# TPC BENCHMARK™ W

# (Web Commerce)

**Specification**

Version 2.0r

Dec 11, 2003

**Transaction Processing Performance Council (TPC)**

PO Box 29920

San Francisco, CA 94129-0920, USA

Phone 415-561-6110

Fax 415-751-4829

http://www.tpc.org

e-mail:admin@tpc.org

## Acknowledgments

Developing a TPC benchmark for a new environment requires a huge effort to conceptualize, research, specify, review, prototype and verify the benchmark.

The TPC-W subcommittee would like to acknowledge the contributions made by the many members during the development of the benchmark specification. It has taken the dedicated efforts of people across many companies, often in addition to their regular duties.

The list of significant contributors to this version includes Steve Barrish, Chris Floyd, Wayne Smith, Steve Morris, Chris Elford, Guy Groulx, Dale Woodford, Alan Chan, Lorna Livingtree, Jerrold Buggert, Brad Lund, Mike Vernal, Cecil Reames, Jim Chen, Mike Molloy, John Benninghoff, Tom Colati, Ram Venkatesh, Matt Hogstrom, Greg Darnell, Priti Mishra, and Shanti Subramanyam.

## TPC Membership

(as of August, 2003)

AMD
BEA Systems, Inc.
Bull S.A.
Compaq Computer Corp
Dell Computers Corp.
Electronic Data Systems Corp.
EMC Corp.
Fujitsu Ltd.
Hewlett-Packard Co.
Hitachi Ltd.
IBM Corp.
Intel Corp.
ITOM International
Microsoft Corp.
NCR Corp.
NEC Systems Laboratory
Netezza
Network Appliance
Oracle Corp.
RackSaver
Silicon Graphics Inc.
Sun Microsystems Inc.
Sybase Inc.
Toshiba Corp.
Unisys Corp.

## Trademarks and Legal Notices

## Document History

| | | |
|---|---|---|
| December 11, 2003 | Draft 2.0r | Revision for December F2F Company Review Vote |
| April 1, 2003 | Draft 2.0a | Revised version 1 to change business model and workload. |

# Table of Contents

# Clause 0 - **Preamble**

## 0.1 Introduction

TPC Benchmark™ W Version 2 (TPC-W Version 2) is a transactional web service benchmark. The workload is performed in a managed environment that simulates the activities of a business oriented transactional application server. The workload exercises commercially available application servers and databases associated with such environments, which are characterized by:

- Multiple on-line business sessions
- Commercially available application environment
- Use of XML documents and SOAP for data exchange
- Business to business application logic
- Dynamic web service response generation with database access and update
- The simultaneous execution of multiple transaction types that span a breadth of complexity
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Transaction integrity (ACID properties)

The performance metric reported by TPC-W is the number of web services processed per second. Multiple web service interactions are used to simulate the business activity of a retail store, and each interaction is subject to a response time constraint. The performance metric for this benchmark is expressed in Web **S**ervice **I**nteractions **P**er **S**econd (SIPS).

All references to TPC-W results must include the primary metrics, which are: the SIPS rate, the associated price per SIPS ($/SIPS), and the availability date of the priced configuration.

Although this specification expresses implementation in terms of a relational data model with a conventional locking scheme, the database may be implemented using any commercially available database management system (DBMS), database server, file system, or other data repository that provides a functionally equivalent implementation. The terms "table", "row", and "column" are used in this document only as examples of logical data structures.

TPC-W uses terminology and metrics that are similar to other benchmarks originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-W results are comparable to other benchmarks. The only benchmark results comparable to TPC-W are other TPC-W results with the appropriate revision.

Despite the fact that this benchmark offers a rich environment that emulates many web service applications, this benchmark does not reflect the entire range of web service or application server requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-W approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, systems design, and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-W should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

Benchmark sponsors are permitted several possible system designs, insofar as they adhere to the model described and pictorially illustrated in clause 6. A Full Disclosure Report of the implementation details, as specified in clause 8, must be made available along with the reported results.

**Comment:** While separated from the main text for readability, comments and the appendices are a part of the standard and are enforced with the following exceptions: Appendix B - ,Appendix D - , Appendix E - ,Appendix F - ...

General Implementation Guidelines

The purpose of TPC benchmarks is to provide relevant, objective performance data to industry users. To achieve that purpose, TPC benchmark specifications require that benchmark tests be implemented with systems, products, technologies and pricing that:

- Are generally available to users.
- Are relevant to the market segment that the individual TPC benchmark models or represents (e.g. TPC-W models and represents high-volume, complex web services and application server environments).
- A significant number of users in the market segment the benchmark models or represents would plausibly implement.

The use of new systems, products, technologies (hardware or software), and pricing is encouraged so long as they meet the requirements above. Specifically prohibited are benchmark systems, products, technologies, pricing, and implementations whose primary purpose is performance optimization of TPC benchmark results without any corresponding applicability to real-world applications and environments. In other words, all "benchmark special" implementations that improve benchmark results but not real-world performance or pricing, are prohibited.

The following characteristics should be used as a guide to judge whether a particular implementation is a benchmark special. It is not required that each point below be met, but that the cumulative weight of the evidence be considered to identify an unacceptable implementation. Absolute certainty or certainty beyond a reasonable doubt is not required to make a judgment on this complex issue. The question that must be answered is this: based on the available evidence, does the clear preponderance (the greater share or weight) of evidence indicate that this implementation is a benchmark special?

The following characteristics should be used to judge whether a particular implementation is a benchmark special:

- Is the implementation generally available, documented, and supported?
- Does the implementation have significant restrictions on its use or applicability that limits its use beyond TPC benchmarks?

- Is the implementation or part of the implementation poorly integrated into the larger product?
- Does the implementation take special advantage of the limited nature of TPC benchmarks (e.g., transaction profile, transaction mix, transaction concurrency and/or contention, transaction isolation) in a manner that would not be generally applicable to the environment the benchmark represents?
- Is the use of the implementation discouraged by the vendor? (This includes failing to promote the implementation in a manner similar to other products and technologies.)
- Does the implementation require uncommon sophistication on the part of the end-user, programmer, or system administrator?
- Is the pricing unusual or non-customary for the vendor or unusual or non-customary to normal business practices? The following pricing practices are suspect:
    - Availability of a discount to a small subset of possible customers.
    - Discounts documented in an unusual or non-customary manner.
    - Discounts that exceed 25% on small quantities and 50% on large quantities.
    - Pricing featured as a close-out or one-time special.
    - Unusual or non-customary restrictions on transferability of product, warranty or maintenance on discounted items.
- Is the implementation being used (including beta) or purchased by end-users in the market area the benchmark represents? How many? Multiple sites? If the implementation is not currently being used by end-users, is there any evidence to indicate that it will be used by a significant number of users?

# Clause 1 - **Web Object and Logical Database Design**

## 1.1 Business and Application Environment

TPC Benchmark™ W comprise a set of basic operations designed to exercise transactional application server functionality in a manner representative of business to business web service environments. These basic operations have been given a real-life context, portraying the business activity of a distributor that supports user online ordering and browsing activity. This is intended to help users relate intuitively to the components of the benchmark. The workload is centered on business logic involved with processing orders and retrieving product catalog items and provides a logical database design.

TPC-W showcases the performance capabilities of a *single* application server. While it is common to have multiple application servers in a typical business environment, for benchmark clarity, TPC-W requires the use of a single application server. The reasoning for this is to clearly demonstrate the performance capabilities of a single system, not that the application can scale outwards with additional application servers. Since the web service interactions are stateless, allowing multiple application servers would create a benchmark where vendors that utilize the most servers would have the leading performance result, not necessarily the vendor with the highest performing application server. In the case of this benchmark, the TPC did not want to confuse clustered versus non-clustered results, and therefore chose the single application server model.

Additionally, TPC-W does not benchmark the logic needed to process or display the presentation layer (for example, HTML) to the clients. The clients in TPC-W represent businesses which utilize web services in order to satisfy their business needs. TPC-W does not represent the activity of any particular business segment, but rather any industry that must market and sell a product or service over the Internet via web services (e.g., retail store, software distribution, airline reservation, etc.). TPC-W does not attempt to be a model of how to build an actual application.

The purpose of this benchmark is to reduce the diversity of operations found in application servers, while retaining the application's essential performance characteristics, namely: the level of system utilization and the complexity of operations. A large number of functions have to be performed to manage an environment which supports order processing and browsing functions. We include a representative set of these functions. Many other functions are not of primary interest for performance analysis, since they are proportionally small in terms of system resource utilization or in terms of frequency of execution. Although these functions are vital for a production system, they merely create unnecessary diversity in the context of a standard benchmark and have been omitted in TPC-W.

The application portrayed by the benchmark is a retail distributor on the internet with ordering and product browse scenarios. The application accepts incoming web service requests from other businesses (or a store front) to place orders, view catalog items and make changes to the catalog, update or add customer information, or request the status of an existing order. The majority of request activity is to generate order purchase activity with a smaller portion of requests browsing the site.

## 1.2 Definitions of Terms

**1.2.1** The term **SYSTEM** (when used in all caps) refers to a dedicated set of hardware (processors, memory, etc) on which executes one instance of an operating system. The set of hardware used by the **SYSTEM** must not change during the test run. For example, a **SYSTEM** could be an entire 4-way computer. A **SYSTEM** could be a static partition of a multiprocessor computer.

**1.2.2** **Managed Environment** – A software abstraction layer that sits between application code and the operating system. It provides a logical runtime environment that insulates the application from the native operating system.

Functions performed by a managed environment include but are not limited to the following:

- Automatic memory management and garbage collection
- Code verification
- Code access security
- Translation of an intermediate language generated by a compiler into native machine code
- Program loading
- Thread creation and scheduling. Threads may be created through API's provided by the runtime libraries that are part of the managed environment.
- Data type checking to ensure that data types are used correctly and consistently.

The execution environment provided must be compliant with one of the following specifications. It may be a superset of the specification and contain vendor specific extensions and enhancements.

- JRE 1.3 or greater as published by Sun Microsystems
- ECMA-335 as published by the European Computer Manufacturers Association

**1.2.3** **Application Server** – A software layer that provides application infrastructure functions and services. It is positioned between the user making a request for information and the underlying business process that satisfies the request. The application server provides core functions and services that include but are not limited to the following:

- Hosts a managed environment for application execution
- Web Services
    - Adheres to WS-I Basic Profile 1.0 Specification
    - Handles HTTP requests and responses
- Network transparency
- Transaction management
- Security
- Multi-threading
- Database connectivity
- Resource pooling
- Application state management

The application server may be an integrated part of the operating system or a separate product that is procured from another source or sources. It is permissible to run more than one application server or multiple instances of a given application server. Application servers and application server components from different sources may be used to perform selected functions within the SUT.

However, the application server(s) must execute on the same SYSTEM. The application server may execute multiple instances of its managed environment. Within each instance of a managed environment, one or more instances of the application program may execute.

**1.2.4** The term **Distributed Transaction Manager** in this specification refers to a software component that allows the coordination of a number of resources that may span several heterogeneous systems on the behalf of an application. It ensures that all operations performed by an application are either completed successfully or leaves no trace of any change. This may involve the synchronization of one or more databases as well as message queues and other resources that may be used by a business transaction. It is able to coordinate the recovery of business transactions in the event of site failure, network failure, or global resource dead locks. The transaction manager must comply with the Open Group's X/Open XA interface, including support of the XA standard for two-phase commit, for communicating with the individual resource managers that may be involved in a business transaction.

**1.2.5** The term **Message Queuing Product** in this specification refers to a software component that provides an API that allows applications and systems to communicate with each other across a heterogeneous network even though some of the systems may be temporarily unavailable. The applications send messages to queues and read messages from queues. The product must have capabilities that include but are not limited to the following:

- Guaranteed message delivery.
- Durable message queues.
- The ability to participate in a transaction with a transaction manager using the X/Open group's XA interface.
- Security capabilities (authentication, authorization, and encryption). The security may be built in to the product. Alternatively, the product may be capable of integrating with another source to provide access to these security technologies.

**1.2.6** The term **Database Server** refers to the software used to implement the data repository that provides a functionally equivalent implementation to the database specification in this benchmark specification. See clause 0.1)

**1.2.7** The terms **external devices** and **appliances** refers to any part of the SUT that is located outside of the main chassis of the SYSTEM.

**1.2.8** The **Test Run** consists of a ramp up followed by a stabilization period followed by the steady state period. Steady state contains the reported Measurement Interval which must be obtained from a valid Test Run. (See clause 5.4 for requirements).

**1.2.9** The term **N unique IDs** is used in this specification to refer to a field that must be able to hold any one ID within a minimum set of N unique IDs, regardless of the physical representation (e.g., binary, packed decimal, alphabetic, etc.) of the field.

**1.2.10** The term **Fixed text, size N** is used in this specification to refer to a field that must be able to hold any string of characters of a fixed length of N. If the string it holds is shorter than N characters, it must be padded with spaces.

**1.2.11** The term **Variable text, size N** is used in this specification to refer to a field that must be able to hold any string of characters of a variable length with a maximum length of N. The field may optionally be implemented as "fixed text, size N".

**1.2.12** The term **random a-string [x..y]** is used in this specification to refer to a string of characters, where the length of the string is generated from a uniform random distribution over the inclusive interval [x,y], and the characters are generated one at a time via a uniform random distribution from the following set:

{a,A,b,B,c,C,d,D,e,E,f,F,g,G,h,H,i,I,j,J,k,K,l,L,m,M,n,N,o,O,p,P,q,Q,r,R,s,S,t,T,u,U,v,V, w,W,x,X,y,Y,z,Z,0,1,2,3,4,5,6,7,8,9}

**1.2.13** The term **random n-string [x..y]** is used in this specification to refer to a string of characters, where the length of the string is generated from a uniform random distribution over the inclusive interval [x,y], and the characters are generated one at a time via a uniform random distribution from the following set:

{0,1,2,3,4,5,6,7,8,9}

**1.2.14** The term **DigSyl** is used in this specification to refer to the following function:

DigSyl(D) where:

D is a positive integer.

DigSyl(D) returns a string of length N where N is twice the number of digits in the decimal representation of D. This string is the concatenation of 2-character syllables constructed by replacing each digit in the decimal representation of D with the corresponding 2-character syllable from the following table:

| **Digit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Syllable** | BA | OG | AL | RI | RE | SE | AT | UL | IN | NG |

**Examples**: DigSyl(15) returns the string "OGSE", DigSyl(345) returns the string "RIRESE" and DigSyl(10003) returns the string "OGBABABARI".

**1.2.15** The term **Date** is used in this specification to refer to a field that must be able to hold any date between 1st January 1800 and 31st December 2100 with a resolution of at least one day. For SOAP message requests and responses this format must comply with ISO 8601 (see http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime).

**1.2.16** The term **Date and time** is used in this specification to refer to a field that must be able to hold any date between 1st January 1800 and 31st December 2100 with a resolution less than or equal to one second. For SOAP message requests and responses this format must comply with ISO 8601 (see http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime).

**1.2.17** The term **Current Date** is used in this specification to refer to a date and time stamp as returned by the operating system. For SOAP message requests and responses this format must comply with ISO 8601 (see http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime).

**1.2.18** The term **Numeric** is used in this specification to refer to a field that must be able to hold any numeric value. A field specified as **Numeric, N digits** must be able to hold any N decimal digits value. A field specified as **Numeric, (n,m) digits** must be able to hold any n decimal digits value before the decimal point and any m decimal digits value after the decimal point. Numeric fields that contain monetary values (Numeric, (n,m) digits) must use data types that give exact representation to at least the smallest monetary unit in the currency being used. For example, O_TOTAL in U.S. dollars may be represented as (12,2) digit signed decimal (with implicit scaling), or scaled to cents in a signed integer of at least 41 bits, or scaled to cents in a double precision (64 bit) REAL.

**1.2.19** The term **Null** is used in this specification to mean out of the range of valid values for a given datatype and always the same value.

**1.2.20** The term **application program** is used in this specification to refer to code that is not part of the commercially available components of the system, but produced specifically to implement the web interactions and the database transactions defined in this benchmark. For example, stored procedures, triggers, and referential integrity constraints are considered part of the application program when used to implement any portion of the web interactions or database transactions, but are not considered part of the application program when solely used to enforce integrity rules (see clause 1.6) or transparency requirements (see clause 1.7) independently of any specific web service or database transaction.

**1.2.21** The term **Measurement Interval** is used in this specification to refer to a continuous subset of the steady state period with a length of at least 2 hours for which the test sponsor is reporting a performance metric. See clause 5.4.2 for detailed requirements.

**1.2.22** The term **Ramp Up** is used in this specification to refer to the time between the submission of the first web service interaction and the time when every EB has submitted at least one web service interaction request.

**1.2.23** The term **Stabilization Period** is used in this specification to refer to the time between the end of the ramp up period and the beginning of the steady state period. The beginning of the steady state period is to be determined by the auditor.

**1.2.24** The term **Steady State Period** is used in this specification to refer to a time following the stabilization period in which the throughput level represents the true sustainable performance of the SUT. See clause 5.4.1.

**1.2.25** The term **Web Service Interaction Response Time** (SIRT) is used in this specification to refer to the time taken to perform a successful web interaction. (See clause 5.2.1 for detailed requirements.)

**1.2.26** The term **SIPS** is used in this specification to refer to the average number of Service Interactions Per Second completed during the Measurement Interval.

**1.2.27** The term **$/SIPS** is used in this specification to refer to the total cost of the SUT (see Clause 7) divided by the number of SIPS measured during the Measurement Interval.

**1.2.28** The **System Under Test (SUT)** comprises all components which are part of the "application" being simulated. This includes, but is not limited to, network connections, the application server and database server. (See clause 6.4)

**1.2.29** The term **client** is used in this specification to refer to an entity (usually a software program; could be a human) that communicates with the SUT via a web service request.

**1.2.30** The term **Emulated Business** (**EB**) is used in this specification to refer to the entity (e.g., a process or a thread) that emulates a client communicating via web services by sending and receiving SOAP messages via HTTP and TCP/IP over a network connection (e.g., a socket) to the SUT.

**1.2.31** The **Remote Business Emulator (RBE)** is the software component that drives the TPC-W workload. It emulates Business using a computer to request services from the System Under Test (SUT). (See clause 6.1)

**1.2.32** The term **web service request** in this specification refers to the communication of all input requirements for a given interaction from the EB to the SUT in a valid SOAP request message. The format of this request must adhere to the WS-I Basic Profile 1.0 specification (see http://www.ws-i.org for the Basic Profile 1.0 specification).

**1.2.33** The term **web service response** in this specification refers to the communication of a SOAP response message from the SUT to the EB. The format of this response must adhere to the WS-I Basic Profile 1.0 specification (see http://www.ws-i.org for the Basic Profile 1.0 specification).

**1.2.34** The term **web service interaction** is used in this specification to refer to a sequence of SOAP messages exchanged between an EB and the SUT such that the first SOAP message in the sequence is a web service request of a given interaction type, sent by the EB to the SUT, and the final SOAP message in the sequence is a web service response. Between the first and final SOAP messages other messages may be sent between the EB and the SUT :

- Web service responses comprising SOAP fault messages from the SUT to the EB.
- Resubmissions of the web service request from the EB to the SUT.
- Non-SOAP HTTP messages, or TCP/IP messages required as part of error recovery actions taken by the SUT or the EB.

**1.2.35** The term **successful web service interaction** is used in this specification to refer to a web service interaction in which the final SOAP message in the sequence is a web service response that meets the Response Definition requirements for the interaction type, and the SUT has performed all the business logic specified in the Processing Definition for the interaction type, and in which no more than 20 resubmissions of the web service request has occurred (see clause 5.4.1.2).

**1.2.36** The term **unsuccessful web service interaction** is used in this specification to refer to a web service interaction that is not a successful web service interaction.

**1.2.37** The term **Business Session** is used in this specification to refer to a continuous period of time during which an EB requests one or more web interactions, starting with the first requested interaction and ending as specified in clause 6.2.4.

**1.2.38** The term **Business Session Length**, BSL, refers to the number of Web Service Interactions that are performed by the EB during a Business Session. (See clause 6.2.)

**1.2.39** The term **Web Logging** is used in this specification to refer to entries in the Web Server Access Log.

**1.2.40** The term **database transaction** as used in this specification refers to one or more operations that result in a unit of work on the database with full ACID properties as described in Clause 3. A web service interaction may be comprised of one or more database transactions. When a database transaction includes more than one operation, the set of operations is enclosed between the tags <Start Transaction> and <End Transaction>.

The order of the data manipulations within the transaction bounds is immaterial, unless otherwise specified, and is left to the latitude of the test sponsor, as long as the implemented transactions are functionally equivalent to those specified in the transaction profiles.

**1.2.41** For the purposes of this benchmark an **atomic operation** is a set of actions where all actions described within the delimiters <start atomic operation>, <end atomic operation> must complete successfully or be fully reversed. If any action within the atomic operation fails, all actions within the atomic operation must be reversed so that it is functionally equivalent (from a business logic perspective) to having never occurred.

This does not include logs.

For the purposes of this benchmark all atomic operations must meet the ACID properties defined in Clause 3.

**1.2.42** The term **obtained** is used in this specification to refer to the action of retrieving the current value of a given field from within the SUT. The location within the SUT from which the value is retrieved, such as database, cache, or other, is not specified and is only constrained by the web interaction definitions and by the ACID requirements.

**1.2.43** The notation <DBMS> and </DBMS> delimit the portions of the processing that MAY utilize the instance of a commercially available DBMS containing the TPC-W primary tables. Processing that occurs within these delimiters must meet ACID requirements for the operations defined within. Any processing outside these delimiters MUST NOT utilize this DBMS instance and MUST be performed on the application server SYSTEM.

**1.2.44** The term **randomly selected within [x.. y]** means independently selected at random and uniformly distributed between x and y, inclusively, with a mean of (x+y)/2, and with the same number of digits of precision as shown. For example, [0.01 .. 100.00] has 10,000 unique values, whereas [1 ..100] has only 100 unique values.

**1.2.45** The term **non-uniform random function (NURand)** is used in this specification to refer to the function used for generating C_ID. This function generates an independently selected and non-uniformly distributed random number over the specified range of values [*x* .. *y*], and is specified as follows:

NURand(A, x, y) = ((random(0, A) | random(x, y)) % (y - x + 1)) + x

Where:

- expr1 | expr2 stands for the bitwise logical OR operation between expr1 and expr2
- expr1 % expr2 stands for expr1 modulo expr2
- random(x, y) stands for randomly selected within [*x* .. *y*]
- A is a constant chosen according to the size of the range [x .. y]

**1.2.46** The term **ICE** refers to the Inventory Control Emulator. This service sits outside the SUT. It emulates an external vendor that the business represented by the SUT would contact to order more stock.

