of data to be generated before each run. The CallingCircle Wizard will generate the necessary files for you. Start the wizard as detailed previously. Press next and select the “Generate Data for Benchmark Run”, press next again. The dialog shown below allows you to enter details of the schema created in Step 2.

**HINT** : Edit the ccwizard.xml to specify the defaults for your environment

The “Benchmark Details” step allows you to enter how many transactions will be generated for a run and where the transaction data will be written to.
The “Number of Transactions” field specifies how many will be created per directory location. It also determines the length of a benchmark run. On a 4 processor 1.6 GHz Xeon Intel white box running Linux Advanced server it takes 1 minute to consume 500 transactions, 3 minutes for 2,000, 25 minutes for 20,000. As a result it is important to create enough transactions to last the length of your demonstration.

After specifying the number of transactions, or accepting the default, hit next and then next again. This will begin the benchmark data generation.

After the data generation has completed the wizard will display the hit ratio (NOTE: this is not an error message), if this ratio exceeds 30% you should consider regenerating the CC benchmark schema. There should now be 3 files located in each of the directory locations specified i.e.

```
[oracle@load1 bin]$ ls -l data
total 9872
-rw-r--r-- 1 oracle oinstall 2531642 Apr  4 10:55 newccprocess.txt
-rw-r--r-- 1 oracle oinstall 322500 Apr  4 10:55 qryccprocess.txt
-rw-r--r-- 1 oracle oinstall 7226736 Apr  4 10:55 updccprocess.txt
```

Optional Step (Start CPU Monitor on target system)

It is possible to start a CPU monitor on the target system where the database is located. You'll need to make sure that the swingbench software is also installed on this system (see step 1). After this is completed invoke the cpu monitor with the following command.

```
[oracle@node1 bin]$ ./cpumonitor
CPU monitor started started Successfully
```
Alternative Step 4a (Run Swinbench with command line options)

It is now possible to invoke swingbench with a command line option such as

```
[oracle@load1  bin]$  ./swingbench  -c  sample/ccconfig.xml  -cs  //node1/ORCL  -dt
oci -D  CC_DATA_DIR_LOC=/home/oracle/swingbench/bin/data/ -uc 20
```

or if you started a cpu monitor on the target system.

```
[oracle@load1  bin]$  ./swingbench  -c  sample/ccconfig.xml  -cs  //node1/ORCL  -dt
oci  -D  CC_DATA_DIR_LOC=/home/oracle/swingbench/bin/data/  -uc  20  -cpuloc  //
node1/CPUMonitor
```

This invokes swingbench using the sample callingCircleconfiguration supplied by default but overriding the connectstring and location of the newly generated data files.

Alternatively you could start minibench or charbench using similar parameters

```
[oracle@load1  bin]$  ./charbench  -c  sample/ccconfig.xml  -cs  //node1/ORCL  -dt
oci -D  CC_DATA_DIR_LOC=/home/oracle/swingbench/bin/data/ -uc 20 -vt -a
```

Author : Dominic Giles
Version : 2.2

Results will be written to results.xml.

<table>
<thead>
<tr>
<th>Time</th>
<th>Users</th>
<th>TPM</th>
<th>Nested TPM</th>
<th>NC</th>
<th>UCD</th>
<th>RCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:13:03</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12:13:05</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12:13:07</td>
<td>20</td>
<td>4</td>
<td>88</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>12:13:09</td>
<td>20</td>
<td>5</td>
<td>459</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>12:13:11</td>
<td>20</td>
<td>20</td>
<td>816</td>
<td>0</td>
<td>14</td>
<td>6</td>
</tr>
</tbody>
</table>
**Alternative Step 4b (Modify the swingconfig.xml File)**

Swingbench supplies sample configuration files for all four benchmarks, these are located in the sample directory. To run the callingcircle benchmark first copy the file ccconfig.xml from the sample directory to swingconfig.xml in the bin directory replacing the existing file (make sure you back this up if you intend to use its values later on). You may wish to edit this file to reflect your database i.e. Connect string, username, password, think times etc. You will also need to ensure that the environment variables specific to the callingcircle benchmark are set correctly. These are located in the following section of the configuration file:

```
<EnvironmentVariables>
  <Variable Key="CC_QUERYPROCESS_FILE_LOC" Value="/tmp/qryccprocess.txt"/>
  <Variable Key="CC_UPDATEPROCESS_FILE_LOC" Value="/tmp/updccprocess.txt"/>
  <Variable Key="CC_NEWPROCESS_FILE_LOC" Value="/tmp/newccprocess.txt"/>
</EnvironmentVariables>
```

**Step 5 (Run the Swingbench Load Generator)**

To launch the `swingbench` load generator run the following command:

```
[oracle@dgiles-uk swingbench]$ cd bin/
[oracle@dgiles-uk bin]$ ./swingbench
```

Ensure the details are correct in the configuration panel and start the load with the “Start” button.

**NOTE**: the users will begin to log off when the load generator runs out of data.