**1.2.47** The term **PGE** refers to the Payment Gateway Emulator. This service sits outside the SUT. It emulates an external vendor that the business represented by the SUT would contact to verify the information.

**1.2.48** The term **POV** refers to the Purchase Order Verification. This service sits outside the SUT. It emulates an external vendor that the business represented by the SUT would contact to authorize the use of a purchase order as an acceptable method of payment.

**1.2.49** The term **MAX_ORDERS** is defined to be 10.

**1.2.50** The term **ITEM_LIMIT** is defined to be 50.

**1.2.51** The term **MaxProductDetailID*s*** is defined to be 10.

**1.2.52** The term **NURandAProductDetail** is defined to be 1023.

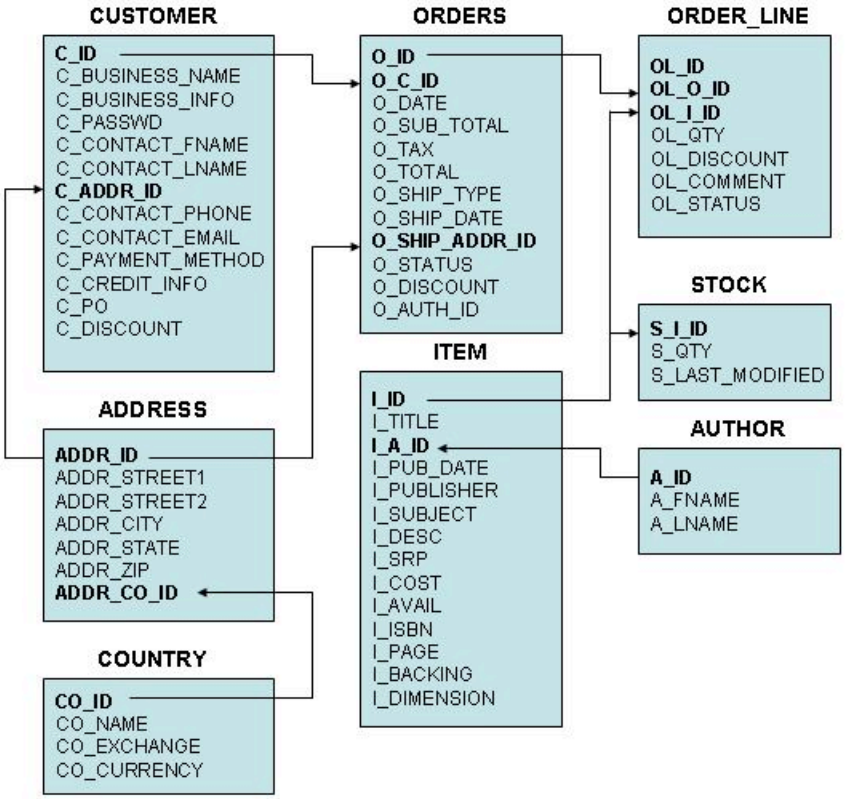
**1.2.53** The term **MaxCreateOrderIDs** is defined to be 10.

**1.2.54** The term **MaxChangeItemIDs** is defined to be 5.

## 1.3 Database Entities, Relationships, and Characteristics

**1.3.1** The components of the TPC-W database are defined to consist of a minimum of eight separate and individual base tables. The relationships among these tables are defined in the entity-relationship diagram shown below and are subject to the rules specified in clause 1.5.

**Comment:** To enable commercial products (commerce or merchant applications) to execute the workload without extensive modifications, a superset of the database schema is allowed. This could be in the form of additional tables. All such additions and/or modifications must be fully disclosed.




**Legend:**

Dotted lines represent one-to-one relationships between non-key fields related through a business rule. These fields are shown in italic.

The arrows point in the direction of one-to-many relationships between tables.

Bold types identify primary and foreign keys.

## 1.4 Table Layouts

The following lists define the structure (list of fields) of each table. For each table, the defined fields can be implemented in any order, using any physical representation available from the tested system.

**Comment**: Table and column names are used for illustration purposes only; different names may be used by the implementation.

### 1.4.1 ITEM Table **Layout**

| Field Name | Field Definition | Comments |
|---|---|---|
| I_ID | Numeric, 9 digits | Unique ID of Item |
| I_TITLE | Variable text, size 60 | Title of Item |
| I_A_ID | Numeric, 9 digits | Author ID of Item |
| I_PUB_DATE | Date | Date of release of the product |
| I_PUBLISHER | Variable text, size 60 | Publisher of item |
| I_SUBJECT | Variable text, size 60 | Subject of Book |
| I_DESC | Variable text, size 500 | Description of Item |
| I_SRP | Numeric, (15,2) digits | Suggested Retail Price |
| I_COST | Numeric, (15,2) digits | Cost of Item |
| I_AVAIL | Date | When item is available |
| I_ISBN | Fixed text, size 13 | Product ISBN |
| I_PAGE | Numeric, 4 digits | Number of pages of book |
| I_BACKING | Variable text, size 15 | Type of book, paper or hard back |
| I_DIMENSIONS | Variable text, size 25 | Size of book in inches |
| Primary Key: (I_ID)<br>(I_A_ID) Foreign Key, references (A_ID) | | |

### 1.4.2 STOCK Table **Layout**

| Field Name | Field Definition | Comments |
|---|---|---|
| S_I_ID | Numeric, 9 digits | Unique ID of Item |
| S_QTY | Numeric, 9 digits | Quantity in stock |
| S_LAST_MODIFIED | Date and time | Time of last received shipment |
| Primary Key: (S_I_ID)<br>(S_I_ID) Foreign Key, references (I_ID) | | |

### 1.4.3 AUTHOR Table **Layout**

| Field Name | Field Definition | Comments |
|---|---|---|
| A_ID | Numeric, 9 digits | Unique Author ID |
| A_FNAME | Variable text, size 20 | First Name of Author |
| A_LNAME | Variable text, size 20 | Last Name of Author |
| Primary Key: (A_ID) | | |

### 1.4.4 CUSTOMER Table Layout

| Field Name | Field Definition | Comments |
|---|---|---|
| C_ID | Numeric, 9 digits | Unique ID per Customer |
| C_BUSINESS_NAME | Variable text, size 20 | Unique Business name |
| C_BUSINESS_INFO | Variable text, size 100 | Miscellaneous information |
| C_PASSWD | Variable text, size 20 | User Password for Business |
| C_CONTACT_FNAME | Variable text, size 15 | First name of Business Contact |
| C_CONTACT_LNAME | Variable text, size 15 | Last name of Business Contact |
| C_ADDR_ID | Numeric, 9 digits | Billing Address ID for this customer |
| C_CONTACT_PHONE | Variable text, size 16 | Phone number of Business Contact |
| C_CONTACT_EMAIL | Variable text, size 50 | Email of Business Contact |
| C_PAYMENT_METHOD | Variable text, size 2 | Payment method text code |
| C_CREDIT_INFO | Variable text, size 300 | Credit Information |
| C_PO | Numeric, 9 digits | Purchase order number |
| C_DISCOUNT (verify if set in New Customer) | Numeric, (3,2) digits | Percentage discount for Customer |
| Primary Key: (C_ID)<br>(C_ADDDR_ID) Foreign Key, references (ADDR_ID) | | |

### 1.4.5 ORDERS **Table** Layout

| Field Name | Field Definition | Comments |
|---|---|---|
| O_ID | Numeric, 18 digits | Unique ID per order |
| O_C_ID | Numeric, 9 digits | Customer ID of Order |
| O_DATE | Date and time | Order Date and time |
| O_SUB_TOTAL | Numeric, (15,2) digits | Subtotal of all order-line items |
| O_TAX | Numeric, (15,2) digits | Tax over the subtotal |
| O_TOTAL | Numeric, (15,2) digits | Total for this order |
| O_SHIP_DATE | Date and time | Date the orders was shipped to customer |
| O_SHIP_TYPE | Variable text, size 10 | Method of delivery |
| O_SHIP_ADDR_ID | Numeric, 9 digits | Address ID to ship order |
| O_STATUS | Variable text, size 16 | Order status |
| O_AUTH_ID | Variable text, size 16 | Authorization code from PGE (for credit orders) |
| O_SHIP_COST | Numeric, (15,2) digits | Shipping costs for this order |
| O_DISCOUNT | Numeric, (3,2) digits | Percentage discount off (note: Need to update Create Order to use this field instead of OL_DISCOUNT) |

Primary Key: (O_ID)
(O_C_ID) Foreign Key, references (C_ID); (O_SHIP_ADDR) Foreign Key, references (ADDR_ID)

### 1.4.6 ORDER_LINE Table **Layout**

| Field Name | Field Definition | Comments |
|---|---|---|
| OL_ID | Numeric, 3 digits | Unique Order Line Item ID |
| OL_O_ID | Numeric, 18 digits | Order ID of Order Line |
| OL_I_ID | Numeric, 9 digits | Unique Item ID (I_ID) |
| OL_QTY | Numeric, 9 digits | Quantity of Item |
| OL_STATUS | Variable text, size 16 | Order Status for this Item |
| OL_COST | Numeric, (15,2) digits | Cost for this line |

Primary Key: (OL_ID, OL_O_ID)
(OL_I_ID) Foreign Key, references (I_ID); (OL_O_ID) Foreign Key, references (O_ID)

### 1.4.7 ADDRESS Table **Layout**

| Field Name | Field Definition | Comments |
|---|---|---|
| ADDR_ID | Numeric, 9 digits | Unique address ID |
| ADDR_STREET1 | Variable text, size 40 | Street address, line 1 |
| ADDR_STREET2 | Variable text, size 40 | Street address, line 2 |
| ADDR_CITY | Variable text, size 30 | Name of city |
| ADDR_STATE | Variable text, size 20 | Name of state |
| ADDR_ZIP | Variable text, size 10 | Zip code or Postal code |
| ADDR_CO_ID | Numeric, 4 digits | Unique ID of Country |
| Primary Key: (ADDR_ID)<br>(ADDR_CO_ID) Foreign Key, references (CO_ID) | | |

**1.4.8** COUNTRY **Table** Layout

| Field Name | Field Definition | Comments |
|---|---|---|
| CO_ID | Numeric, 4 digits | Unique Country ID |
| CO_NAME | Variable text, size 50 | Name of Country |
| Primary Key: (CO_ID) | | |

## 1.5 Implementation Rules

**1.5.1** The physical clustering of records within the database is allowed.

**1.5.2** All tables must have the properly scaled number of rows as defined by the database population requirements (see clause 4.3).

**1.5.3** Horizontal partitioning of tables is allowed. Groups of rows from a table may be assigned to different files, disks, or areas. If implemented, the details of such partitioning must be disclosed.

**1.5.4** Vertical partitioning of tables is allowed. Groups of fields (columns) of one table may be assigned to files, disks, or areas different from those storing the other fields of that table. If implemented, the details of such partitioning must be disclosed (see clause 1.7 for limitations).

**1.5.5** Replication is allowed for all tables. Manipulation of data in all copies of tables which are replicated must meet all requirements for atomicity, consistency, and isolation as defined in Clause 3. If implemented, the details of such replication must be disclosed.

**Comment**: Only one copy of a replicated table needs to meet the durability requirements defined in Clause 3.

**1.5.6** Fields may not be added or duplicated from one table defined in clause 1.4 into any of the other tables defined in clause 1.4. This does not preclude the use of views, temporary tables or permanent tables which do combine fields from multiple tables. However these are done, they must meet the ACI requirements. The application must maintain the original tables as specified in clause 1.4.

**1.5.7** Each field, as described in clause 1.4, must be logically discrete and independently accessible by the data manager. For example, A_FNAME and A_LNAME cannot be implemented as two sub-parts of a discrete attribute A_NAME.

**1.5.8** Each field, as described in clause 1.4, must be accessible by the data manager as a single field. For example, C_BUSINESS_INFO cannot be implemented as two discrete fields C_BUSINESS_INFO_1 and C_BUSINESS_INFO_2.

**Comment:** The following fields are exceptions to this clause: All fields holding a time-and-date value (i.e., O_DATE) can be implemented as a combination of two fields: a date field and a time field. No vertical partitioning can be defined between the two fields used to implement it.

**1.5.9** No field(s) may represent the physical disk addresses of the row or any offsets thereof. The application may not reference rows using relative addressing since they are simply offsets from the beginning of the storage space. This does not preclude hashing schemes or other file organizations which have provisions for adding, deleting, and modifying records in the ordinary course of processing.

**Comment 1**: It is the intent of this clause that the application program (see clause 1.2.20) executing the database transaction, or submitting the database transaction request, not use physical identifiers, but logical identifiers for all accesses, and contain no user written code which translates or aids in the translation of a logical key to the location within the table of the associated row or rows. For example, it is not legitimate for the application to build a "translation table" of logical-to-physical addresses and use it to enhance performance.

**Comment 2**: Internal record or row identifiers, for example, tuple IDs or cursors, may be used under the condition that within each database transaction, initial access to any row must be via a logical key comprised only of fields from that row. Initial access includes insertion, deletion, retrieval, and update of any row.

**1.5.10** While inserts and deletes are not performed on all tables, the system must not be configured to take special advantage of this fact during the test. Although inserts are inherently limited by the storage space available on the configured system, there must be no restriction on inserting in any of the tables a minimum number of rows equal to 5% of the table cardinality and with a key value of at least double the range of key values present in that table.

**Comment:** It is required that the space for the additional 5% table cardinality be configured for the Test Run (see clause 5.5) and priced accordingly. If a commercial product is used for the application which requires a superset of the database schema, then the space configured and priced should include the additional storage needed for the additional tables and/or fields. For systems where space is configured and dynamically allocated at a later time, this space must be considered as allocated and included in the priced system (see clause 4.4).

**1.5.11** The minimum decimal precision for any computation performed as part of the application program (see clause 1.2.20) must be the maximum decimal precision of all the individual items in that calculation. The application code must handle the entire range of values as defined in clause 1.4.

## 1.6 Integrity Rules

In any committed state, the primary key values must be unique within each table. For example, in the case of a horizontally partitioned table, primary key values of rows across all partitions must be unique.

## 1.7 Data Access Transparency Requirements

Data access transparency is the property of the system which removes from the application program any knowledge of the location and access mechanisms of partitioned data. An implementation which uses vertical and/or horizontal partitioning must meet the requirements for transparent data access described here.

No finite series of tests can prove that the system supports complete data access transparency. The requirements below describe the minimum capabilities needed to establish that the system provides transparent data access.

**Comment**: The intent of this clause is to require that access to physically and/or logically partitioned data be provided directly and transparently by services implemented by commercially available layers below the application program such as the data/file manager (DBMS), the operating system, the hardware, or any combination of these.

**1.7.1** All tables used by the application must be identified by names which have no relationship to the partitioning of tables. All data manipulation operations in the application program (see clause 1.2.20) must use only these names.

**1.7.2** The system must prevent any data manipulation operation performed using the names described in clause 1.7.1 which would result in a violation of the integrity rules (see clause 1.6).

**1.7.3** Using the names which satisfy clause 1.7.1, any arbitrary non-TPC-W application must be able to manipulate any set of rows or columns:

- Identifiable by any arbitrary condition supported by the underlying DBMS
- Using the names described in clause 1.7.1 and using the same data manipulation semantics and syntax for all tables.

For example, the semantics and syntax used to update an arbitrary set of rows in any one table must also be usable when updating another arbitrary set of rows in any other table.

**Comment**: The intent is that the TPC-W application program uses general purpose mechanisms to manipulate data in the database.

# Clause 2 - **Web Service Interactions and Workload Profile**

## 2.1 Implementation Rules

**2.1.1** The use of commercially available products is mandated. The following functions, if used by the SUT, must be provided by commercially available products and there can be no user written code that directly references these functions during the measurement interval:

- Operating system
- Device drivers
- Network
- Multiplexing
  - Routing
  - Load Balancing
- Caching (see clause 6.4.3)

**2.1.2** The following functions, if used by the SUT, must be provided by commercially available products:

- Distributed transaction management (see clause 1.2.4)
- Database (see clause 1.2.6)
  - Client library used to access database
- Application server (see clause 1.2.3)
- Managed environment (see clause 1.2.2)
  - "Post processing" byte code optimizers
- Serialization
  - Converting application objects to XML and back again (see clause 2.1.7)
  - Converting SOAP messages into XML and back again (see clause 2.1.6)
- Messaging
  - Sending and receiving of SOAP messages
  - Web server and web client functionality (see clause 2.1.10)
  - SSL/TLS encryption/decryption/handshake (see clause 2.1.12)
  - Queue management (see clause 1.2.5)
  - Protocols

If these products are used, code, APIs and configuration files may be used to manage the operation of the commercial products. If used, the code, APIs and configuration files must use general purpose facilities (i.e., not TPC-W benchmark specific).

User code cannot implement any of the functions in the list above.

**2.1.3** All web service interactions must be secure. In secure web interactions all communications between the EB and the SUT must be encrypted and communicated with SSL version 3, or with TLS (RFC2246), using SSL_RSA_WITH_RC4_128_MD5 as the cipher suite. The private key for the server digital certificate must be at least 1024 bits. A description of the cipher suite can be found at:

http://wp.netscape.com/eng/ssl3/

**2.1.4** In each Business Session that executes a secure interaction, the EB associated with the Business Session must establish a new SSL/TLS session by executing a full handshake. SSL/TLS sessions must not be shared by multiple Business Sessions.

**2.1.5** The SUT must parse all SOAP messages for the incoming web service requests. The parser must be a commercially available component of the SUT.

**2.1.6** The SOAP encoding for the web service response must be handled by a commercially available component of the SUT.

**2.1.7** A commercially available product must be used to serialize and de-serialize all XML messages that are a part of the processing that takes place on the SUT. This consists of the following:

- XML documents exchanged between the SUT and the RBE
- XML documents exchanged between the SUT and the external emulators (PGE, POV and ICE).
- XML documents that are exchanged between the web service interactions and the background processes (shipping and stock management). These documents are sent from one logical entity to another via the messages queues that are defined as part of the benchmark.

**Comment:** The intent is to avoid implementations including XML serialization and/or de-serialization for these functions using user written or modified code. Communication methods between the application server and DBMS server remain unconstrained.

**2.1.8** The SUT must manage all network connections between the SUT and external network endpoints by using commercially available components in the SUT.

**2.1.9** User written code which performs queuing operations to the Shipping and Stock Management processes is not allowed.

**Comment 1:** This does not preclude the use of commercially available API's that perform queuing operations. For instance, an API call which places a message on the Stock Management queue is permitted, however, user written code that manages and handles the queue operations is not permitted.

**2.1.10** The SUT must manage all HTTP communications between the SUT and external network endpoints by using commercially available code on the SUT that provides a framework and/or API's for managing HTTP functionality. This functionality includes, but is not limited to, listening for incoming HTTP connections, accepting and maintaining open sockets for these connections, receiving and parsing HTTP headers, sending response payloads and logging incoming HTTP requests in CLF format (see clause 2.1.13). If necessary, user code may effect the creation and management of HTTP sessions and connections by using these API exposed by the commercially available HTTP layer directly.

In addition, opening outgoing HTTP connections, sending HTTP requests and payloads (e.g., send()) and receiving responses from external providers (e.g., PGE , etc.) must be performed by using commercially available code on the SUT that provides APIs for managing HTTP functionality. User code may effect the creation and management of HTTP sessions and connections by using these API's directly.

Comment 1: The intent of this clause is to not explicitly require a separate, commercially available, web server that hosts the web services. This clause allows hosted applications to use APIs that hook to commercial functionality that provides needed HTTP functionality.

Comment 2: The intent of this clause is to ensure that application code does not explicitly open and utilize socket calls for incoming or outgoing HTTP requests. This low level functionality must be provided by a commercially supported component of the SUT removing this functionality from the application program.

**2.1.11** The web service application must return an error code if an error is encountered. An HTTP status code in the 4xx-5xx range must be returned to the EB if processing errors occurred.

**2.1.12** The SUT must manage all SSL/TLS communications between the SUT and external network endpoints by using commercially available code on the SUT that provides API's for managing SSL/TLS functionality. This functionality includes, but is not limited to, SSL/TLS session establishment and resumption (handshaking), and encryption and decryption operations. User code may effect the creation and management of SSL/TLS sessions and SSL/TLS connections by using these API's directly.

**2.1.13** The clauses titled Input Requirements define the set of data required by the SUT as input to the web services. The underlying format for the WSDL documents are not mandated by this specification. Sample WSDLs are provided in Appendix E - ..

Comment: This clause is not intended to allow an implementation to circumvent parsing overheads. The intent of this clause is to allow Web Services IDEs to have flexibility in defining their own WSDL rather than mandating a specific WSDL.

**2.1.14** The clauses titled Processing Definition define the business logic that must be executed by the SUT as part of the interaction.

**2.1.15** The clauses titled Response Definition include a set of requirements for the SOAP response message produced by the SUT for that web service. Sample WSDL for the SOAP responses are found in Appendix E - ..

**2.1.16** If caching is used, it must meet all the requirements of clause 6.4.3.

**2.1.17** The Web Server Access Log data must be collected with a minimum resolution of one second and written in Common Log Format (CLF) at least once every 30 seconds to a persistent media see Appendix B.2. The fields of CLF are as follows:

- client: DNS Name or IP address of remote client
- dd: Day
- mm: Month
- yyyy: Year
- hh: hour
- mm: minutes
- ss: seconds
- request: First line of HTTP request as sent by the client
- ddd: status code returned by the server

- bbbb: Total number of bytes sent, not including HTTP header

**Comment:** The logging requirements defined above are required only for systems that run application code. Primary store is defined as the place the data is stored assuming all caches are empty. This does not mean the disk subsystems or RAID controllers, but the system that manages these components.

**2.1.18** The Change Item, Shipping, and Stock Management interactions are administrative tasks. They are not components that a regular external business would see, but exist as part of the model depicted by the workload. Shipping and Stock Management do not require a web service interface, since they are executing as background processes.

**2.1.19** Each EB must use at least one independent random number generator (i.e., not shared with any other EB). See clause 6.3 for random number generator requirements. Each random number generator must be seeded with a unique value. Although the selection of the unique seeds is left to the implementer, it must not be done in a fashion that would improve performance. The start seeds must be selected in such a manner that they are not duplicates from previous test runs.

**Comment:** The intent is to avoid EBs following identical processing steps from previous runs that would result in any performance advantage (for example, resulting in a reduction of inserts to the ADDRESS table). While it is unreasonable to prove that the chosen start seeds were not used in any previous run, the auditor should verify that the seeding algorithm generates unique seeds for each run, and does not generate a similar set of seeds from the previous runs.

**2.1.20** The SUT is required to generate random numbers for certain Web Interactions. If these random numbers are generated by multiple random number generators for a particular Web Interaction, the initial random number generator seeds must be unique. See clause 6.3 for random number generator requirements.

**Comment**: The intent of this Clause is to prevent a performance advantage that could result from all the EB's or SUT systems using random numbers generated from a set of non-unique seeds which may generate non-random accesses and requests to the web services and database objects.

**2.1.21** Data within an interaction is only required to be obtained once, even though it may be referred to multiple times within that interaction.

## 2.2 New Customer Web Service

**2.2.1** Overview

This web service interaction accepts a web service request for a new customer registration and returns the new customer ID (C_ID) and a new purchase order (C_PO) to the EB.

**2.2.2** Input Requirements

This web service interaction is invoked by a web service request and requires the following input data:

- BUSINESS_NAME is a random a-string [15..20]
- PASSWORD generated as BUSINESS_NAME in lowercase
- CONTACT F_NAME is a random a-string [8..15]
- CONTACT L_NAME is a random a-string [8..15]
- PHONE is a random n-string [9..16]
- EMAIL see clause 4.6.4.
- BUSINESS_INFO is a random a-string[40..100]
- PAYMENT_METHOD is chosen as either "PO" or "CC", where "CC" is chosen as defined in clause 4.6.9
- CREDIT_INFO is a random a-string [40..300]
- BILLING_ADDR1 is a random a-string [15..40]
- BILLING_ADDR2 is a random a-string [15..40]
- BILLING_CITY is a random a-string [4..30]
- BILLING_STATE is a random a-string [2..20]
- BILLING_ZIP is a random a-string [5..10]
- BILLING_COUNTRY is chosen according to clause 4.6.5

**2.2.3** Processing Definition

**2.2.3.1** If PAYMENT_METHOD is "PO", the SUT initiates a web service request to the POV validating the CREDIT_INFO. See clause 6.6 for details on PGE requests.

This is an XML/SOAP request containing the following fields:

- BUSINESS_NAME
- CREDIT_INFO

2.2.3.2 <DBMS>

<Start atomic operation>

The SUT attempts to match the customer's billing address (BILLING_STREET1, BILLING_STREET2, BILLING_CITY, BILLING_STATE, BILLING_ZIP, BILLING_COUNTRY) with an address in the ADDRESS table. If a match is found, then the C_ADDR_ID is set to ADDR_ID for the matching record. If no match is found, a new record is created in the ADDRESS table using the customer's address with a new unique ADDR_ID (not necessarily sequential nor contiguous).

> **Comment:** The ADDR_CO_ID is obtained by comparing the BILLING_COUNTRY to the CO_NAME value in the COUNTRY table.

2.2.3.3 Insert the following information into the CUSTOMER table.

- C_ID = The SUT will generate a unique C_ID for the new customer
- C_BUSINESS_NAME = BUSINESS_NAME
- C_BUSINESS_INFO = BUSINESS_INFO
- C_PASSWORD = PASSWORD
- C_CONTACT_FNAME = CONTACT_FNAME
- C_CONTACT_LNAME = CONTACT_LNAME
- C_ADDR_ID
- C_CONTACT_PHONE = PHONE

- C_CONTACT_EMAIL = EMAIL
- C_PAYMENT_METHOD = PAYMENT_METHOD
- C_CREDIT_INFO = CREDIT_INFO
- C_DISCOUNT generated as uniform [0.00 .. 0.50]

<End atomic operation>
- C_PO = C_ID

</DBMS>

**2.2.4** Output Requirements

The SUT returns the new C_ID and C_PO to the client in a web service response. See Appendix E - for a sample SOAP WSDL.

## 2.3 Change Payment Method

**2.3.1** Overview

This web service interaction allows the customer to request a change of their payment.

**Comment 1:** For ease of benchmarking purposes, the EB will randomly choose the new payment method. The EB will choose 'PO' 50% of the time, and 'CC' 50% of the time. There is no requirement for the EB to have knowledge of the current customer payment method.

**2.3.2** Input Requirements

This web service interaction is invoked by a web service request and requires the following input data:

- Customer ID (C_ID) where C_ID has been determined by clause 6.2.3
- PAYMENT_METHOD is either 'PO' or 'CC'. 'PO' is chosen 50% of the time.
- CREDIT_INFO is a random a-string [40..300]

**2.3.3** Processing Requirements

**2.3.3.1** If PAYMENT_METHOD is 'PO' :

**2.3.3.1.1** <DBMS> Retrieve the C_BUSINESS_NAME for the given C_ID. </DBMS>

**2.3.3.1.2** The SUT initiates a POV request using CREDIT_INFO as input to the POV. See clause 6.5 for details on POV processing requirements. This is an XML/SOAP request with the following fields:

- C_BUSINESS_NAME

- CREDIT_INFO

The POV will return a unique AUTH_ID upon successful processing of the POV web service request. Further processing may proceed only upon receipt of this response.

**2.3.3.2** <Start atomic operation>
<DBMS>
The SUT updates the following information for the given C_ID within a single database transaction:
- C_PAYMENT_METHOD = PAYMENT_METHOD
- C_CREDIT_INFO = CREDIT_INFO

<End atomic operation> </DBMS>

**2.3.4** Output Requirements

The SUT returns the new C_PAYMENT_METHOD to the client in a web service response. A sample SOAP WSDL can be found in Appendix E - .

## 2.4 Create Order

**2.4.1** Overview

This web service creates an order on the database and then notifies an order fulfillment system to ship the order.

**2.4.2** Input Requirements

This web service interaction is invoked by a web service request and requires the following input data:

- Customer ID (C_ID) [1..NUM_CUSTOMERS] selected as defined in clause 6.2.3.
- A list of item number and quantity pairs for each item to be purchased (I_ID, QTY). The values for I_ID are selected as defined in clause 2.4.2.1. The length of the I_ID/QTY list is determined as defined in clause 2.4.2.8.
- Street address (SHIPPING_STREET1) is a random a-string [15..40]
- Street address (SHIPPING_STREET2) is a random a-string [15..40]
- City (SHIPPING_CITY) is a random a-string [4..30]
- State (SHIPPING_STATE) is a random a-string [2..20]
- Zip code (SHIPPING_ZIP) is a random a-string [5..10]
- Country Name is selected as defined in clause 2.4.2.3
- Shipping Type is selected as defined in clause 2.4.2.4
- The PO number is set to C_ID from above.
- Credit card type (CC_TYPE) is selected as defined in clause 2.4.2.5
- Credit card number (CC_NUMBER) is a random n-string[16..16]
- Credit card expiration date (CC_EXPIRY) in ISO8601 format to adhere to XML standards is selected as defined in clause 2.4.2.6
- Name on the credit card (CC_NAME) is random a-string [10…30].

**2.4.2.1** Selection of input item id (I_ID)

Selected from a uniform random distribution [1..NUM_ITEMS]

**2.4.2.2** Selection of input item quantities

The quantity of the items must be selected to ensure that a minimum of 10% of the orders created must have one "back ordered" item. The initial quantities of S_QTY is set in clause 4.7 . Back orders are generated by using S_QTY initial value plus one.

**2.4.2.3** Selection of input country

Selected with a random uniform distribution from the table specified in clause 4.6.5

**2.4.2.4** Selection of input shipping type

Selected with a random uniform distribution from the following list:

- AIR
- UPS
- FEDEX
- SHIP
- COURIER
- MAIL

**2.4.2.5** Selection of input credit card type

Selected with a random uniform distribution from the following list:

- VISA
- MASTERCARD
- DISCOVER
- AMERICAN EXPRESS
- DINERS

**2.4.2.6** Selection of input credit card expiration date

Selected with a random uniform distribution between the following dates: [Current Date+30 days...Current Date+1080 days]

**2.4.2.7** Generation of Shipping Address

A new shipping address will be generated for 5% of the requests to create order. If a new shipping address is not generated then the shipping address section of the XML document used as input for this web service will exist but contain no data.

**2.4.2.8** Selection of the length of the list of items

The length of the list of I_ID/QTY pairs should be chosen on a uniform random distribution between [1..MaxCreateOrderIDs]. See clause 1.2.53.

**2.4.3** Processing Definition

**2.4.3.1** Validation

**2.4.3.1.1** <DBMS> Access the database to determine the current payment method (PO or CC) </DBMS>

If the payment method is credit card then PGE emulator web service is invoked to validate the credit card. This is an XML/SOAP request with the following fields:

- CC_NUMBER
- CC_EXPIRY in ISO8601 format to adhere to XML standards
- CC_NAME

The PGE will return a unique authorization code (AUTH_ID).

**2.4.3.2** Update the ADDRESS table if necessary

<DBMS>
If the shipping address information in the input for this web service request does not exist in the address table then a new unique ADDR_ID must be created and a record containing the shipping address information must be added to the ADDRESS table.

**Comment:** The ADDR_CO_ID is obtained by comparing the Country Name to the CO_NAME value in the COUNTRY table.

**2.4.3.3** Calculate the costs of the order

<start atomic operation>

**2.4.3.4** Obtain the value of C_DISCOUNT for the customer with the input C_ID from the CUSTOMER table in the database.

**2.4.3.5** For each item id (I_ID) contained in the input request obtain the value of I_COST from the database.

**2.4.3.6** Calculate the following values:

- SUB_TOTAL = sum (I_COST * QTY) * (1-C_DISCOUNT)
- TAX = SUB_TOTAL * 0.0825
- SHIP_COST = 3.00 + (1.00 * sum (QTY))
- TOTAL = SUB_TOTAL + SHIP_COST + TAX

**2.4.3.7** Determine whether the order can be satisfied with existing stock:
For each line item in the order that was received as input for this web service:

- Retrieve the value of S_QTY from the STOCK table in the database where I_ID is the item id for this line item
- If S_QTY is greater than or equal to the QTY for this line item then set OL_STATUS to "PENDING" for this line item
- If S_QTY is less than QTY for this line item then set OL_STATUS to "BACK_ORDERED"

**2.4.3.8** Create the Order on the database

**2.4.3.8.1** A record is added to the ORDERS table with:

- O_ID set to - a new and unique order number.
- O_C_ID is set to C_ID
- O_DATE is set to the current operating system time and date of the application server
- O_SUB_TOTAL = SUB_TOTAL
- O_TAX = TAX
- O_TOTAL = TOTAL
- O_SHIP_COST = SHIP_COST
- O_SHIP_TYPE = Shipping Type from the input request
- O_SHIP_ADDR_ID is set to C_ADDR_ID or the ADDR_ID returned in clause 2.4.3.2.
- O_AUTH_ID = AUTH_ID or 'PO' if this is a purchase order transaction.
- O_DISCOUNT is set to C_DISCOUNT

**2.4.3.8.2** If any OL_STATUS field for this order was set to "BACK_ORDERED" then set O_STATUS to "BACK_ORDERED" otherwise set O_STATUS to "PENDING"

**2.4.3.8.3** If the O_STATUS is "PENDING" set the S_LAST_UPDATE = the current system time for the first OL_I_ID in the order.

**2.4.3.8.4** For each item in the input request for the web service a record is added to the ORDER_LINE table with:

- OL_ID is unique within the ORDER_LINE record for the order (not necessarily sequential or contiguous)
- OL_O_ID is set to O_ID
- OL_I_ID is set to the I_ID of the item in the input request
- OL_QTY is set to QTY of the item in the input request
- OL_COST is set to the current value of I_COST for this item
- OL_STATUS is set to the value from clause 2.4.3.7

</DBMS>

**2.4.3.8.5** Queue a single XML message to the SHIPPING queue with the following information:

- SHIPPING_O_ID = O_ID
- NOTE to committee: at one point C_ID was passed to the shipping process. We removed it because we could not see what benefit it gave, but if needed, we can add it back in.SHIPPING_REQUEST_TIME = O_DATE in IS08601 format to adhere with XML standards
- SHIPPING_STATUS = O_STATUS

<end atomic operation>

**Comment:** The message queuing product must be commercially available. It must support durable message queues and transactions.

**2.4.4** Output Requirements

The following information is returned to the EB in a web service response.

- Order ID (O_ID)
- Order status (O_STATUS)
- Order total (O_TOTAL)

A sample SOAP WSDL can be found in Appendix E -

## 2.5 Shipping

**2.5.1** Overview

This process handles messages sent from the Create Order interaction via the Shipping queue. It performs the required shipping business logic. For orders where stock is sufficient to ship (flagged as "PENDING") the order is updated to indicate that it has been shipped. For orders that are flagged as "BACK_ORDERED", a message is sent to the STOCK_MANAGEMENT queue indicating a restock is necessary.

**2.5.2** Input Requirements

The input will come in the form of an XML document that is retrieved from the SHIPPING queue. This document will contain the following information:

- Order ID (O_ID)
- SHIPPING_REQUEST_TIME = O_DATE in IS08601 format to adhere with XML standards
- SHIPPING_STATUS

**2.5.3** Processing Definition

For the following atomic operation, if at any point during the following steps an error occurs rollback the operation and restart processing at clause 2.5.3.

<start atomic operations>

**2.5.3.1** Remove the next message from the SHIPPING queue. If there are no messages queued then the business logic should wait until a message is available or the shipping process is shut down.

**2.5.3.2** <DBMS> If the SHIPPING_STATUS is "PENDING" then the order ship date (O_SHIP_DATE) is set to the current operating system time and date of the application server and the order status (O_STATUS) is set to "SHIPPED".

</DBMS>

**2.5.3.3** If the SHIPPING_STATUS is "BACKORDERED":

- <DBMS> Retrieve the orderlines for this order where OL_STATUS is "BACKORDERED". </DBMS>
- Queue a single XML message to the StockManagement queue containing
  - SHIPPING_REQUEST_TIME in ISO8601 format to adhere with XML standards
  - The O_ID
  - The OL_I_ID and OL_QTY for the orderlines having a status of "BACKORDERED".

<end atomic operations>

**2.5.3.4** The business logic defined above should be repeated by going back to clause 2.5.3 and removing the next message from the Shipping queue.

**Comment:** The operation of removing the message from the queue and the database updates must be atomic. Typically, coordination of these operations would be performed by transaction management software. The transaction manager may have to be able to handle multiple resources, such as message queues and database updates. This process is typically known as a two phase commit.

**2.5.4** Output Requirements
None.

## 2.6 Stock Management Process

**2.6.1** Overview

This process handles messages sent from the shipping business process via the STOCK_MANAGEMENT queue. It performs the required stock management business logic.

**2.6.2** Input Requirements

The input will come in the form of an XML document that is retrieved from the STOCK_MANAGEMENT queue. This document will contain the following information:

- The O_ID that is associated with this reorder request.
- The SHIPPING_REQUEST_TIME in ISO8601 format to adhere with XML standards
- A list of REORDER_I_IDs
- For each REORDER_I_ID, there will be a corresponding REORDER_QTY.

**2.6.3** Processing Definition

For the following atomic interaction, if at any point during the following steps an error occurs, immediately rollback the transaction and continue the processing at clause 2.6.3.4.

<start atomic operations>

**2.6.3.1** Remove the next message from the STOCK_MANAGEMENT queue. If there are no messages queued then the business logic should wait until a message is available or the stock management process is shut down. If the de-queue operation fails the transaction is rollback.

**2.6.3.2** For each REORDER_I_ID in the message a SOAP/XML request is sent to the external vendor web service as defined in clause 6.7 (ICE) with the Item ID (REORDER_I_ID), the Quantity (REORDER_QTY), and the O_ID. If the transmission of the message fails, a rollback occurs. Otherwise, the SOAP response must then be received. If the response indicates success then this transaction is committed, otherwise this transaction is rolled-back.

**Comment: The rollback logic may be initiated with, but not implemented with user written code (e.g., re-insertion of records into the queue is not permitted).**

<end atomic operations>

**2.6.3.3** <DBMS> Update the STOCK table setting S_LAST_MODIFIED to the current time for each REORDER_I_ID.

**Comment:** This step was intentionally left out of the atomic operation. This processing would normally occur days later when the external vendor shipment is received. All required business logic for shipping and receiving from the external vendor are not included in the application for ease of benchmarking purposes. This update is included here to force updates to the stock table to prevent the table from being read-only.

**2.6.3.4** The processing of the message that was removed from the STOCK_MANAGEMENT queue is then logged. The exact format of the log is not defined but it must contain the following information:

- The SHIPPING_REQUEST_TIME
- The O_ID for this request.
- The status of the request (either ICESUCCESS or ROLLBACK)
- The time and date that the above processing steps were completed on the application server.

</DBMS>

- **Comment1:** If the status is ROLLBACK only the available information need be logged.
- **Comment2:** To ensure that the full processing for ICE requests occurs, there must be no more than 0.5% ICE requests that result in errors.
- **Comment3:** The Auditor may require a translation of the above log to human readable text.

**2.6.3.5** The business logic defined above should be repeated by going back to clause 2.6.3 and removing the next message from the STOCK_MANAGEMENT queue.

**2.6.4** Output Requirements
**None**

## 2.7 Order Status

**2.7.1** Overview

This web service interaction returns the last *n* orders placed by the customer.

**2.7.2** Input Requirements
This web service interaction is invoked by a web service request and requires the following input data:

- Customer id (C_ID) as chosen in clause 6.2.3.
- Maximum number of orders to return for this request is randomly selected within [1..MAX_ORDERS]. See clause 1.2.49.

- Customer user name (C_BUSINESS_NAME) as defined in clause 4.6.1 if the Business Session has not performed any new customer interactions or the saved BUSINESS_NAME from the last new customer interaction performed by the Business Session (see clause 2.2.2).
- Customer password (C_PASSWD) as defined in clause 4.6.2. if the Business Session has not performed any new customer interactions or the saved PASSWORD from the last new customer interaction performed by the Business Session (see clause 2.2.2).

**2.7.3** Processing Definition

**2.7.3.1** <DBMS> The SUT retrieves the values for C_BUSINESS_NAME and C_PASSWD from the CUSTOMER table for this C_ID.

**2.7.3.2** If the C_BUSINESS_NAME and C_PASSWD retrieved from the database match the C_BUSINESS_NAME and C_PASSWD contained in the input request:

- The SUT retrieves the following customer information:
  - Customer first name (C_CONTACT_FNAME)
  - Customer last name (C_CONTACT_LNAME)
  - Customer phone number (C_CONTACT_PHONE)
  - Customer E-mail address (C_CONTACT_EMAIL)
  - Billing address

- Street address 1 (ADDR_STREET1)
- Street address 2 (ADDR_STREET2)
- City (ADDR_CITY)
- State (ADDR_STATE)
- Zip code (ADDR_ZIP)
- Country name (CO_NAME)

For the max last orders requested the SUT retrieves:

- Order id (O_ID)
- Order date (O_DATE)
- Order subtotal (O_SUB_TOTAL)
- Order tax (O_TAX)
- Order Total (O_TOTAL)
- Order shipping type (O_SHIP_TYPE)
- Order ship date (O_SHIP_DATE)
- Order status (O_STATUS)
- Discount (O_DISCOUNT)
- Order shipping costs (O_SHIP_COST)
- Shipping address
    - Street address 1 (ADDR_STREET1)
    - Street address 2 (ADDR_STREET2)
    - City (ADDR_CITY)
    - State (ADDR_STATE)
    - Zip code (ADDR_ZIP)
    - Country name (CO_NAME)

For each item on the order

- Item id (OL_I_ID)
- Title (I_TITLE)
- Publisher (I_PUBLISHER)
- Cost (OL_COST)
- Quantity (OL_QTY)
- Status (OL_STATUS)

</DBMS>

***2.7.3.3*** If the C_BUSINESS_NAME and C_PASSWD retrieved from the database do not match the C_BUSINESS_NAME and C_PASSWD contained in the input request the SUT returns a standard SOAP exception.

**Note:** This should not happen with the RBE but needs to be checked to ensure correct RBE functions.

**2.7.3.4** If there are no orders for the supplied C_ID, the customer info will be returned and the order info will be left empty.

**2.7.4** Output Requirements

**2.7.4.1** The following information is returned to the EB in a web service response.

- Customer first name (C_CONTACT_FNAME)
- Customer last name (C_ CONTACT_LNAME)
- Customer phone number (C_ CONTACT_PHONE)
- Customer E-mail address (C_ CONTACT_EMAIL)
- Billing address
- Street address 1 (ADDR_STREET1)
- Street address 2 (ADDR_STREET2)
- City (ADDR_CITY)
- State (ADDR_STATE)
- Zip code (ADDR_ZIP)
- Country name (CO_NAME)

A sample SOAP WSDL can be found in Appendix E - .

**2.7.4.2** For each order selected.

- Order id (O_ID)
- Order date (O_DATE) in ISO8601 format to adhere to XML standards
- Order subtotal (O_SUB_TOTAL)
- Order tax (O_TAX)
- Order Total (O_TOTAL)
- Order shipping type (O_SHIP_TYPE)
- Order ship date (O_SHIP_DATE) in ISO8601 format to adhere to XML standards
- Order status (O_STATUS)
- Shipping address
- Street address 1 (ADDR_STREET1)
- Street address 2 (ADDR_STREET2)
- City (ADDR_CITY)
- State (ADDR_STATE)
- Zip code (ADDR_ZIP)
- Country name (CO_NAME)

**2.7.4.3** For each item on the order

- Item id (OL_I_ID)
- Title (I_TITLE)
- Publisher (I_PUBLISHER)
- Cost (OL_COST)
- Quantity (OL_QTY)
- Status (OL_STATUS)

## 2.8 New Products Web Service Interaction

**2.8.1** Overview

This web service interaction returns a list of recently modified items.

**2.8.2** Input Requirements

This web service interaction is invoked by a web service request and requires the following input data:

- The CUTOFF_DURATION is randomly selected within [1..10] minutes.
- A string to use to filter the returned items, SUBJECT_STRING. The EB selects the subject string at random from the set of valid subjects.. (See clause 4.6.3)
- A limit on the number of items to return, ITEM_LIMIT. See clause 1.2.50.

**2.8.3** Processing Definition

**2.8.3.1** <DBMS> Of the entire set of (I_ID, I_TITLE) pairs for items on the selected subject and published after the cutout time, sorted by descending I_PUB_DATE and ascending I_TITLE, the first ITEM_LIMIT entries are returned (or fewer if there are fewer items in the set).

**2.8.3.2** The following search predicate is used:

I_ SUBJECT = SUBJECT_STRING and I_PUB_DATE >= current data and time minus CUTOFF_DURATION

</DBMS>

**2.8.4** Output

The SUT returns the following information in a web service response.

For each item retrieved in clause 2.8.3.1 the SUT returns:

- I_ID
- I_TITLE
- A_FNAME
- A_LNAME

A sample SOAP WSDL can be found in Appendix E - .

## 2.9 Product Detail

**2.9.1** Overview

This web services function returns detailed item information on a number of requested items.

**2.9.2** Input Requirements

This web service interaction is invoked by a web service request and requires the following input data:

- A list of 1 to *MaxProductDetailIDs* I_IDs. See clause 1.2.51.
- The number of I_IDs is chosen randomly from a uniform random distribution.

- The I_IDs are chosen from the non-uniform random distribution NURand(*NURandAProductDetail*, 1, NUM_ITEMS). See clause 1.2.52.

**2.9.3** Processing Definition

<DBMS> The SUT obtains the following information for each selected I_ID.

- I_ID
- I_TITLE
- A_FNAME
- A_LNAME
- I_PUB_DATE
- I_PUBLISHER
- I_SUBJECT
- I_DESC
- I_COST
- I_SRP
- I_AVAIL
- I_ISBN
- I_PAGE
- I_BACKING
- I_DIMENSIONS

</DBMS>

**2.9.4** Output

The SUT returns to the EB the detail information obtained in clause 2.9.3 for all of the requested I_IDs in a web service response.

## 2.10 Change Item

**2.10.1** Overview

This web services function modifies the I_PUB_DATE for a set of items.

**2.10.2** Input Requirements

This web services function is invoked by a web service request and requires the following input data:

- A list of 1 to MaxChangeItemIDs I_IDs. The number of I_IDs is randomly selected within [1..MaxChangeItemIDs]. See clause 1.2.53.
- The I_IDs are randomly selected within [1..NUM_ITEMS]. The I_IDs must be unique within the input list of I_IDs. If uniform random generation generates a duplicate I_ID, iterate the selection until a unique I_ID is chosen.

**2.10.3** Processing Definition

<DBMS> The SUT updates the I_PUB_DATE value for each of the I_IDs, setting I_PUB_DATE =current date and time. </DBMS>

#### 2.10.4 Output Requirements

The SUT returns the following information to the EB in a web service response.

- For each I_ID the SUT returns the following:
  - I_ID
  - I_COST
  - I_PUB_DATE in ISO 8601 format to adhere to XML standards as defined in clause 1.2.15

A sample SOAP WSDL can be found in Appendix E - .

# Clause 3 - **TRANSACTION AND SYSTEM PROPERTIES**

## 3.1 Transaction ACID Properties

It is the intent of this section to define the ACID properties requirements for web service interactions and to specify a series of tests that must be performed to demonstrate that these requirements are met.

### 3.1.1 Introduction

3.1.1.1 All interactions with any database maintaining the tables defined in Clause 1 must be made through a database transaction supporting full ACID properties, as defined in clauses 3.1.2 to 3.1.5.

3.1.1.2 All interactions involving message queue operations must be performed using a commercially available message queue product that supports full ACID properties.

3.1.1.3 No finite series of tests can prove that the ACID properties are fully supported. Passing the specified tests is a necessary, but not sufficient, condition for meeting the ACID requirements. However, for fairness of reporting, only the tests specified here are required and must appear in the Full Disclosure Report for this benchmark.

**Comment**: These tests are intended to demonstrate that the ACID properties are supported by the SUT and enabled during the performance Measurement Interval. They are not intended to be an exhaustive quality assurance test.

3.1.1.4 All mechanisms needed to insure full ACID properties must be enabled during both the ACID test period and the Test Run (as defined in clause 5.4). For example, if the system under test relies on undo logs, then logging must be enabled for all transactions. When this benchmark is implemented on a distributed system, tests must be performed to verify that distributed database transactions (database transactions that are processed on two or more nodes) support the ACID properties.

3.1.1.5 Although the ACID tests may not exercise all types of TPC-W transactions, the ACID properties must be satisfied for all types.

3.1.1.6 Test sponsors reporting TPC-W results on a SUT containing several systems must perform the ACID tests on the system containing the message queuing product, and the database system. If the benchmark utilizes more than one system for message queuing, it is not necessary to perform the ACID tests on all message queuing systems. All Full Disclosure Reports must identify the systems used to verify ACID requirements and full details of the ACID tests conducted and results obtained.

### 3.1.2 Atomicity

3.1.2.1 Atomicity Property Definition

The system under test must guarantee that transactions are atomic. Within a transaction the system will either perform all individual operations on the data, or will assure that no operations leave any effects on the data.

3.1.2.2 Atomicity Tests

The atomicity tests require that the Create Order web service interaction be instrumented so that the update to the database may be aborted while in progress, affecting the final outcome of the update, but without affecting the ability of the SUT to complete the web interaction. The following atomicity tests are completed without other web service interactions in progress.

The following steps describe the atomicity test 1:

Step 1. Request and complete a Create Order web service interaction.

Step 2. Request and complete an Order Status web service interaction for the same EB used in Step 1, using a value of 1 for NUM_ORDERS.

Step 3. The information presented in the SOAP response document from Step 2 must match the order entered in Step 1.

The following steps describe the atomicity test 2:

Step 1. Request an instrumented Order Status for a given EB to view the last order committed.

Step 2. Disable the SHIPPING process such that the queuing software is accepting messages but no messages are processed.

Step 3. Request an instrumented Create Order web service interaction for the EB used in Step 1 with different data than Step 1 and rollback the updates after sending the SHIPPING message but before committing the transaction.

Step 4. Request and complete an Order Status web service interaction for the EB used in Step 1.

Step 5. The information presented in the SOAP response document from Step 4 must make no mention of the order entered in Step 3.

Step 6. The message queue must not contain the message queued in Step 3.

The following steps describe the atomicity test 3:

Step 1. Ensure that the stock management queue is empty.

Step 2. Request an instrumented Create Order web service interaction. The request should include one backorder request (I_QTY > 10). Note the O_ID returned in the SOAP response.

Step 3. The SHIPPING queue should receive and process the new shipping message. The processing should perform the required updates on the database and send the backorder request to the stock management queue, but rollback the changes before committing the transaction and pause the processing of the shipping queue.

Step 4. View the stock management queue. There must be no message on the queue with the O_ID from Step 2.

Step 5. View the shipping queue. It must contain the shipping request with the O_ID from Step 2.

Step 6. View the stock management log and the ICE log. The must be no mention of the O_ID from Step 2.

The following steps describe the atomicity test 4:

Step 1. Request an instrumented New Customer web service interaction. The request must contain an address that is not already in the ADDRESS table.

Step 2. Insert the customer and address data but rollback the transaction instead of committing.

Step 3. Verify that the database does not contain either the customer or address information inserted in Step 2.

### 3.1.3 Consistency

#### 3.1.3.1 Consistency Property Definition

The system under test must guarantee that database transactions are consistent. Assuming that the database is initially in a consistent state, the system will ensure that any TPC-W database transaction takes the database from one consistent state to another.

3.1.3.2 Consistency Conditions

A consistent state for the TPC-W database is defined to exist when:

1. (I_A_ID) is a valid Foreign Key reference to an existing (A_ID)
2. (C_ADDR_ID) is a valid Foreign Key reference to an existing (ADDR_ID)
3. (O_C_ID) is a valid Foreign Key reference to an existing (C_ID)
4. O_SHIP_ADDR is a valid Foreign Key reference to an existing (ADDR_ID)
5. (OL_I_ID) is a valid Foreign Key reference to an existing (I_ID)
6. (OL_O_ID) is a valid Foreign Key reference to an existing (O_ID)
7. (ADDR_CO_ID) is a valid Foreign Key reference to an existing (CO_ID)
8. (S_I_ID) is a valid Foreign Key reference to an existing (I_ID)
9. All O_IDs in the stock management log must be reflected in the database as "BACKORDERED".

3.1.3.3 A TPC-W database, when populated as defined in clause 4.7, must meet the consistency condition defined in clause 3.1.3.2.

3.1.3.4 If data is replicated, as permitted under clause 1.5.5, each copy must meet the consistency condition defined in clause 3.1.3.2. The implementation of the web service interactions must ensure that all consistency conditions defined in clause 3.1.3.2 are maintained without relying on a limited range of input data. However, the implementation of the benchmark is not required to maintain these consistency conditions under arbitrary database transactions.

**Comment**: This implies that no referential integrity is required to be enforced at the database level.

3.1.3.5 Consistency Test

The verification of the consistency between the CUSTOMER and ADDRESS tables (condition #2), between the ORDERS and CUSTOMER tables (condition #3), between ORDER_LINE and ITEM (condition #5), and between the stock management log and the ORDERS table (condition #9) is done as part of each Durability test (see clause 3.1.5.5).

***3.1.1.1*** **Comment:** While maintaining other consistency conditions defined in clause 3.1.3.2 is required, their verification is left to the discretion of the auditor.

### 3.1.4 Isolation

3.1.4.1 Isolation Property Definition

Isolation can be defined in terms of phenomena that can occur during the execution of concurrent database transactions. The following phenomena are considered, given two atomic database transactions, T1 and T2:

P0 ("Dirty Write"): Database transaction T1 reads a data element and modifies it. Database transaction T2 then modifies or deletes that data element, and performs a COMMIT. If T1 were to attempt to re-read the data element, it may receive the modified value from T2 or discover that the data element has been deleted.

P1 ("Dirty Read"): Database transaction T1 modifies a data element. Database transaction T2 then reads that data element before T1 performs a COMMIT. If T1 were to perform a ROLLBACK, T2 will have read a value that was never committed and that may thus be considered to have never existed.

P2 ("Non-repeatable Read"): Database transaction T1 reads a data element. Database transaction T2 then modifies or deletes that data element, and performs a COMMIT. If T1 were to attempt to re-read the data element, it may receive the modified value or discover that the data element has been deleted.

P3 ("Phantom"): Database transaction T1 reads a set of values N that satisfy some <search condition>. Database transaction T2 then executes statements that generate one or more data elements that satisfy the <search condition> used by database transaction T1. If database transaction T1 were to repeat the initial read with the same <search condition>, it obtains a different set of values.

The following table defines four isolation levels with respect to the phenomena P0, P1, P2, and P3.

| Isolation Level | P0 | P1 | P2 | P3 |
|---|---|---|---|---|
| 0 | Not Possible | Possible | Possible | Possible |
| 1 | Not Possible | Not Possible | Possible | Possible |
| 2 | Not Possible | Not Possible | Not Possible | Possible |
| 3 | Not Possible | Not Possible | Not Possible | Not Possible |

The following database transactions are defined:

$T_r$ = Any read-only database transaction used to implement a TPC-W web interaction

$T_u$ = Any update database transaction used to implement a TPC-W web interaction

$T_n$ = Any arbitrary transaction (Although arbitrary, this transaction may not do dirty writes)

Unless otherwise specified, the system under test will ensure that the isolation requirements defined in the table below are met by all database transactions.

| Req. # | For transactions in this set: | these phenomena: | must NOT be seen by this transaction: | Textual Description: |
|---|---|---|---|---|
| 1. | {$T_u$, $T_u$} | P0, P1, P2, P3 | $T_u$ | Level 3 isolation between any two TPC-W update transactions. |
| 2. | {$T_u$, $T_n$} | P0, P1, P2 | $T_u$ | Level 2 isolation for any TPC-W update transactions relative to any arbitrary transaction. |
| 3. | {$T_r$, $T_n$} | P0, P1 | $T_n$ | Level 1 isolation for any TPC-W read-only transaction relative to any arbitrary transaction. |

3.1.4.2 Isolation Tests

The isolation tests require that several web interactions be modified so that a query or an update to the database may be halted while in progress, without affecting the final outcome of the query or the update, and without affecting the ability of the SUT to complete the web service interaction.

3.1.4.2.1 Isolation Test 1

To verify isolation between two TPC-W update transactions, perform the following steps:

Step 1. From EB 1, perform an instrumented Change Item web service interaction to modify any I_ID.

Step 2. Interrupt the processing of the Change Item web service interaction from EB 1 after updating the I_PUB_DATE, but before committing the update.

Step 3. From EB 2, request a Product Detail For the item from Step 1. Verify that the Product Detail web service interaction either waits for EB 1 to resume, or completes. If the Product Detail web service interaction completes verify that the I_PUB_DATE returned is the original date, not the date modified from Step 1.

Step 4. Resume the processing of the Change Item web service interaction interrupted in Step 2.

Step 5. Verify that EB 1 A receives a Change Item SOAP response containing the item's new I_PUB_DATE.

Step 6. Allow EB2 to continue processing. The Product Detail response should reflect the new I_PUB_DATE.

3.1.4.2.2 Isolation Test 2

To verify isolation between a TPC-W update transaction and an arbitrary transaction, perform the following steps:

Step 1. From an EB, modify an item's I_PUB_DATE by initiating an instrumented Change Item web service interaction.

Step 2. Interrupt the processing of the Change Item web service interaction after updating the I_PUB_DATE, but before committing the change.

Step 3. Using a database query and update utility, request a transactional update of the price of the item from step 1. Commit this update as soon as allowed by the database. The update request may hang waiting for the processing of the Change Item web interaction to complete. If the database query utility does not wait verify that it receives an error.

Step 4. Resume the processing of the Change Item web interaction interrupted in step 2.

Step 5. Verify that the EB receives a SOAP response containing the item's new I_PUB_DATE and the old price.

Step 6. If the database query utility waited in step 3, verify that it commits. Verify that the database now reflects both updates.

Step 7. If the database query utility does not wait in step 3, verify that it received an error.

3.1.4.2.3 Isolation Test 3

To verify isolation between a TPC-W read-only transaction and an arbitrary transaction, perform the following steps:

Step 1. From an EB, request and complete an Order Status web service interaction. Ensure that at least one order is returned.

Step 2. Using a database query and update utility, request an transactional update of the quantity of the first line item of the first order returned in Step 1, but do not commit this update.

Step 3. From the same EB, request and complete another Order Status web service interaction to retrieve the same order as in Step 1

Step 4. Verify that the EB's Order Status web service interaction either waits for the update utility to commit, or return an Order Status Page containing the same information as displayed in Step 1.

Step 5. Using a database query and update utility commit the pending updates to the database.

Step 6. If the EB was waiting in Step 4, verify that it has resumed and that it receives an Order Status SOAP response containing the new quantity for the first line item. If the browser was not waiting in Step 4, request and complete a new Order Status web service interaction to retrieve the same order as in Step 1 and verify that the Order Status SOAP response contains the new quantity for the first line item.

3.1.4.2.4 Isolation Test 4

This test is only required if de-queuing of the Shipping queue is multithreaded.

To verify isolation between messages removed from the Shipping queue:

Step 1. Disable the Shipping process such that no shipping queue messages are processed.

Step 2. From an EB, request and complete two successful Create Order web service interactions.

Step 3. Modify the Shipping process such that the messages are removed from the queue and logged to a file by the application, but further processing is paused.

Step 4. Enable the Shipping process. Allow it to remove all items from the queue.

Step 5. Verify that there are no duplicates of the messages in the files(s) from Step 3.

3.1.4.2.5 Isolation Test 5

This test is only required if de-queuing of the Stock Management queue is multithreaded.

To verify isolation between messages removed from the Stock Management queue:

Step 1. Disable the Stock Management process such that no Stock Management queue messages are processed.

Step 2. From an EB, request and complete two successful Create Order web service interactions which force one backordered item (OL_I_QTY > 10) each.

Step 3. Modify the Stock Management process such that the messages are removed from the queue and logged to a file by the application, but further processing is paused.

Step 4. Enable the Stock Management process. Allow it to remove all items from the queue.

Step 5. Verify that there are no duplicates of the messages in the file(s) from Step 3.

### 3.1.5 Durability

3.1.5.1 Durability Property Definition

The system under test must guarantee that transactions are durable. The system will preserve the effects of any committed transaction after recovery from any single point of failure.

**Comment**: No system provides complete durability (i.e., durability under all possible types of failures). The specific set of single failures addressed in clause 3.1.5.3 is deemed sufficiently significant to justify demonstration of durability across such failures. However, the limited nature of the tests listed must not be interpreted to allow other unrecoverable single points of failure.

3.1.5.2 Committed Transaction Definition

A transaction is considered committed when the transaction manager components of the system have either written the log or written the data for the committed updates associated with the transaction to a durable medium.

**Comment 1**: Database transactions can be committed without the user subsequently receiving notification of that fact, as the web interaction may fail after the database transaction has been committed.

**Comment 2**: For interactions that require distributed transaction(s), the distributed transaction manager must also meet the above requirement to satisfy the durable message queue requirements.

3.1.5.3 List of Single Points of Failure

The Single Points of Failure apply to components of the SUT that contribute to the durability requirement.

**Comment 1**: A single test can adequately satisfy the requirements of multiple single points of failure (e.g., A single "system crash test" could be used for the memory failure, system failure, and power failure requirements.

**Comment 2**: The power failure requirement can be satisfied by including sufficient UPS's to guarantee system availability of all components that fall under the power failure requirement for a period of at least 30 minutes. Use of a UPS-protected configuration must not introduce new single points of failure that are not protected by other parts of the configuration. This requirement may be proven either through a measurement or through a calculation of the 30-minute power requirements (in watts) for the portion of the SUT that is protected multiplied by 1.4.

**Medium Failure**: Permanent irrecoverable failure of any single durable medium active during any Measurement Interval.

If main memory is used as a durable medium, then it must be considered as a potential single point of failure. Sample mechanisms to survive single durable medium failures are database archiving in conjunction with a redo (after image) log, and mirrored durable media. If memory is the durable medium and mirroring is the mechanism used to ensure durability, then the mirrored memories must be independently powered.

**Memory Failure**: Failure of all or part of memory (loss of contents).

This implies that all or part of memory has failed. This may be caused by a loss of external power or the permanent failure of a board equipped with memory.

**System Failure**: Instantaneous interruption (system crash/system hang) in processing which causes all or part of the processing of atomic transactions to halt.

**Comment 1**:This may imply abnormal system shutdown which requires loading of a fresh copy of the operating system from the boot device. It does not necessarily imply loss of volatile memory. When the recovery mechanism relies on the pre-failure contents of volatile memory, the means used to avoid the loss of volatile memory (e.g., an Un-interruptible Power Supply) must be included in the system cost calculation. A sample mechanism to survive an instantaneous interruption in processing is an undo/redo log.

**Comment 2**: In configurations where more than one System participates in an atomic transaction and are connected via a medium other than an integrated bus (e.g., bus extender cable, high speed LAN, or other connection methods between the Systems that could be vulnerable to a loss from physical disruption), the instantaneous interruption of this communication is included in this definition as an item that needs to be tested. Interruption of one instance of redundant connections is required.

**Comment 3**: It is not the intention of this clause to require interruption of communication to disk towers or a disk subsystem where redundancy exists. For example, log disks can be assumed to provide redundancy for data disks.

**SUT Power Failure**: Loss of all external power to the SUT for an indefinite time period. This must include at least all portions of the SUT that participate in the database portion of transactions.

This type of failure is sufficiently exercised by removing power from every system that contributes to satisfying the durability requirement. If a group of systems is providing a given function (e.g., a clustered database server), then failing any one system in that group is a sufficient test scenario.

3.1.5.4 Durable Medium Definition

A durable medium is a data storage medium that is either:

1. An inherently non-volatile medium (e.g., magnetic disk, magnetic tape, optical disk, etc.) or
2. A volatile medium that will ensure the transfer of data automatically, before any data is lost, to an inherently non-volatile medium after the failure of external power independently of reapplication of external power. (A configured and priced Un-interruptible Power Supply (UPS) is not considered external power.)

**Comment**: A durable medium can fail; this is usually protected against by replication on a second durable medium (e.g., mirroring) or logging to another durable medium. Memory can be considered a durable medium if it can preserve data long enough to satisfy the requirement stated in item 2 above, for example, if it is accompanied by an Un-interruptible Power Supply, and the contents of memory can be transferred to an inherently non-volatile medium during the failure. Note that no distinction is made between main memory and memory performing similar permanent or temporary data storage in other parts of the system (e.g., disk controller caches).

3.1.5.5 Durability Tests

3.1.5.5.1 Durability Test for DBMS

For each component susceptible to one of the failure types defined in clause 3.1.5.3 perform the following steps:

Step 1. Obtain the total number of rows in the ORDERS table to determine the current count of orders (order_count1) in the database. Also obtain the total number of rows in the ORDERS table with a status of "BACKORDERED" (back_count1).

Step 2. Ensure that the STOCK_MANAGEMENT log and all queues are empty.

Step 3. Start the mix of web service interactions using the same number of EB's used for the reported SIPS metric.

Step 4. Once all EB's have started requesting web service interactions, run for at least 5 minutes and keep a count of the number of Create Order web service interactions successfully completed by all EB's as well as the count of Create Order web service interactions which had a "BACKORDERED" status (OL_I_QTY >10)

Step 5. Cause the failure selected from the list in clause 3.1.5.3.

Step 6. Stop the RBE and collect the total number of Create Order web service interactions successfully completed by all EB's (RBE-order_count). Obtain from the RBE a count of all Create Order interactions which had a status of "BACKORDERED" (RBE-back_count).

Step 7. If necessary, stop and restart the system under test using normal recovery procedures, where applicable.

Step 8. Repeat step 2 to determine the current count of orders (order_count2) in the database. Verify that (order_count2 - order_count1) is greater than or equal to the number of successfully completed Create Order web service interactions (RBE-order_count). If there is an inequality, the difference must be less than or equal to the number of EB's active during this test.

**Comment**: This difference should be due only to database transactions which were committed on the system under test, but for which the SOAP response message was not returned to the EB before the failure.

Step 9. Verify that (back_count2 - back_count1) is greater than or equal to the number of successfully completed Create Order web service interactions resulting in a status of "BACKORDERED" (RBE-back_count). If there is an inequality, the difference must be less than or equal to the number of EB's active during this test.

**Comment**: This difference should be due only to database transactions which were committed on the system under test, but for which the SOAP response message was not returned to the EB before the failure.

Step 10. Verify that Consistency Conditions 2, 3, (7 and 9 ), as specified in clause 3.1.3.2, are still met.

3.1.5.5.2 Durability Test for Message Queues and Distributed Transaction Manager.

Step 1. Ensure the ICE log is empty. Modify the ICE process such that the minimum delay is 30 seconds. Modify the Shipping Process such that there is a minimum delay of 5 seconds after the database update but before committing the atomic operation.

Step 2. Ensure the Shipping queue and Stock Management queue and log are empty.

Step 3. Start the workload. Wait a minimum of 5 minutes.

Step 4. For the system containing the Stock Management and Shipping queues, cause the failure selected from the list in clause 3.1.5.3.

Step 5. Stop the workload.

Step 6. Recover the system. Allow the distributed transaction manager to recover and resume the Shipping and Stock Management processes. Allow the Shipping process and Stock Management process to complete all items remaining on their respective queues.

(note: It is permissible to adjust the delay time in the ICE and Shipping processes to expedite this step)

Step 7. Verify that the DBMS contains no ORDERS with a status of "PENDING".

Step 8. Obtain from the ICE log the count of unique O_IDs.

Step 9. Obtain from the RBE the count of all Create Order interactions which returned a response status of "BACKORDERED" .

Step 10. Verify that the counts obtained in Steps 8 and 9 are equivalent. If the counts do not match the difference must be no greater than the number of configured EBs and the count from Step 8 must be greater than Step 9.

# Clause 4 - **Scaling and Database Population**

## 4.1 General Scaling Rule

**4.1.1** The throughput of the TPC-W benchmark is driven by the activity of the Emulated Browsers (EB's) connected to the store front. Each EB emulates only one Business Session at a time. To increase the throughput demand on the SUT, the number of EB's configured has to be increased. The business requires a number of rows to populate the tables of the database along with some storage space to maintain the data generated during a defined period of activity called the 60-day period. The following requirements define how storage space and database population scale with throughput.

**4.1.2** The intent of the scaling requirements is to maintain the ratio between the web interaction load presented to the SUT, the cardinality of the tables accessed by the interactions, the required space for storage, and the number of EB's generating the transaction load.

**4.1.3** Should any scaling value in clause 4.2 be exceeded, the others must be increased proportionally to maintain the same ratios among them as in clause 4.2.

**4.1.4** The reported throughput may not exceed the maximum allowed by the scaling requirements in clause 4.2 and the pacing requirements in clause 5.2. While the reported throughput may fall short of the maximum allowed by the configured system, the price/performance computation (see clause 7.1) must report the price for the system as actually configured.

**4.1.5** The configured EB's must remain active and generating web interactions throughout the entire measurement interval.

## 4.2 Scaling Requirements

**4.2.1** Database scaling is defined by the number of EB's configured for SIPS, i.e., it is defined by the size of the supported customer population.

**4.2.2** The cardinality of the ITEM table, NUM_ITEMS, will be fixed at 100,000.

**4.2.3** The cardinality of the STOCK table is a function of the ITEM table, as defined in clause 4.3.

**4.2.4** The cardinality of the AUTHOR table is a function of the ITEM table, as defined in clause 4.3

**4.2.5** The cardinality of the COUNTRY table is fixed.

**4.2.6** The initial cardinality of the other tables is a function of the number of EB's configured for the SIPS metric. The number of EB's active during the Test Run must be at least 90% and not more than 100% of the EB's configured in the database.

**4.2.7** The reported SIPS throughput is required to satisfy the following inequalities:

$$0.89 * \#\text{active EB's} \leq \text{SIPS} \leq 1.78 * \#\text{active EB's}$$

**Comment**: The intent of this clause is to prevent reporting a throughput that exceeds the maximum, where the maximum throughput is achieved with infinitely fast web service interactions resulting in a minimal average response time. This is computed to be 1.78 SIPS per active EB. To prevent over-scaling the SUT, the throughput cannot fall short of the above 0.89 SIPS per active EB, which represents 50% of the computed maximum throughput.

## 4.3 Database Cardinality

The following scaling requirements represent the initial configuration for the test described in Clause 5:

For each table that composes the database, the cardinality of the initial population is specified as follows:

| Table Name | Cardinality (in rows) | Typical Row Length (in bytes) | Typical Table Size (in bytes) |
|---|---|---|---|
| CUSTOMER | 192 * (number of configured EBs) | 575 | 11,040k |
| COUNTRY | 92 | 70 | 6.44 k |
| ADDRESS | 1.4 * CUSTOMER | 154 | 4,139 k |
| ORDERS | 1 * CUSTOMER | 85 | 1,632 k |
| ORDER_LINE | 5 * ORDERS | 166 | 15,936K |
| AUTHOR | .25 * NUM_ITEMS | 49 | 1,225k (assuming NUM_ITEMS = 100K) |
| ITEM | NUM_ITEMS | 822 | 82,200K (assuming NUM_ITEMS = 100K) |
| STOCK | NUM_ITEMS | TODO | TODO (assuming NUM_ITEMS = 100K) |

**Note 1**: Table sizes are computed for 100 EB's

**Note 2**: The typical row lengths and table sizes given above are examples of what could result from an implementation. They are not requirements. They do not include storage and access overheads.
**Note 4:** No variation is allowed on table cardinality except on ORDER_LINE where the cardinality will vary slightly due to the random number of rows generated per order as specified in clause 4.7.1. Cardinality must meet a minimum requirement of 4.95 times the number of rows in the ORDER table.

- The increment (granularity) for scaling the EB population is one EB.
- Typical lengths and sizes given here are examples of what could result from an implementation. They are not requirements. They do not include storage/access overheads.

- The symbol “k” means one thousand and “M” means one million.

## 4.4 60-Day Space Computation

The storage space required for the database for the 60-day period must be determined as follows:

1. The test database must be built including the initial database population (see clause 4.7.1) and all indices present during the test.

2. The test database must be built to sustain the reported throughput during an eight hour period. If operations are required to meet this space requirement, these must occur within the Measurement Interval.

3. The growth of the database, G in bytes, should be measured as the initial size of the database compared against the size of the database at the end of the Test Run (see clause 5.4).

4. Assuming TI is the total number of web service interactions processed during the duration of the Test Run, and SIPS is the reported throughput, 60-day space in bytes is calculated as follows:

   60-Day-Space = Initial Space + ((G/TI) * SIPS * 3600 * 8 * 60)

5. The free space present in the test database is considered as part of the 60-Day-Space.

## 4.5 Log Requirements

### 4.5.1 Web Server Access Log

There must be enough space configured on the SUT to store Web Server Access Logs, in Common Log Format, as specified in clause 2.1.17, for a period of 8 hours. The space required for this is determined as follows:

8-hour-web-log-space = (LW/TI) * SIPS * 3600 * 8

TI is the total number of web service interactions processed during the duration of the Test Run, and SIPS is the reported throughput. LW must be measured as the size of the Web Server log file at the end of the test duration minus its size at the beginning of the Test Run. At the end of the Test Run, it should be verified that the data for the log file is completely written to durable media.

The service that is performing the above logging function must be capable of switching logs while continuing to process incoming request.

#### 4.5.2 Stock Management Log Requirements

There must be enough space configured on the SUT to store the Stock Management Log as specified in clause 2.6.3.4, for a period of 8 hours. The space required for this is determined as follows:

$$\text{Stock Management Log Space} = (LS/TI) * SIPS * 3600 * 8$$

TI is the total number of web service interactions processed during the duration of the Test Run, and SIPS is the reported throughput. LS must be measured as the size of the stock management log at the end of the Test Run minus the size of the stock management log at the beginning of the Test Run. At the end of the Test Run, it should be verified that the data for the log file is completely written to durable media.

#### 4.5.3 Database Log Requirements

There must be enough space configured on the SUT to store the Database Log for a period of 8 hours. The space required for this is determined as follows:

$$\text{Database Log Space} = \text{Initial Space} + ((LD/TI) * SIPS * 3600 * 8)$$

TI is the total number of web service interactions processed during the duration of the Test Run, and SIPS is the reported throughput. LD must be measured as the size of the database log at the end of the Test Run minus the size of the database log at the beginning of the Test Run.

### 4.6 Database Population

The test described in Clause 5 requires that the database be properly scaled with the initial population. It is allowed, but not required, to reload or rollback the database to its initial population before any Test Run. No other alteration to the defined database population is allowed at any time other than the use of one of the web interaction mixes defined in clause 5.1.

**4.6.1** The customer user name (C_BUSINESS_NAME) must be generated as the string returned by DigSyl(C_ID).

**Example:** Given a C_ID of 3719, C_BUSINESS_NAME is generated as: DigSyl(3719) = RIULOGNG.

**Comment:** Because C_ID's are unique numbers, DigSyl associates to each C_ID a unique C_BUSINESS_NAME.

**4.6.2** The customer password (C_PASSWD) must be generated as the string returned by DigSyl(C_ID) converted to all lower case characters.

**Example:** Given a C_ID of 3719, the resulting C_BUSINESS_NAME is RIULOGNG and the resulting C_PASSWD is riulogng.

**4.6.3** The item subject (I_SUBJECT) must be chosen from a uniform random distribution consisting of the values in the following list:

| | | |
|---|---|---|
| ARTS | HOME | RELIGION |
| BIOGRAPHIES | HUMOR | ROMANCE |
| BUSINESS | LITERATURE | SELF-HELP |
| CHILDREN | MYSTERY | SCIENCE-NATURE |
| COMPUTERS | NON-FICTION | SCIENCE-FICTION |
| COOKING | PARENTING | SPORTS |
| HEALTH | POLITICS | YOUTH |
| HISTORY | REFERENCE | TRAVEL |

**4.6.4** The customer email address field (C_CONTACT_EMAIL) is generated by the concatenation of the corresponding value in C_BUSINESS_NAME followed by the special character "@" followed by a random a-string [2 .. 9] followed by the string of characters ".com".

Example: Given a C_BUSINESS_NAME of RIULOGNG, C_CONTACT_EMAIL, the customer email address, may be RIULOGNG@bjs2acKd.com.

**4.6.5** The country name (CO_NAME) must be chosen from a uniform random distribution consisting of the values in the following list, shown here:

| | | | |
|---|---|---|---|
| United States | Botswana | Fiji | Mauritius |
| United Kingdom | Brazil | Finland | New Zealand |
| Canada | Bulgaria | Gabon | Norway |
| Germany | Cayman Islands | Gibraltar | Pakistan |
| France | Chad | Greece | Philippines |
| Japan | Chile | Guam | Poland |
| Netherlands | China | Hong Kong | Portugal |
| Italy | Christmas Island | Hungary | Romania |
| Switzerland | Colombia | Iceland | Russia |
| Australia | Croatia | India | Saudi Arabia |
| Algeria | Cuba | Indonesia | Singapore |
| Argentina | Cyprus | Iran | Slovakia |
| Armenia | Czech Republic | Iraq | South Africa |
| Austria | Denmark | Ireland | South Korea |
| Azerbaijan | Dominican Republic | Israel | Spain |
| Bahamas | Eastern Caribbean | Jamaica | Sudan |
| Bahrain | Ecuador | Jordan | Sweden |
| Bangla Desh | Egypt | Kazakhstan | Taiwan |
| Barbados | El Salvador | Kuwait | Thailand |
| Belarus | Estonia | Lebanon | Trinidad |
| Belgium | Ethiopia | Luxembourg | Turkey |
| Bermuda | Falkland Island | Malaysia | Venezuela |
| Bolivia | Faroe Island | Mexico | Zambia |

**4.6.6** The item backing (I_BACKING) must be chosen from a uniform random distribution consisting of the values in the following list:

- HARDBACK
- PAPERBACK
- USED
- AUDIO
- LIMITED-EDITION

**4.6.7** The item dimensions (I_DIMENSIONS) must be generated by the concatenation of 3 numbers randomly selected within [00.01..99.99] separated by an "x".

Example: 12.25x16.50x1.25

**4.6.8** The author id (I_A_ID) associated with each item must generated as follows:

```
if I_ID <= (NUM_ITEMS/4)
        I_A_ID = I_ID
else
        I_A_ID = randomly selected within [1 .. NUM_ITEMS/4]
```

**4.6.9** The customer payment method (C_PAYMENT_METHOD) must be chosen as a uniform random distribution over the enumerated set {"PO" ,"CC"}.

## 4.7 Table Population Requirements

**4.7.1** The initial database population must be comprised of the following (where NUM_CUSTOMERS is the number of records in the Customer table):

- **NUM_ITEMS rows in the ITEM table with:**
- I_ID unique within [1 .. NUM_ITEMS]
- I_TITLE random a-string [14..60]
- I_A_ID generated according to clause 4.6.8
- I_PUB_DATE random date between January 1, 1930 and current date
- I_PUBLISHER random a-string [14 .. 60]
- I_SUBJECT generated according to clause 4.6.3
- I_DESC random a-string [100 .. 500]
- I_SRP random within [1.00 .. 9,999.99]
- I_COST generated as I_SRP – (random within [(0 .. 0.5) * I_SRP])
- I_AVAIL generated as I_PUB_DATE + (random within [1 .. 30] days)
- I_ISBN random a-string of 13 characters
- I_PAGE random within [20 .. 9,999]
- I_BACKING, variable size text, generated according to clause 4.6.6
- I_DIMENSIONS (length x width x height of the book), generated according to clause 4.6.7

- **NUM_ITEMS rows in the STOCK table with:**
- S_I_ID unique within [1 .. NUM_ITEMS]
- S_QTY fixed at 10
- S_LAST_MODIFIED current system time

- **92 rows in the COUNTRY table with:**
- CO_ID unique within [1 .. 92]
- CO_NAME selected from the table in clause 4.6.5

- **(NUM_ITEMS / 4) rows in the AUTHOR table with:**
- A_ID unique within [1 .. (NUM_ITEMS / 4)]
- A_FNAME random a-string [3 .. 20]
- A_LNAME random a-string [3 .. 20]

- **(192 * # EB's) rows in the CUSTOMER table with:**
- C_ID unique within [1 .. 192 * # EB's]
- C_BUSINESS_NAME generated according to clause 4.6.1
- C_BUSINESS_INFO random a-string [40..100]
- C_PASSWD generated according to clause 4.6.2
- C_CONTACT_FNAME random a-string [8 .. 15]
- C_CONTACT_LNAME random a-string [8 .. 15]

- C_ADDR_ID = C_ID
- C_CONTACT_PHONE random n-string [9 .. 16]
- C_CONTACT_EMAIL generated according to clause 4.6.4
- C_PAYMENT_METHOD generated according to clause 4.6.9
- C_CREDIT_INFO random a-string [40..300]
- C_PO set to equal C_ID
- C_DISCOUNT random within [0.00 .. 0.50]

- **(10 * NUM_CUSTOMERS) rows in the ORDERS table with:**
- O_ID unique within [1 .. (10 * NUM_CUSTOMERS)]
- O_C_ID selected so each C_ID is used exactly 10 times
- O_DATE generated as current date and time – random within [1 .. 60] days
- O_SUB_TOTAL generated as sum of (OL_I_COST * OL_QTY) for this ORDER
- O_TAX generated as O_SUB_TOTAL * 0.0825
- O_SHIP_TYPE selected at random from the following:
    - AIR, UPS, FEDEX, SHIP, COURIER, MAIL
- O_SHIP_DATE generated as O_DATE + random within [0 .. 7] days
- O_SHIP_COST = 3.00 + (1.00 * count_of_items_in_order)
- O_TOTAL generated as O_SUB_TOTAL + O_TAX + O_SHIP_COST
- O_SHIP_ADDR_ID = random within [1 .. (1.4 * NUM_CUSTOMERS)]
- O_STATUS selected 90% as SHIPPED, and 10% as BACK ORDERED
- O_DISCOUNT set to zero
- O_AUTH_ID set to 'INITIALPOP'

- **For each row in the ORDERS table, a number of rows in the ORDER_LINE table, where the number is selected at random within [1..10] with:**
- OL_ID unique within [1 .. number of ORDER_LINEs selected for this ORDER]
- OL_O_ID = O_ID
- OL_I_ID random within [1 .. NUM_ITEMS] and unique within each ORDER
- OL_QTY random within [1 .. 10] except for first line. If O_STATUS is BACK ORDERED, OL_QTY for first line is set to S_QTY +1.
- OL_STATUS set to SHIPPED except for first line. If O_STATUS is BACK ORDERED, OL_STATUS is set to BACK ORDERED.

- OL_I_COST random within [1.00 .. 9,999.99]

- **(1.4 * NUM_CUSTOMERS) rows in the ADDRESS table with:**
- ADDR_ID unique within [1 .. (1.4 * NUM_CUSTOMERS)]
- ADDR_STREET1 random a-string [15 .. 40]
- ADDR_STREET2 random a-string [15 .. 40]
- ADDR_CITY random a-string [4 .. 30]
- ADDR_STATE random a-string [2 .. 20]
- ADDR_ZIP random a-string [5 .. 10]
- ADDR_CO_ID random within [1 .. 92]

**4.7.2** The implementation may not take advantage of the fact that some fields are initially populated with a fixed value. For example, storage space cannot be saved by defining a default value for any particular field and storing this value only once in the database.

## 4.8 Customized Load Utilities

The load program must use the random number generator as defined in clause 6.3 for all random data generation requirements.

# Clause 5 **PERFORMANCE METRICS AND RESPONSE TIME**

The two primary metrics of the TPC-W benchmark are the number of Service Interactions Per Second (SIPS), and a price performance metric defined as Dollars/SIPS ($/SIPS).

## 5.1 Web Service Interaction Mix Requirements

The TPC-W workload comprises a set of web service interactions specified in detail in Clause 2. Over the Measurement Interval, each EB must maintain the mix of web service interactions specified in the table below. The EB may deviate from the specified percentages as a natural result of the finite random selection process. The maximum deviation allowed is (0.01 x P) where P is any of the required mix percentages. **Example**: If the New Products List web service interaction is 10.00%, it is required that the average mix percentage of New Product List during the Measurement Interval falls between 9.99% and 10.01%.

**Comment1**: For the purpose of computing the mix, only successful web service interactions may be counted.

**Mix of Web Service Interactions**

| Web Service Interaction | Required Mix |
|---|---|
| **New Customer** | 1.00% |
| **Change Payment Method** | 5.00 % |
| **Create Order** | 50.00 % |
| **Order Status** | 20.00 % |
| **New Products List** | 10.00 % |
| **Product Detail** | 12.00 % |
| **Change Item** | 2.00 % |

### 5.1.1 Regulation of Transaction Mix

Transaction types must be selected randomly while maintaining the required percentage of mix for each transaction type over the measurement interval. This must be done using the following technique:

A weight between 0.0 and 1.0 is associated to each web interaction type specified in the above table. The sum of all the weights must be 1.0. The required mix is achieved by selecting each new web interaction randomly from this weighted distribution. For the purpose of achieving the required transaction mix, the EB can not dynamically adjust the weight associated to each web interaction during the measurement interval.

The EB can have at most one outstanding web service interaction at any point in time. This means that the EB must wait for the preceding web service interaction to complete successfully before submitting a request for the next interaction.

**5.1.2** Stock Management and Shipping Requirements

Shipping and Stock management are background processes that are not driven by the RBE directly. They are not counted as part of the mix.

For every Create order Web Service interaction, a corresponding message is queued to the Shipping process (see clause 2.5). During the Measurement Interval, the Shipping service must process at least 99.8% of the order requests placed on the Shipping queue during the measurement Interval.

For every Shipping queue entry with a status of "BACKORDER", a corresponding message is queued to the Stock Management process (see clause 2.6). During the Measurement Interval, the Stock Management service must successfully process at least 99.8% of the reorder requests placed on the Stock Management queue during the measurement Interval. See clause 8.7 for reporting requirements.

## 5.2 Web Service Interaction Response Time

**5.2.1** Web Service Interaction Response Time (SIRT)

The Web Service Interaction Response Time (SIRT) is the time taken to perform a successful web interaction by an EB. It is defined as:

SIRT = T2 - T1

where:

T1 and T2 are measured at the RBE machine;

T1 = a time measured before the first byte of the web service interaction's first SOAP message is sent on the wire by the RBE to the SUT; and

T2 = a time measured after the last byte of the web service interaction's last SOAP message is received off the wire by the RBE from the SUT.

***5.2.1.1*** The resolution of the time stamps must be at least 0.1 seconds.

***5.2.1.2*** SIRTs are defined only for successful web service interactions.

***5.2.1.3*** For the purpose of calculating SIRT, the bytes of a SOAP message include the HTTP request message, or the HTTP response message, that encapsulates the SOAP message.

***5.2.1.4*** Any time spent opening a TCP/IP connection in order to send the first byte of the first

SOAP message must be included in the SIRT. That is, T1 must occur prior to this connection open.

***5.2.1.5*** Any time spent doing SSL/TLS handshakes, or other SSL/TLS protocol, in order to send the first byte of the first SOAP message, must be included in the SIRT. That is, T1 must occur prior to these SSL/TLS communications.

***5.2.1.6*** T1 may be measured prior to sending on the wire the first byte of the web service interaction's first SOAP message, but after XML-serialization, or after encryption, of that message, provided that the constraints of Comments 4 and 5 are met. (For example, if a new bulk encryption key must be negotiated prior to the encryption, T1 must be measured prior to the associated SSL/TLS communications.)

***5.2.1.7*** T2 may be measured after receipt off the wire of the last byte of the last SOAP message, but prior to decryption, or prior to XML-deserialization, of this message. However, for this T2 to be a proper measurement, the RBE or EB must subsequently confirm that the interaction's Output Requirements have been met, (For example, the last SOAP message must be decrypted and checked for the absence of a SOAP fault or other error message from the SUT.)

***5.2.1.8*** T1 and T2 may be measured at any point of execution within the RBE machine provided that all constraints on T1 and T2 are met.

**5.2.2** During the Measurement Interval, at least 90% of the successful web service interactions of each type must have a SIRT of less than the constraint specified (in seconds) for that web service interaction in the table below. For example, at least 90% of all successful Create order web service interaction must have a SIRT of less than 4 seconds.

| | New Customer | Change Payment | Create Order | Order Status | New Products | Product Detail | Change Item |
|---|---|---|---|---|---|---|---|
| **90% SIRT Constraint** | 3 | 3 | 4 | 4 | 1 | 1 | 1 |

**5.2.3** Over the Measurement Interval, for each type of web service interaction, the average SIRT of the successful web service interactions must be no more than 0.1 second longer than the 90$^{th}$ percentile SIRT of those interactions.

## 5.3 Computation of Throughput Rating

**5.3.1** The throughput for the Measurement Interval is computed as the total number of successful web service interactions within the Measurement Interval divided by the length of the Measurement Interval in seconds.

**Comment**: Web service interactions can be included in the computation of the reported throughput only if their SIRT is fully within the bounds of the Measurement Interval.

**5.3.2** The reported throughput must be measured, rather than interpolated or extrapolated, and expressed to exactly one decimal place, rounded down to the tenths place. For example, suppose 105.548 SIPS is measured on a test for which all 90% SIRT constraints are met and 117.572 SIPS is measured on a test for which some 90% SIRT constraints are exceeded. Then the reported SIPS is 105.5 rather than some interpolated value between 105.548 and 117.572. This means any run that does not meet the 90% requirements is not valid.

## 5.4 Test Run Requirements

The Test Run consists of a ramp up followed by stabilization period followed by steady state which contains the reported measurement interval. At least one checkpoint must be started and completed after ramp-up and before the start of the measurement interval.

**5.4.1** Steady State

***5.4.1.1*** The reported throughput must be computed over a Measurement Interval during which the throughput level is in a steady state condition that represents the true sustainable performance of the SUT. Steady state is easy to define (e.g., sustainable throughput) but difficult to prove. The auditor is required to report the method used to verify steady state sustainable throughput and the reproducibility of measured results. The auditor is encouraged to use available monitoring tools to help determine the steady state. The steady state period must be at least 3 hours.

***5.4.1.2*** In order to meet the requirement that all EB's remain active throughout the stabilization and steady states, (see clause 5.4.1), an EB, which encounters an error or failure during a web service interaction, must try to complete the web interaction successfully through an error recovery process. This could include retrying the interaction by resending the web service request to the SUT. For example, if a TCP-IP connection to the SUT is abnormally closed by the peer, the socket may need to be reinitialized and a retry performed.

For each retry that occurs within a web service interaction, the retry must be logged along with the interaction type, a unique identifier for the interaction, and a timestamp indicating when the error occurred. While the format of the log is not specified it must contain the above requirements. For example, if the 3476$^{th}$ Create Order from the 7$^{th}$ EB retries 3 times, the log might contain the following:

| | | |
|---|---|---|
| RETRY | CreateOrder.007.0003476 | 03/10/15 14:13:05.7 |
| RETRY | CreateOrder.007.0003476 | 03/10/15 14:13:05.9 |
| RETRY | CreateOrder.007.0003476 | 03/10/15 14:13:06.3 |

The implementation of the RBE must ensure that any retries that occur during the Test Run are reflected in the log as occurring during the Test Run independent of whether the associated interaction completes during the Test Run (i.e., if a given web service interaction has not completed by the end of the Test Run, any retries it performed must nonetheless be logged).

No web service interaction is permitted to retry more than 20 times. If this occurs, the Test Run is non-compliant. This condition can be tested by verifying the log of retries contains no more than 20 entries for a given unique web service interaction identifier.

**Comment 1:** In general, one cannot assume that web service interactions are repeatable operations. For the purposes of this benchmark however, all interactions are considered repeatable. If the environment used to implement the EB provides a facility to implicitly retry web service interactions under certain failure scenarios, to be used, that facility must provide some means to synthesize the information equivalent to that required for retry logging above.

**Comment 2**: It not permissible to prematurely close the Business Session as an error recovery mechanism.

**Comment 3:** The EB must not insert any artificial delay as part of the retry process (See clause 6.1.11~~6.1.12~~)

***5.4.1.3*** Some aspects of the benchmark implementation can result in systematic but small variations in sustained throughput. These variations must be limited to 2% of the reported throughput over the steady state period. During steady state, no interval with a duration of 30 minutes (with a granularity of 30 seconds) can have a computed throughput that is lower than 98% of the reported throughput.

**Comment:** This is an attempt to detect situations that would degrade performance over a longer run (e.g., memory leaks, disk fragmentation, etc...)

***5.4.1.4*** To prevent significant alteration to the properly scaled database population, the mix of web service interactions and the requirements summarized in clause 5.5 must be followed during the entire Test Run as well as during any other time between Test Runs.

**5.4.2** Measurement Interval

***5.4.2.1*** The measurement interval must extend uninterrupted for a minimum of 2 hours within steady state.

***5.4.2.2*** Although the Measurement Interval may be as short as 120 minutes, the SUT must be configured so that it is possible to run the mix of web service interactions for 8 hours of uninterrupted execution (as defined in clause 6.10.1) while maintaining full ACID properties.

**Comment 1**: For example, the media used to store the database log data until it can be archived without interruption of processing must be configured if required to recover from any single point of failure

**Comment 2**: An example of a configuration that would not comply is one where a log file is allocated such that better performance is achieved during the measured portion of the run than during the remaining portion of any full throughput period, perhaps because a dedicated device was used initially but space on a shared device is used later in the full throughput period.

**Comment 3**: An example of a compliant implementation would be one where the entire database is placed on redundant storage and the database log data is stored on circular files large enough to span a complete checkpoint cycle.

**5.4.3** Checkpoints

***5.4.3.1*** Some systems do not write modified database records/pages to durable media at the time of modification, but instead defer these writes. At some subsequent time, the modified records/pages are written to make the durable copy current. This process is defined as a checkpoint in this document.

***5.4.3.2*** It is a requirement that no recovery data older than 15 minutes prior to an instantaneous interruption be used for recovery. The consequence of this requirement is that the database contents stored on durable media cannot at any time during the Test Run be more than 15 minutes older than the most current state of the database (±5%).

**Comment:** It is assumed that if a checkpoint starts and completes within 15 minutes, the above requirement is met. It is not required to complete two checkpoints in 15 minutes.

***5.4.3.3*** All work required to perform a checkpoint must occur in full at least once before each Measurement Interval begins but after all EBs are active.

***5.4.3.4*** At least 90% and at most 100% of the configured EB's must be active throughout the Stabilization and Steady State periods (see clause 4.2.6).

## 5.5 Measurement Interval Requirements

The following requirements must all be met for a measurement interval to be considered valid. All of these requirements refer to metrics that were taken during the measurement interval.

- 98% of the items put on the shipping queue by the Create Order web interaction during the measurement interval must be processed by the shipping process during the measurement interval.
- 98% of the items put on the stock management queue by the shipping process during the measurement interval must be processed by the shipping process during the measurement interval.
- The percentage of errors returned by the ICE must be less than 5%. If a request is retried and subsequently succeeds this is not counted as an error.

- The percentage of New Customer requests that chose PO as the payment method must be between 49.4% and 50.5%.
- The percentage of Change Payment Method requests that chose PO as the payment method must be between 49.4% and 50.5%.
- The percentage of Create Order requests that include a shipping address change must be between 4.95% and 5.05%.
- The number of Create Order requests that had one item backordered must be at least 10%.
- The mix of web service interactions must meet the requirements as specified in clause 5.1.
- The Measurement Interval must be selected from a valid Test Run (see clause 5.4).

The response time constraints must be met as per clause 5.2.

# Clause 6 - **SUT, RBE AND NETWORK DEFINITIONS**

## 6.1 Remote Business Emulator (RBE)

**6.1.1** The Remote Business Emulator (RBE) is the software component that drives the TPC-W workload. It emulates Business using a computer to request services from the System Under Test (SUT).

**6.1.2** The RBE creates and manages an Emulated Business (EB) computer for each emulated Business. The term RBE, when used in this specification, includes the entire population of EB.

**6.1.3** The RBE communicates with the SUT using TCP/IP sockets.

**6.1.4** The RBE must use secure communication for all SUT web service interactions (see clause 2.1.1).

**6.1.5** The RBE is responsible for the following:

- Conforming to all execution rules specified in this document
- Conforming to all industry standards specified in this document, e.g. SSL/TLS
- Generating random numbers, timestamps, strings as required to implement the benchmark
- Selection of interaction types for EB's.
- Recording the counts and percentages of the mix of web service interactions requested and completed.
- Recording SIRT and throughput in accordance with the requirements of Clause 5

**6.1.6** The RBE may not perform any processing ordinarily performed by the SUT. This includes, but is not limited to:

- Caching Web objects, files or database tables, except as specifically noted in Clause 2, as required to execute the RBE functions.
- Searching files or databases except as required to execute RBE functions

- Caching disk addresses or pointers to database records on the SUT
- Performing computations such as shipping cost, tax, etc. that belong to the SUT
- Communicating information to the SUT regarding future web interactions or navigation
- Executing active elements (e.g., Java scripts or applets) communicated by the SUT
- Any load balancing or routing across multiple components of the SUT

**6.1.7** The RBE may not use proprietary communication protocols when communicating with the SUT. The RBE is required to use TCP/IP sockets (i.e. RFC 1122, etc.) for all network communication. A new socket must be created for each Business Session. Business Sessions cannot share a socket connection. However, the RBE may open multiple socket connections per Business Session.

The SUT and RBE must open a new socket connection for each EB. A socket connection cannot be used by more than one EB. Furthermore, if the SUT allows persistent sessions, also known as keep-alive (as specified in HTTP 1.1), the duration parameters for persistent sessions (i.e. time limit and request limit) must be disclosed.

**6.1.8** The communication between the EB's and the SUT must not use a TCP Maximum Segment Size (MSS) greater than 1,460 bytes.

**6.1.9** The value of TCP TIME_WAIT must be at least 60 seconds.

**6.1.10** Require all EB's to use the identical URL to request the Web Service interactions. The only part of the URL that can change is the parameter that uniquely identifies the customer.

**Comment:** It is realized that dynamic DNS caches will allow the RBE's to cache the IP address and this is believed to have no significant performance advantage.

**6.1.11** During the Test Run an EB must not introduce any artificial delay.

## 6.2 Business Session Length

**6.2.1** The **Business Session Length (BSL)** is the number of Web Service Interactions that are performed by the EB during a Business Session.The EB generates random BSLs in accordance with the discrete distribution defined by the BSL Cumulative Distribution Function (CDF) in Appendix C - .

**Comment 1**: In generating its BSLs, the EB must use an independent random number generator. (See Clause 2.1.14.)

**Comment 2:** Appendix D - describes how to obtain source for code that may be used to generate the BSLs. It is recommended, but not required, that this code be used. However, if this code is not used, or if a modification or augmentation of this code is used, then the code used to generate BSLs must disclosed in the FDR.

**6.2.2** For a given Measurement Interval, the average BSL is computed as the average of the BSL for all sessions beginning during that Measurement Interval. If the session does not end before the close of the Measurement Interval, the BSL calculated for that session should still be included in the average BSL.

**6.2.3** Customer IDs, Business Name, and Customer Password

***6.2.3.1*** The RBE chooses the web service interaction to be performed based upon the mix specified by clause 5.1. Each EB will choose the starting C_ID at the beginning of each Business Session. During the Business Session this C_ID will be used for the Change Payment Method, New Order and Check Status web interactions. The EB must maintain and use this value for the C_ID for all subsequent web service requests until an Add Customer interaction is chosen. After an Add Customer interaction, the EB must retain and use this new C_ID for subsequent interactions. The method for choosing the initial C_ID is defined below:

The C_ID must be chosen using the NURand function defined in clause 1.2.45 with the following parameters:

- X = 1
- Y = NUM_CUSTOMERS where NUM_CUSTOMERS is the initial cardinality of the CUSTOMER table.
- A is chosen from the following table:

**Comment**: It is possible that the EB would not use the C_ID that is generated due to the interaction types chosen.

| For NUM_CUSTOMERS in this range | Value for A |
|---|---|
| 1 – 9,999 | 1,023 |
| 10,000 – 39,999 | 4,095 |
| 40,000 – 159,999 | 16,383 |
| 160,000 – 639,999 | 65,535 |
| 640,000 – 2,559,999 | 262,143 |
| 2,560,000 – 10,239,999 | 1,048,575 |
| 10,240,000 – 40,959,999 | 4,194,303 |
| 40,960,000 – 163,839,999 | 16,777,215 |
| 163,840,000 – 635,359,999 | 67,108,863 |

***6.2.3.2*** After any New Customer interaction, the newly generated C_ID must be used until the end of the Business Session or until another New Customer interaction is performed, whichever occurs first.

***6.2.3.3*** The business name (C_BUSINESS_NAME) is generated as defined in clause 4.6.1 when the C_ID is chosen at random at the beginning of the business session and saved in the RBE with the chosen C_ID. When a New Customer Interaction is generated, the business name is generated as defined in clause 2.2.2 and saved with the new C_ID until the end of the Business Session or until another New Customer interaction is performed which ever occurs first.

***6.2.3.4*** The customer password (C_PASSWD) is generated as defined in clause 4.6.2 when the C_ID is chosen at random at the beginning of the business session and saved in the RBE with the chosen C_ID. When a New Customer Interaction is generated, the customer password is generated as defined in clause 2.2.2 and saved with the new C_ID until the end of the Business Session or until another new Customer interaction is performed which ever occurs first.

**6.2.4** Business Session Termination

***6.2.4.1*** A Business Session ends when the EB has completed the BSL number of Web Service Interactions.

***6.2.4.2*** When the Business Session ends, the EB takes the following actions:

- The EB closes any SSL/TLS session that is currently established for the Business Session.
- The EB closes all network connections that are currently established for the Business Session.
- After taking these actions, the EB is allowed to immediately start a new Business Session.

## 6.3 Random Number Generation

The RBE, the SUT, and the database loader must use the Marsaglia's multiply-with-carry generator (described in Appendix A - ):

```
#define znew ((z=A*(z&65535)+(z>>16))<<16)
#define wnew ((w=B*(w&65535)+(w>>16))&65535)
#define IUNI (znew+wnew)
#define UNI (znew+wnew)*2.32830643708e-10
```

where z and w are static 32 bit unsigned integers that are initialized to unique non-zero values across all Ebs. The term "start seeds" refers to this unique combination of z and w.

The two constants A and B must be chosen as any pair of distinct constants from this list:
18000 18030 18273 18513 18879 19074 19098 19164 19215 19584
19599 19950 20088 20508 20544 20664 20814 20970 21153 21243
21423 21723 21954 22125 22188 22293 22860 22938 22965 22974
23109 23124 23163 23208 23508 23520 23553 23658 23865 24114
24219 24660 24699 24864 24948 25023 25308 25443 26004 26088
26154 26550 26679 26838 27183 27258 27753 27795 27810 27834
27960 28320 28380 28689 28710 28794 28854 28959 28980 29013
29379 29889 30135 30345 30459 30714 30903 30963 31059 31083

## 6.4 System Under Test (SUT)

The SUT comprises all components which are part of the "application" being simulated. This includes network connections, the application server and database server.

### 6.4.1 SUT Contents

**SUT Components**

The SUT may consist of at most two SYSTEMs. No external devices or external appliances may be used to perform any functions of the SUT. All network connection management, including SSL/TLS processing, must take place inside the SYSTEM(s) that make up the SUT.

There is no limitation on the type or number of cards that may be connected to the internal bus of a SYSTEM (e.g. PCI bus). When the SUT contains only a single SYSTEM all SUT functions must be performed on that SYSTEM. When a SUT contains two SYSTEMs then the first SYSTEM may host only the DBMS and any components that are required to implement the ACID properties of the database and distributed transactions. The SYSTEM that hosts the DBMS may not perform any XML serialization of the output response that is returned to the EB. The second SYSTEM must host the application server(s) and must perform all other functions of the SUT. All business logic (as defined in the processing sections of clause 2) must be performed on the SYSTEM that hosts the application server(s).

**Comment 1:** This clause does not preclude the use of external disk storage devices, however those devices must only be used for persistent data storage and durability. The intent is to prohibit the use of SAN technologies from offloading TPC-W workload requirements.

The network equipment that connects the SUT to the EBs may not perform any functions other than connectivity. This network equipment is considered to be outside the SUT.

The functions that this network equipment may not perform include but are not limited to the following:

- Load balancing
- SSL/TLS processing
- Connection management
- XML serialization or de-serialization
- Communication between SUT components

**Comment 2:**

The use of external devices is limited in an effort to control the horizontal scaling of the benchmark.

**Comment 3:**

A typical customer would offload work from the DBMS to the greatest extent possible. This would permit a more scalable solution (scale out)

**6.4.2** Functions of the SUT

***6.4.2.1*** The SUT services the following:

- SOAP requests via the HTTP protocol (adhering to the WS-I Basic Profile 1.0 Specification), by returning the output requirements in a web service response (see clause 1.2.32).

***6.4.2.2*** The SUT performs the following operations:

- All database accesses. The database may be accessed using any commercially available interface.
- All communication functions to the EB, POV, PGE and ICE. These communications must adhere to the WS-I Basic Profile 1.0 Specification.
- All application functionality required to implement the web service interactions.

**6.4.3** SUT Restrictions

***6.4.3.1*** All caching of durable data must maintain full atomicity, isolation, and consistency (ACI). The test sponsor cannot maintain these properties via user written code or code modified by the user.

The intent is to disallow application server caching where the application program takes advantage of the known size of the tested database or the known size of the tested configuration. For example, taking advantage of the fact that the benchmark requires the use of a single application server is not allowed. The application program must be capable of running on multiple app servers in parallel. However, the benchmark does allow for a configuration where the database and the application server are combined into a single product.

Caching operations performed by the SUT are restricted to the commercially available components of the SUT.

***6.4.3.2*** Operations to insert, update, or remove items from a cache cannot be performed by the user written or modified code.

***6.4.3.3*** If a programmatic interface is required by the caching software, this must be part of a commercial product. Insertion or modification of API calls by the user for this purpose in the application program is prohibited.

***6.4.3.4*** The operations performed by user written or modified code during a given web service request may not perform any step of the processing definition on behalf of any other web service request.

**Comment:** This is to prevent user code from forwarding data from one request to another.

## 6.5 Purchase Order Validation (POV)

**6.5.1** The Purchase Order Validation (POV) represents an external system which authorizes credit for a new customer CREDIT_INFO.

**6.5.2** The POV is not included in the SUT.

**6.5.3** The POV must perform the following functions:

- Establish an SSL/TLS session at the SUT's request
- Receive properly formed SOAP messages from the SUT
- For each encrypted SOAP message received from the SUT:
- Decrypt the message. The message must contain the following fields:
  BUSINESS_NAME
  CREDIT_INFO
- Generate an authorization code, AUTH_ID, as a unique a-string of 16 characters
- Record the BUSINESS_NAME and AUTH_ID on a durable medium
- Encrypt the updated SOAP response message with the authorization code. The SOAP response message must contain the field AUTH_ID.
- Send the encrypted SOAP response message back to the SUT

Each SSL/TLS response must be sent separately.

**6.5.4** The response time between the reception of a given message from the SUT and the communication of the POV's response message to the SUT must be no less than 2 seconds.

## 6.6 Payment Gateway Emulator (PGE)

**6.6.1** The Payment Gateway Emulator (PGE) represents an external system which authorizes payment of funds as part of the New Order Web Services interaction.

**6.6.2** The PGE is not included in the SUT.

**6.6.3** The PGE must perform the following functions:

- Establish an SSL/TLS session at the SUT's request
- Receive properly formed SOAP messages from the SUT
- For each encrypted SOAP message received from the SUT:
- Decrypt the message
- Extract the CC_NAME, CC_NUMBER, and CC_EXPIRY from the PGE request message
- Generate an authorization code, AUTH_ID, as a unique a-string of 16 characters
- Record the O_ID and AUTH_ID on a durable medium
- Encrypt the updated SOAP response message. The response message must be a properly formed SOAP response containing the AUTH_ID field.
- Send the encrypted SOAP response message back to the SUT

Each SSL/TLS response must be sent separately.

**6.6.4** The response time between the reception of a given message from the SUT and the communication of the PGE's response message to the SUT must be no less than 2 seconds.

## 6.7 Inventory Control Emulator (ICE)

**6.7.1** The Inventory Control Emulator (ICE) represents an external system which receives requests for additional item stock.

**6.7.2** The ICE is not included in the SUT.

**6.7.3** The ICE must perform the following functions:

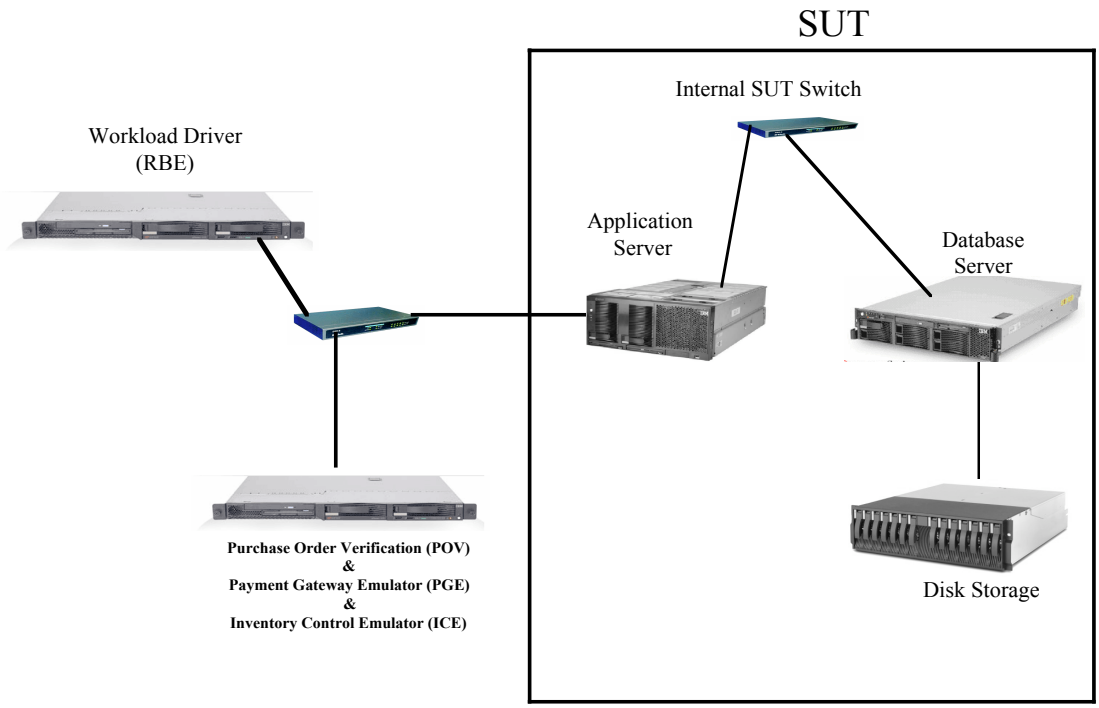
- Establish an SSL/TLS session at the SUT's request
- Receive properly formed SOAP messages from the SUT
- For each SOAP message received from the SUT:
- Decrypt the message
- The message consists of the I_ID, S_QTY quantity, and O_ID.
- Record the I_ID, S_QTY, and O_ID on a durable medium.
- Send the encrypted SOAP success response message back to the SUT.

Each SSL/TLS response must be sent separately.

**6.7.4** The response time between the reception of a given message from the SUT and the communication of the ICE's response message to the SUT is not restricted in any way.

## 6.8 Model of the Complete Tested System

The following diagram shows an example layout of RBE, POV, PGE, ICE and SUT components. All components inside the box are part of the SUT and, as such, must be priced (see Clause 7).




## 6.9 Communications Interfaces Definitions

### 6.9.1 Network Protocol

The network protocol for communicating between the SUT and the RBE as well as between the SUT and the PGE, POV, and ICE must be TCP/IP. The network protocol for communication within the SUT is not restricted.

### 6.9.2 Application Protocol

The application protocol is the data-level protocol which is used between the RBE and the SUT and between the SUT and the external emulators (ICE, POV, and PGE). All protocols which are used must be commercially available. All web interactions must adhere to the WS-I Basic Profile 1.0 Specification. All web service interactions must use secure communications (see clause 2.1.1).

### 6.9.3 Communications within the SUT

The communications protocols within the SUT are not restricted.

## 6.10 Operational Characteristics of the Merchant Commerce Application

The objective of this benchmark is to represent a 7X24 operating environment for a business to business application environment. The following are requirements of the benchmark application that make it consistent with a 7X24 operational model. Some of these functional characteristics may not be actually exercised during the benchmark's Measurement Intervals.

### 6.10.1 Uninterrupted Execution

The benchmark application must be capable of continued execution for at least 8 hours with the measured load (see also clause 5.5).

To demonstrate continued execution capability, the following data must be collected for each system that is part of the SUT at the start and end of the test run for the reported Measurement Interval (see clause 1.2.21), prior to any shutdown of the application:

- Total disk space utilization in bytes
- Disk space for system swapping and paging

The total growth in these resources (RG) uses the following formula for each resource:

$$RG = IR + (G / TI * SIPS * 3600 * 8)$$

Where:

TI is the total interactions during the test run,
G is the growth over the test run, and
IR is the initial resource utilization;

To be considered a valid TPC-W result, the resource requirements for disk, as computed above, must not exceed the capacity of the system.

**Comment**: If the above requirement is implemented by backing-up some files, such as the Web Server Access Logs, to tapes at regular intervals, the storage space archived to tape can be excluded from the above storage requirement.

### 6.10.2 Non-Disruptive Maintenance

The following activities are considered indispensable for normal system maintenance and must be possible during the 8 hours of uninterrupted execution, as defined above:

- Site administration
- Security administration
- Offload or archive of system, database, application or Web Server logs
- Processing accounting information
- Creating Database Checkpoints
- Processing Database Image Copies

Site administration includes but is not limited to activities such as message queue maintenance, changing currency exchange rates, item descriptions, customer discounts, etc.

Security administration involves protecting resources such as data sets, databases, programs and operator commands from access by unauthorized users. It includes creating generic security profiles, changing permissions for users and resources, allowing the access, update and creation of data sets, and resetting passwords, if necessary.

Processing accounting information refers to the analysis and reporting of accounting data using a program or an accounting software package. Depending upon specific platform implementations, there may also be a requirement to dump or offload accounting records before the analysis program(s) can operate upon them.

Database checkpoints are snapshots of the database at some specific moment in time, for the purposes of establishing a consistent state which can be used for recovery, if necessary. Checkpoints normally occur at predetermined time intervals (e.g. every 25 minutes), when switching from one active log data set to another, after a successful restart or at normal termination.

Database Image Copies are normally created using a database utility program to capture a likeness of the data and/or control information. Image copies may be subsequently used in a recovery operation. Most DBMS support both full image copies (i.e. a dump of the entire database(s)) and incremental image copies (i.e., a copy of the information that has changed since the last image copy).

**Comment**: There is no specific requirement for a demonstration of non-disruptive maintenance, but the auditor has the option to require a demonstration of non-disruptive maintenance.

# Clause 7 - **PRICING**

## 7.1 Pricing Methodology

**7.1.1** The intent of this section is to define the methodology to be used in calculating the 3-year pricing and the price/performance (price/SIPS). The fundamental premise is that what is tested and/or emulated is priced and what is priced is tested and/or emulated. Exceptions to this premise are noted below.

**7.1.2** The proposed system to be priced is the aggregation of the SUT components that would be offered to achieve the reported performance level, as defined in Clause 6. Calculation of the priced system consists of:

Price of the SUT as tested and defined in Clause 6.

Price of on-line storage required for the initial database population and data generated over 30 8-hour days of processing at the reported SIPS, allowing for a minimum of 8 hours of uninterrupted processing.

The system software necessary to create, operate, administer, and maintain the application program.

Price of additional products that are required for the operation, administration, performance monitoring or maintenance of the priced system.

**7.1.3** The following pricing methodology must be used:

All hardware and software used in the pricing calculations must be announced and orderable by customers. For any product not already generally released, the Full Disclosure Report must include a committed general delivery date (see clause 8.9.3). That date must not exceed 6 months beyond the Full Disclosure Report submittal date.

**Comment:** Inclusion of a product in a TPC-W FDR constitutes public announcement of that product. Once the price of a product is quoted in a TPC-W FDR and once customers can use the source of the quote to initiate a transaction that will guarantee delivery of that product, it is considered orderable.

Generally available discounts for the priced configuration are permissible.

Generally available packaged pricing is acceptable.

Local retail pricing and discount structure should be used in each country for which results are published.

Price should be represented by the currency with which the customer would purchase the system.

All hardware components used in the priced system must be new (i.e., not reconditioned or previously owned).

For test sponsor(s) who have only indirect sales channels, pricing must be actual generally available pricing from indirect channels which meet all other requirements of Clause 7.

Prices should be shown as whole integer amounts (e.g., dollars but not cents). All fractional amounts should be rounded up to the nearest integer value.

**Comment 1:** The intent of the pricing methodology is to allow packaging and pricing that is generally available to customers and to explicitly exclude promotional and/or limited availability offerings.

**Comment 2:** Revenue discounts based on total price are permissible. Any discount must be only for the configuration being priced and cannot be based on past or future purchases. Individually negotiated discounts are not permitted. Special customer discounts (e.g. GSA schedule, educational schedule) are not permitted.

**Comment 3:** The intent is to benchmark the actual system which the customer would purchase. However, it is realized that vendors may announce new products and disclose benchmark results before the products have actually shipped. This is allowed, but any use of one-of-a-kind hardware/software configurations, which the vendor does not intend to ship in the future is specifically excluded. All products must be generally announced and orderable in the country where the SUT is priced.

**7.1.4** The test sponsor (s) must disclose all pricing sources and effective date(s) of the prices.

**7.1.5** The sponsor is required to state explicitly all the items and services which are not directly available through the sponsor. Each supplier's items and prices, including discounts, must be listed separately. Discounts may not be dependent on purchases from any other suppliers.

**7.1.6** Non-Sponsor Pricing

***7.1.6.1*** In the event that any hardware, software, or maintenance is provided by a vendor who is not a sponsor of the benchmark, the pricing must satisfy all requirements for general availability, standard volume discounts, and full disclosure. Furthermore, any pricing which is not directly offered by the test sponsor(s) and not derived from the non-sponsoring vendor's generally available pricing and discounts must be guaranteed by the vendor in a written price quotation for a period not less than 60 days from the date the results are submitted. This written quotation must be included in the Full Disclosure Report and state that the quoted prices are generally available, the time period for which the prices are valid, the basis of all discounts, and any terms and conditions which may apply to the quoted prices. The test sponsor must still comply with price changes as described in Clause 8.

***7.1.6.2*** For items provided by a vendor who is not a sponsor of the benchmark, and for which the aggregated price is less than 1% of the total price, the requirement for a written price quotation from the vendor can be replaced by including in the FDR a copy of the source document from which the price was obtained (e.g., printed advertisement or on-line catalogue page).

**7.1.7** Pricing shown in the Full Disclosure Report must reflect line item pricing for hardware, software, and maintenance from the vendor's price books. Line items must reflect quantity one pricing with details providing the basis of any discounts, whether applied to the line item or applied to overall dollar value from a vendor, disclosed separately.

**Comment**: The intent of this clause is that the pricing reflect the level of detail that an actual customer would see on an itemized billing. The pricing excludes domestic taxes and shipping charges that would be incurred in the country for which the results are published. It is not intended to exclude tariffs, custom duties/fees, and shipping to a domestic port of entry if the component originates in another country.

**7.1.8** For publishing in another country other than the country for which the results are originally published, it is permitted to substitute local components from the original report providing the substituted products are sold to the same product description or specifications.

**Comment**: The intent of this clause is to encourage local country pricing by allowing substitution of equipment for country specific reasons such as voltage, product numbering, industrial/safety, keyboard differences, etc., which do not affect performance.

## 7.2 Priced System

**7.2.1** SUT

The entire price of the SUT as configured during the test must be used, including all hardware (new purchase price), software (license charges) and hardware/software maintenance charges over a period of 3 years (36 months). In the case where the Driver System provides functionality in addition to the RBE described in Clause 6, then the price of the emulated hardware/software described in clause 7.2.2.1 are to be included.

**Comment 1**: The intent is to price the tested system at the full price a customer would pay. Specifically prohibited are the assumptions of other purchases, other sites with similar systems, or any other assumption which relies on the principle that the customer has made any other purchase from the vendor. This is a one time, stand-alone purchase.

**Comment 2**: The number of users for TPC-W is defined to be equal to the number of business emulated in the tested configuration (EBs). Any usage pricing for the above number of clients should be based on the pricing policy of the company supplying the priced component.

**7.2.2** Client Business and Network Pricing

***7.2.2.1*** The price of the Remote Business Emulator (RBE), the Payment Gateway Emulator (PGE), the Inventory Control Emulator (ICE), and the Credit Check Emulator (CCE) are not included in the pricing calculation. Please refer to clause 6.4.1 for a description of the components of the SUT, all of which must be priced.

**7.2.3** Database Storage and Recovery Log Pricing

***7.2.3.1*** Within the priced system, there must be sufficient on-line storage to support any expanding system files, excluding database recovery log (see clause 4.5.3), and the durable database population resulting from executing the TPC-W transaction mix for 60 eight hour days at the reported SIPS (see clause 4.4), allowing for a minimum of 8 hours of uninterrupted processing. Tape is not considered on-line storage. On-line storage may include magnetic disks, optical disks, or any combination of these.

**Comment 1**: The intent of this clause is to consider as on-line any storage device capable of providing an access time to data, for random read or update, even if this access time requires the creation of a logical access path not present in the tested database.

***7.2.3.2*** It is permissible to not have the storage required for the 60-day space on the tested system. However, any additional storage device included in the priced system but not configured on the tested system must be of the type(s) actually used during the test and must satisfy normal system configuration rules.

**Comment**: Storage devices are considered to be of the same type if they are identical in all aspects of their product description and technical specifications.

7.2.3.3 The database recovery log must be configured and priced to include at least 8 hours of on-line storage. (See clause 4.5.3 for database log requirements.)

**7.2.4** Additional Operational Components

***7.2.4.1*** Additional products that might be included on a customer installed configuration, such as operator consoles and magnetic tape drives, are also to be included in the priced system if explicitly required for the operation, administration, or maintenance, of the priced system.

***7.2.4.2*** Copies of the software, on appropriate media, and a software load device, if required for initial load or maintenance updates, must be included.

***7.2.4.3*** The price of an Uninterruptible Power Supply, specifically contributing to a durability solution, must be included (see clause 3.1.5).

***7.2.4.4*** The price of all cables used to connect components of the system must be included.

**7.2.5** Additional Software

***7.2.5.1*** The priced system must include the software licenses necessary to create, compile, link, and execute this benchmark application as well as all run-time licenses required to execute on the SUT.

In the event that the application program development is required to occur on a system other than the SUT, the price of that system and any compilers and other software used must also be included as part of the priced system. Additional software that exists on the SUT which is not used during the Test Run does not need to be part of the priced system.

## 7.3 Maintenance

**7.3.1** Hardware and software maintenance must be figured at a standard pricing which covers at least 7 days/week, 24 hours/day coverage, either on-site, or if available as standard offering, via a central support facility. Hardware maintenance maximum response time must not exceed 4 hours, on any part whose replacement is necessary for the resumption of operation. The 4-hour maintenance response time requirement may be met by customer spareable and replaceable hardware, as described in clause 7.3.4.

**Comment 1**: Resumption of operation means the priced system must be returned to the same configuration that was present before the failure.

**Comment 2**: The intent of hardware maintenance pricing is not met by pricing based on the cost to fix specific failures, even if the failure rate is calculated from Mean Time Between Failure (MTBF). The maintenance pricing must be independent of actual failure rates over the 3 year period, no matter how many failures occur during that period. The intent is to preclude the use of MTBF to directly compute the maintenance cost for this benchmark.

**7.3.2** If central support is claimed, then the appropriate connection device, such as auto-dial modem must be included in the hardware price. Also any software required to run the connection to the central support, as well as any diagnostic software which the central support facility requires to be resident on the tested system, must not only be included in pricing, but must also be installed during the benchmark runs.

**7.3.3** Software maintenance must include maintenance update distribution for both the software and its documentation. If software maintenance updates are separately priced, then pricing must include at least 2 updates over the 3 year period.

**Comment**: Software maintenance, as defined above, means a standard offering which includes acknowledgment of new and existing problems within 4 hours and a commitment to fix defects within a reasonable time.

**7.3.4** It is acceptable to incorporate, for pricing purposes, the use of customer spareable and replaceable hardware items under the following conditions:

An additional 10% of the number of configured units of the replaceable items, with a minimum of 2, must be priced for spares.

The vendor must include a support service which guaranties replenishment on-site within 7 days throughout the 3 year maintenance period and covers every unit of the replaceable items (i.e., configured units and spare units).

The items must be generally available as spareable and replaceable for any customer installation.

The designation of the items as spareable and replaceable cannot depend on a threshold of purchased quantity.

It must be verifiable that the action of diagnosing the spareable and replaceable items as having failed can be positively accomplished by the customer within 4 hours of failure.

The method for diagnosis and replacement of the replaceable items must have complete customer documentation.

**Comment 1:** Diagnosis may take the form of a hardware indicator or a diagnosis procedure. The intent is that the diagnosis must reach a positive conclusion as to the state of the item within 4 hours.

**Comment 2:** The use of spares is intended to assist in complying with the 4-hour maximum hardware maintenance response requirement. It cannot be a substitute for maintenance support, as the priced configuration must maintain the same quantities of components, including spares, for 3 years.

## 7.4 Required Reporting

**7.4.1** Two metrics will be reported with regard to pricing. The first is the total 3 year pricing as described in the previous clauses. The second is the total 3 year pricing divided by the reported Service Interactions Per Second (SIPS), as defined in clause 5.3.

**7.4.2** The 3-year pricing metric must be fully reported in the basic monetary unit of the local currency (see clause 7.1.3) rounded up and the price/performance metric must be reported to a minimum precision of three significant digits rounded up. Neither metric may be interpolated or extrapolated. For example, if the total price is US$ 5,734,417.89 and the reported throughput is 105 SIPS, then the 3 year pricing is US$ 5,734,418 and the price/performance is US$ 54,700/SIPS (5,734,418/105).

# Clause 8 - **Clause 8 Full Disclosure Report**

## 8.1 General Requirements

**8.1.1** A Full Disclosure report is required in order for results to be considered compliant with the TPC-W benchmark specification. All reported metrics must be compliant with the requirements in the TPC-W specification to be considered a valid TPC-W result

**Comment:** The intent of this disclosure is for a customer to be able to replicate the results of this benchmark given the appropriate documentation and products. This section includes a list of requirements for the Full Disclosure report.

**8.1.2** The order and titles of sections in the Test Sponsor's Full Disclosure report must correspond with the order and titles of sections from the TPC-W standard specification (i.e., this document). The intent is to make it as easy as possible for readers to compare and contrast material in different Full Disclosure reports. All sections of the report, except the appendices, must be printed using a minimum font size of 10 points. The appendices must be printed using a minimum font size of 8 points.

**8.1.3** The TPC Executive Summary Statement must be included as the first pages of the Full Disclosure report. An example of the Executive Summary Statement is presented in Appendix F - . The latest version of the required format is available from the TPC Administrator.

**8.1.4** A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

**8.1.5** The numerical quantities listed below must be included in the Executive Summary detailing the following quantities In tabular form the following information for the Measurement Interval:

For each service Interaction
- 90th percentile response time
- Mean response time
- Interaction count
- Mix percentage

**Comment: Appendix F -** contains an example of such a summary. The intent is for data to be conveniently and easily accessible in a familiar arrangement and style. It is not required to precisely mimic the layout shown in **Appendix F -** .

**8.1.6** Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including but not limited to:

- Database tuning parameters.
- Web server tuning and logging options.
- Application server tuning and logging
- Driver Parameters
- Configuration files
- Recovery/commit options.
- Consistency/locking options.
- Operating system and application configuration parameters.
- Compilation and linkage options and run-time optimizations used to create/install/run applications, OS, DBMS, web server, and/or any other commercial product.

**Comment 1:** This requirement can be satisfied by providing a full list of all parameters, options, and flags.

**Comment 2:** The intent of the above clause is that anyone attempting to recreate the SUT has sufficient information to compile, link, optimize, and execute all software used on the SUT.

**Comment 3**: This clause only applies to the SUT and not to components that are outside of the SUT (ICE,PGE,POV,RBE etc)

**8.1.7** Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:
- Number and type of processors.
- Physical memory actually present on the SUT.
- Size of allocated memory, and any specific mapping/partitioning of memory unique to the test.
- Number, type, brand and model of disk units and controllers.
- Number of channels or bus connections to disk units, including their protocol type, brand and model.
- Number, brand and model of LAN (e.g., Ethernet) equipment, including routers, switches, NICS, etc., that were physically used in the test or are incorporated into the pricing structure.
- Type and the run-time execution location of software components (e.g., DBMS, web server, application server or program, transaction monitors, etc.)

- Number, type, brand and model of cryptographic processors or cryptographic accelerators, if applicable.

**Comment:** Detailed diagrams for system configurations and architectures can widely vary, and it is impossible to provide exact guidelines suitable for all implementations. The intent here is to describe the system components and connections in sufficient detail to allow independent reconstruction of the measurement environment.

## 8.2 Executive Summary

The Executive Summary is meant to be a high level overview of a TPC-W implementation. It should provide the salient characteristics of a benchmark execution (metrics, configuration, pricing, etc.) without the exhaustive detail found in the FDR. The Executive Summary has three components:

- Overview Page
- Pricing Page
- Numerical Quantities Summary Page

### 8.2.1 Page Layout

Each component of the Executive Summary should appear on a page by itself. Each page should use a standard header and format, including:

- 1/2 inch margins, top and bottom;
- 3/4 inch left margin, 1/2 inch right margin;
- At least 2 pt. frame around the body of the page. All interior lines should be 1 pt except for the horizontal line that precedes the software descriptions (before Web Server) which is 3 pt.;
- Sponsor identification and System(s) name(s), each set apart by a 1 pt. rule, in 16-20 pt. bold Times font;
- Benchmark name (i.e., TPC-W), revision using three tier versioning (e.g., 1.2.3) and report date, separated from other header items and each other by a 1 pt. Rule, in 9-12 pt. bold Times font;
- The two primary metrics:
- The Price Performance in 12-14 pt. bold Times font.
- The Performance in 12-14 pt. bold Times font.
- The Availability date in 10-12 pt bold Times font.

**Comment 1:** It is permissible to use or include company logos when identifying the sponsor.

**Comment 2:** The report date and availability date must be disclosed with a precision of 1 day. The precise format is left to the test sponsor.

### 8.2.2 Overview Page

**Appendix F -** contains a sample Executive Summary. The sample specifies the mandatory format and layout of the overview page. The sample for the pricing page and numerical quantities summary page are included to help clarify the requirements in Clause 8.2 and are provided solely as examples.

The overview page contains 4 sets of data, each laid out across the page as a sequence of boxes using 1 pt. rule, with a title above the required quantity. Both titles and quantities should use a 9-12 pt. Times font unless otherwise noted.

***8.2.2.1*** The first section contains the following metrics that were obtained from the reported Test Runs in a 10-12 pt Time font:

**Table 1: Executive Summary Metrics**

| Title (in Bold) | Quantity | Precision | Units |
|---|---|---|---|
| Total System Cost | 3 Year Cost of Ownership | 1 | $[1] |
| Performance | Service Interactions per Second | .1 | SIPS |
| Price Performance | Price per SIP | .1 | $[1] |
| Users | Number of active EBs | 1 | Sessions |
| Response Time | Average Response Time of all Web Service Interactions | .01 | Seconds |
| Number of Systems | Number of Systems | 1 | Systems |

1) Pricing should be reported in the currency of the country where the system is priced.

***8.2.2.2*** The second section details the system configuration in 10-12 pt Time font

**Table 2: System Configuration Overview**

| Primary Function (in Bold) | Description |
|---|---|
| Application Server Brand | Software Version of Application Server |
| Database Manager Brand | Software Version of DBMS used |
| Managed Runtime Environment Brand | Software Version of Product Used for Runtime Environment |
| Distributed Transaction Manager Brand | Software Version of Product Used for Management of Distributed Transactions |
| Message Queueing Software Brand | Software Version of Message Queuing Software |

| Other Software Brand | Software Version of Other Products |
|---|---|

***8.2.2.3*** The third section contains a configuration diagram of the priced system.

***8.2.2.4*** The fourth and final section of the Implementation Overview contains a synopsis of the SUT's major system components, including for each system in the configuration:

- System's primary function as described in 8.2.2.2 (e.g. Database Server, Application Server, etc.);
- System brand name and type (model)
- Operating system
- Processor type and model, count and speed in MHz;
- Main memory sizes;

**8.2.3** Pricing Page

The pricing spreadsheet required by Section 7 must be reproduced in its entirety on the Pricing Page. It must include a section for each tier or system that performs a different function. Each section must include the hardware and software for that function. Refer to **Appendix F -** for a sample pricing spreadsheet.

**8.2.4** Numerical Quantities Summary Page

The Numerical Quantities Summary Page contains three sections used to summarize quantities in a concise manner. It is not intended to be an exhaustive collection of all reported numbers, but rather a high level overview of some of the more pertinent reported metrics.

***8.2.4.1*** The first section contains the SIPS Interaction Summary with the following information for each interaction type

- Interaction Name
- Interaction Rate in Interactions per Second
- Mix Percentage
- Average Response Time
- 90th Percentile Response Time
- Maximum Response Time

The second section contains a graph of the measured throughput versus elapsed time (i.e., wall clock time) for the entire Test Run. The x-axis represents the elapsed time from the start of the run. The y-axis represents the throughput in SIPS. A maximum interval size of 30 seconds may be used. A sample chart is shown in the sample Executive Summary in **Appendix F -** . The following information must be marked on the chart:

- Ramp-up time end.
- Stabilization period end
- The beginning and end of the measurement interval.

## 8.3 Clause 1 - Web Object and Logical Database Design

**8.3.1** Listings must be provided for all table definition statements and all other statements used to set-up the database. Enough detail must be provided to replicate the database structures.

**8.3.2** The physical organization of tables and indices, within the database, must be disclosed. When logical disk arrays (for example, RAID) are used, the mapping of database entities (for example, tables, views) must be specified. Additionally the mapping of logical units to physical disks must be disclosed.

**Comment:** The concept of physical organization includes, but is not limited to: record clustering (i.e., rows from different logical tables are co-located on the same physical data page), index clustering (i.e., rows and leaf nodes of an index to these rows are co-located on the same physical data page), and partial fill-factors (i.e., physical data pages are left partially empty even though additional rows are available to fill them).

**8.3.3** Any horizontal or vertical partitioning of tables or rows in the TPC-W benchmark must be disclosed (see clause 1.5.3 and 1.5.4). Replication of tables, if used, must be disclosed (see clause 1.5.5).

**8.3.4** Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

## 8.4 Clause 2 - Service Interactions and Workload

**8.4.1** The number of EBs configured must be disclosed.

**8.4.2** A description of how the security requirements were met as defined in clause 2.1.1, must be disclosed:
- Protocol and version
- Cipher
- Cipher strength
- Hash
- Hash strength
- Key exchange
- Key exchange strength

**8.4.3** The request and response XML for one representative Web Service Interaction for each interaction type must be included in the Full Disclosure Report.

**8.4.4** The XML for one representative entry that is placed on the Stock Management and Shipping process queues must be disclosed.

**8.4.5** A statement must be provided describing the development language(s), IDE and the types of API's used between commercial components to implement the interactions. This includes, but is not limited to, the interfaces to the database server, web server, commerce package or application, or any other commercial product used. Changes to the IDE parameters that effect the compilation, linking or execution of the application must be disclosed.

**8.4.6** The results of the WS-I compliance test must be disclosed.

**8.4.7** The method that the auditor used to determine "steady state" must be reported.

**8.4.8** The source code of the application program or programs as defined in clause 1.2.20 as well as all input for product generated code, must be disclosed.

## 8.5 Clause 3 - Transaction and System Properties

**8.5.1** The results of the Atomicity tests described in clause 3.1.2 must be disclosed along with a description of how the Atomicity requirements were met.

**8.5.2** The results of the Consistency tests described in clause 3.1.3 must be disclosed along with a description of how the Consistency requirements were met.

**8.5.3** The results of the Isolation tests described in clause 3.1.4 must be disclosed along with a description of how the Isolation requirements were met.

**8.5.4** The results of the Durability tests described in clause 3.1.5 must be disclosed along with a description of how the Durability requirements were met.

**Comment:** The order of the tests should be disclosed. Additionally, how the tests were combined must be disclosed.

## 8.6 Clause 4 - Scaling and Database Population

**8.6.1** The cardinality (e.g., number of rows) of each table, as initially populated (see clause 4.2 & 4.3), must be disclosed.

**8.6.2** The space required to sustain 60 days of the reported throughput as defined in clause 4.4 must be disclosed.

**8.6.3** The space required for 8 hours of log space as defined in clause 4.5 must be disclosed. This includes but is not limited to the following logs:
- Web Server
- Stock Management
- Database log

**8.6.4** The method for distributing table and log data across all media must be described. A detailed diagram or listing of database files indicating the disks or volumes on which they reside must be included. Simple diagrams can be used for clarification. The logs to which this clause refers include but are not limited to:
- Web Server
- Database log
- Stock Management log
- Distributed Transaction Manager log

**Comment:** Detailed diagrams for layout of database files on disks can widely vary, and it is difficult to provide exact guideline suitable for all implementations. The intent is to provide sufficient detail to allow independent reconstruction of the test database and access logs. A

script that was used to create the log files would be sufficient to satisfy the requirements of this clause.

**8.6.5** The method used by the auditor to verify that the database population meets the requirements in clauses 4.7 (table population) must be disclosed. This could include code reviews of the loader as well as SQL scripts run by the auditor to determine database validity. This description should be provided by the auditor.

**8.6.6** The source code for the database loader must be disclosed.

## 8.7 Clause 5 Performance Metrics and Response Times

**8.7.1** The SIPS and price per SIPS must be disclosed.

**8.7.2** The duration of the Measurement Interval must be disclosed.

**8.7.3** The following statistics must be reported for the web service interactions executed during the reported measurement interval. They must be displayed in tabular form as in the following example. Percentages must be reported to 3 decimal places.

- Mix Percentage
- Minimum Response Time
- Maximum Response Time
- 90th Percentile Response Time
- Average Response Time

**Table 3: Workload Statistics**

| Web Service | Mix Percentage | Min Response Time | Max Response Time | 90th Percentile Response Time | Av Res T |
|---|---|---|---|---|---|
| New Customer | 1.00% | 1.34 | 1.65 | 1.60 | 1.55 |
| Change Payment Method | 5.00% | 1.50 | 1.99 | 1.93 | 1.88 |
| Create Order | 50.00% | 2.03 | 2.66 | 2.60 | 2.50 |
| Order Status | 20.00% | .33 | .43 | .50 | .40 |
| New Products List | 10.00% | .42 | .66 | .63 | .55 |
| Order Status | 20.00% | .78 | .97 | .83 | .85 |
| Order Status | 20.00% | .56 | .78 | .75 | .66 |

**8.7.4** A list of the start times and duration of all checkpoints that are fully contained within the Measurement Interval must be disclosed.

**Comment:** Some DBMS products do not use checkpoints as described in clause 5.4.3. If the DBMS product in use does not use the checkpoint mechanism then the method used to meet the requirements in clause 5.4.2.2 must be disclosed.

**8.7.5** The percentage of service requests to the Create Order Web Service that contained a change of address as part of the input must be reported

**8.7.6** The number of sessions that were contained within the measurement interval must be reported. To be counted a session must begin and end within the measurement interval.

**8.7.7** Shipping Queue performance during the measurement interval

- The number of items put on the shipping queue during the Measurement Interval by the Create Order web Service
- The number of items that were put on the shipping queue during the Measurement Interval that were removed from the shipping queue by the Shipping Process during the Measurement Interval.
- The percentage of items processed during the measurement interval. (Items put on the queue during the Measurement Interval/ Items put on the queue during the Measurement Interval that were removed from the queue during the Measurement Interval) * 100 (see clause 5.5).
- Number of items placed on the shipping queue that had an O_STATUS of "BACK_ORDEDERED"
- The distribution of queue response times times as measured by the difference between the time that the shipping process completes its processing and the SHIPPING_REQUEST_TIME. This distribution must be reported with a maximum granularity of 0.1 seconds.

**8.7.8** Stock Management Queue performance during the measurement interval

- The number of items put on the stock management queue during the Measurement Interval by the shipping process
- The number of items put on the stock management queue during the Measurement Interval that were removed from the stock management queue by the Stock Management Process during the Measurement Interval
- The percentage of items processed during the measurement interval. (Items put on the queue during the Measurement Interval/ Items put on the queue during the Measurement Interval that were removed from the queue during the Measurement Interval) * 100 (see clause 5.5).

**8.7.9** The following metrics must be reported for the ICE activity during the Measurement Interval:

- Total Requests to the ICE
- Number of errors that occurred
- Percentage of ICE requests that resulted in an error (see clause 5.5).

**8.7.10** The percentage of New Customer web service requests that chose PO as the payment method (see clause 5.5).

**8.7.11** The percentage of Change Payment web service requests that specify PO as the payment method (see clause 5.5)

**8.7.12** The percentage of Create Order web service requests that included a shipping address change (see clause 5.5).

**8.7.13** The percentage of orders submitted to the Create Order web service that resulted in a backorder (see clause 5.5).

**8.7.14** The average number of Service Interactions in a user session must be disclosed.

**8.7.15 Performance Statistics**

*8.7.15.1* The monitoring tools used during each Measurement Interval must be described and the operational methods invoked must be reported. For example, the monitor was invoked as a started task, and binary data was recorded every 15 seconds to a disk for post-processing.

*8.7.15.2* If a sampling technique is used to obtain performance data then the sample rate (per second) must be reported. The data collection sampling interval is required to be a maximum of 1 second.

**Comment:** The intent of this clause is to report the sample rate for non-event-driven system monitor facilities. An example of this sample rate is found in determining "channelbusy", where the status of the channel must be sampled in order to determine if the resource is indeed busy. If the system monitoring facility is an event counter, it is not sampling.

*8.7.15.3* If there is no relevance or meaning to a specific performance statistic for the system and products used in this specific benchmark implementation, then the symbol (N/A) for not applicable may be used. This exclusion may not be used to avoid required reporting.

*8.7.15.4* The following performance statistics must be collected for both application server and the database server. Refer to the table in **Appendix F -** for specific details of reporting layouts and accuracy.

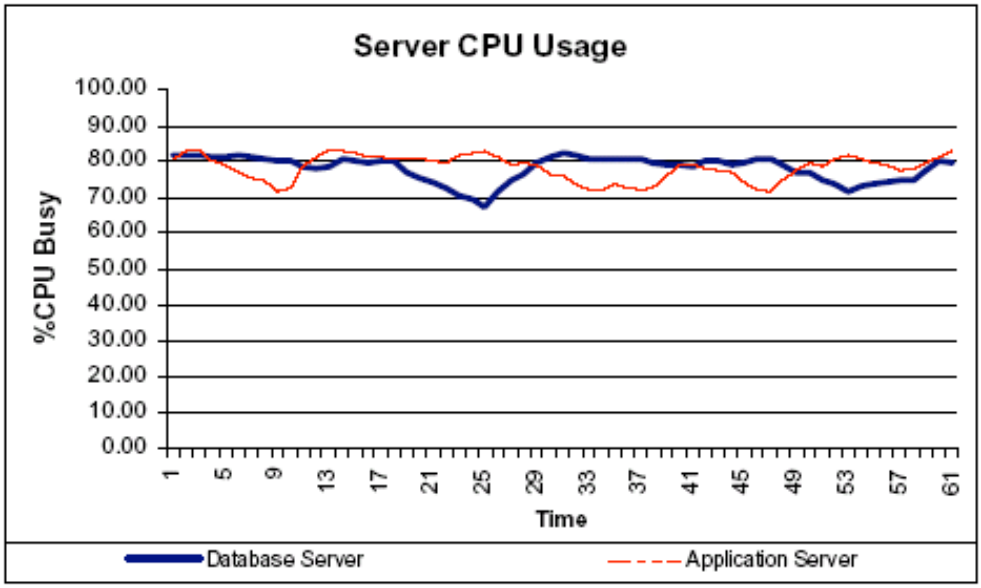
**8.7.15.4.1 CPU Utilization**

A key parameter of any capacity measurement is CPU utilization. All enterprise-class system monitoring facilities provide a means of recording CPU utilization.

**8.7.15.4.1.1** The required reporting method is to record CPU utilization, i.e. the percentage of time for which the CPU was busy.

**8.7.15.4.1.2** For the Measurement Interval, a graph illustrating CPU utilization for each system must be reported. The x-axis represents the elapsed time from the start of the Measurement Interval. The y-axis represents the CPU utilization on a scale of 0 to 100%. Discrete one minute intervals must be graphed across the Measurement Interval, where each value is the system's average CPU utilization over that one minute interval.

An example of such a graph is shown below:




#### 8.7.15.4.2 Memory Usage

The manner in which memory is allocated and controlled by the host operating system can indicate how efficiently the vendor operating system and/or DBMS supports the demands of the TPC-W benchmark. To further this understanding:

**8.7.15.4.2.1** The total amount of physical memory per SYSTEM must be reported.

**8.7.15.4.2.2** The total amount of memory used by the following software components.

- Memory used by the DBMS
- Memory used by the Web Server (including cache(s))

#### 8.7.15.4.3 System I/O Activity

**8.7.15.4.3.1** The average I/O rate, data transfer size and service time for each disk device as seen by the SYSTEM must be reported. Mapping of the logical structures, as seen by the operating system, to the physical disk devices must be reported. The total I/O rates may not be calculated as an average of the individual I/O rates, but must be calculated as the total number of I/O's divided by the length of the Measurement Interval in seconds.

**8.7.15.4.3.2** The average network I/O rate, data transfer size, and bytes per second for each network adapter must be reported. The total I/O rates may not be calculated as an average of the individual I/O rates, but must be calculated as the total number of I/O's divided by the length of the Measurement Interval in seconds.

#### 8.7.15.4.4 Application Server Statistics

**8.7.15.4.4.1** Connections per second - The number of connections requested by the RBE and accepted by the SUT per second. The intent is to count only the number of new connections made successfully by the RBE in generating the load for the benchmark. This statistic is not the number of concurrent connections the Web Server handles at any given point in time.

**8.7.15.4.4.2** Connections per second - The number of connections requested by the SUT to the external emulators (PGE,ICE, POV).

**Comment:** It is not necessary to separate these counts by emulator type.

**8.7.15.4.4.3** HTTPS requests per second from the RBE to the SUT.

**8.7.16** The monitored metrics must be reported in a table or tables similar to the following example:

| Metric | Application Server | Database Server |
|---|---|---|
| % CPU Busy | | |
| Memory in use (visible to OS) | | |
| System I/O Rate | | |
| Disk I/O Rate | | |
| Avg Disk Utilization | | |
| Avg Disk Service time | | |
| Avg Disk I/O Block Size | | |
| Total Logging I/O Rate | | |
| Additional data rows for other metrics | | |

A careful definition of the data fields reported must be provided to allow the data to be understood and properly interpreted.

**8.7.17** The sponsor must disclose information on the HTTP status of requests that occurred during the measurement interval. The following HTTP status information must be disclosed in a tabular form for each returned HTTP status code:

- Status Code
- Count

| HTTP Status | Count |
|---|---|
| 200 | 720000 |
| 404 | 20 |
| 501 | 5 |

## 8.8 Clause 6 - SUT, RBE and Network

**8.8.1** The rated bandwidth of the network(s) component used in the measured configuration must be disclosed along with any setting restricting that bandwidth. This includes, but is not limited to, the inter-node connections within the SUT, between the SUT and the RBE, and between the SUT and the PGE, POV, ICE.

**8.8.2** If the configuration requires operator intervention to meet the requirements of performance levels and uninterrupted operations, the mechanism and the frequency of this intervention must be disclosed.

**8.8.3** A count, by interaction type, of the retries that occurred during the measurement interval must be disclosed.

**8.8.4** The RBE method of generating the start seeds for the EBs must be disclosed. It must be clear that the start seeds are different for each EB. Additionally it must be clear that the start seeds are different from test run to test run.

**Comment:** It is the intent of this clause to prevent generating duplicate start seeds that will result in a reduction of inserts to the database (for example, ADDRESS table)

**8.8.5** The number of random number generators used for each EB must be disclosed along with their purpose(s). Examples of purposes include: choosing the next web service interaction, building random text strings, deciding whether the shipping address is changed, etc…

## 8.9 Clause 7 - Pricing

**8.9.1** A detailed list of hardware and software used in the priced system must be reported in the Executive Summary. Each separately orderable item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, vendor part number of the package and a description uniquely identifying each of the components of the package must be disclosed.

Pricing source(s) and effective date(s) of price(s) must also be reported. An example of the Executive Summary Statement is presented in **Appendix F -** .

**8.9.2** The total 3-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

**8.9.3** The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available. This single date must be reported on the first page of the Executive Summary. All availability dates, whether for individual components or for the SUT as a whole, must be disclosed to a precision of one day. See clause 8.12 for the requirements regarding revisions to the FDR.

**8.9.4** For any usage pricing, the sponsor must disclose:
- Usage level at which the component was priced.
- A statement of the company policy allowing such pricing.

**Comment:** Usage pricing may include, but is not limited to, the operating system and database management software. System pricing must include line item indication where non-sponsoring companies' brands are used. System pricing must also include line item indication of third party pricing.

## 8.10 Clause 9 - Audit Related Items

**8.10.1** The auditor's name, address, phone number and e-mail address must be included in the Full Disclosure Report.

**8.10.2** A copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report.

## 8.11 Availability of the Full Disclosure Report

**8.11.1** The Full Disclosure Report must be submitted to the TPC for publication on the TPCweb site prior to the results being made public.

**8.11.2** The official full disclosure report must be available in English but may be translated to additional languages.

## 8.12 Revisions to the Full Disclosure Report

**8.12.1 Required Revisions**

Revisions to the full disclosure documentation are required to be published under the following circumstances:

***8.12.1.1*** During the normal product life cycle problems will be uncovered that require changes, sometimes referred to as ECOs, FCOs, Patches, Updates, etc. When the cumulative result of applied changes causes the SIPS rating of the system to decrease by more than 2% from the reported SIPS, then the test sponsor is required to re-validate the benchmark results and publish the revised performance and price/performance.

***8.12.1.2*** When cumulative price changes have resulted in an increase of 2% or more from the reported price/performance, the test sponsor must submit revised price/performance results to the TPC within 30 days of the effective date of the price change(s) to remain in compliance. The benchmark need not be re-run to remain compliant.

**Comment:** The intent of this clause is that published price/performance reflect actual current price/performance.

***8.12.1.3*** Re-pricing of current results must be reviewed and approved by the auditor if there is a change to the pricing model. Changes in prices of line item components do not constitute a pricing model change.

***8.12.1.4*** Change in the committed availability date for the priced system which is later than the published availability date must published in a revised Full Disclosure Report.

***8.12.1.5*** All substitutions must be fully supported products. Additionally Hardware or Software product substitutions within the SUT, with the exceptions noted below require the benchmark to be re-run with the new components in order to re-establish compliance. The exceptions are:

- Secondary components such as switches, hubs, terminators and the like may be substituted. The connectivity capability of the substituted item must be greater than or equal to the original item. The substitute must be demonstrated to be at least equivalent to the original in performance. The performance tests and data used to demonstrate equal or greater performance must be disclosed.
- Disks may be substituted only under the following conditions:
- To substitute disk drives, the previous drives used in the SUT must have the same interface type and no longer be available. The substituted drives track to track seek time, interface speed, on disk buffer size, rotational speed, and media density must be at least the same.
- The number of new drives substituted for drives that are no longer available, must be greater than or equal to the number of drives in the published result.
- Disk HBA's may be substituted if the following conditions are met:
  - The interface type must be the same (SCSI,Fiber Channel etc)
  - The speed of the interface must be the same or greater.
  - The connectivity capacity of the HBA must be the same or greater.
  - The disk drive addressing capability must be the same or better.
- Software patches for any commercially available product must be for the same major and minor revision level. The cumulative effect of all patches must not degrade performance by more than 2%
- Memory substitution is allowed under the following conditions:
  - The same memory technology is used (PC100,PC133,DDR,RAMBUS, etc)
  - The memory latency specifications are at least as good of better
  - The memory capacity does not decrease
  - The number of memory modules in the system does not decrease
  - The memory has the same or better error checking capabilities
  - The memory has the same or better error correcting capabilities.
- No substitutions will be made for the following components:
  - CPU
  - Disk Controller
  - Network Interface Card

**Comment 1:** The intent is to allow substitutions when the change would produce performance at least equivalent to the reported SIPS. The auditor may require additional tests to be run if the proof by documentation is not considered adequate.

The auditor's letter of attestation must be attached to the revised full disclosure report.

**Comment 2:** Substitution of any other component not specified above is not allowed.
**Comment 3:** The component substitution will be open to challenge for a 60 day period.
**Comment 4:**Patches to the application are not permitted.

***8.12.1.6*** Full Disclosure Report revisions may be required for other reasons according to TPC policies (see TPC Policy Document).

***8.12.1.7*** The revised report should be submitted as defined in clause 8.11.

**8.12.2 Permitted Revisions**

Revisions to the full disclosure documentation are permitted to be published under the following circumstances:

***8.12.2.1*** When cumulative price changes have resulted in a decrease of 2% or more from the reported price/performance, the test sponsor may submit revised price/performance results to the TPC. The benchmark need not be re-run to remain compliant.

***8.12.2.2*** Re-pricing of current results must be reviewed and approved by the auditor if there is a change to the pricing list. Changes in prices of line item components do not constitute a pricing list change.

***8.12.2.3*** A change in the committed availability date for the priced system which is earlier than the published availability date may be published in a revised Full Disclosure Report. The availability date can not be moved earlier than the submission date of the revised FDR.

Additionally the availability date can not be moved to more than 6 months after the original publication date.

***8.12.2.4*** A report may be revised to add or delete Clause 7 related items for country specific priced configurations.

***8.12.2.5*** The revised report should be submitted as defined in clause 8.11

# Clause 9 - **Auditing**

**9.1.1** An independent audit of the benchmark results by an auditor certified by the TPC is required. The current audit checklist may be obtained from one of the TPC certified auditors. The term "independent" is defined as: "the outcome of the benchmark carries no financial benefit to the auditing agency other than fees earned directly related to the audit." In addition, the auditing agency cannot have supplied any performance consulting under contract for the benchmark under audit. The term "certified" is defined as: "the TPC has reviewed the qualification of the auditor and certified that the auditor is capable of verifying compliance of the benchmark result." Please see the TPC Audit Policy for a detailed description of the auditor certification process.

In addition, the following conditions must be met:

1. The auditing agency cannot be financially related to the sponsor. For example, the auditing agency is financially related if it is a dependent division, the majority of its stock is owned by the sponsor, etc.
2. The auditing agency cannot be financially related to any one of the suppliers of the measured/priced components, e.g., the DBMS supplier, application server vendor, etc.

**9.1.2** The auditor's attestation letter must be made readily available to the public as part of the Full Disclosure Report, but a detailed report from the auditor is not required.

The auditor must keep the checklist for any publication for the entire 60-day challenge period. This checklist should be available to any of the parties involved in a challenge during that period.

**9.1.3** For the purpose of the audit, the auditor may not modify the workload for the test run. The auditor may not test for attributes of the SUT that are not required by this specification.

**9.1.4** In the case of audited TPC-W results, which are used as a basis for new TPC-W results, the sponsor of the new benchmark can claim that the results were audited if, and only if:

1. The auditor ensures that the hardware and software products are the same.
2. The application code remains the same.
3. The auditor reviews the Executive Summary of the new results and ensures that it matches what was contained in the original sponsor's Executive Summary.
4. The auditor can attest to Clauses 9.2.8.

The auditor is not required to follow any of the remaining auditor's check list items from clause 9.2.

## 9.2 Auditors Checklist

**9.2.1** Clause 1 - Logical Database Design

***9.2.1.1*** Verify that the specified tables, rows and columns exist and conform to the required population and scaling.

***9.2.1.2*** Verify that the row identifiers are not disk or file offsets and that no user written code accesses row(s) by physical location or offsets.

***9.2.1.3*** Verify that any additions to the tables attributes are disclosed

***9.2.1.4*** Verify that all tables support retrievals, updates, inserts and deletes.

***9.2.1.5*** Verify that all tables can support a growth of 5% additional rows.

***9.2.1.6*** Verify whether any replication of tables has been used, and if so, that it complies with the ACID requirements.

***9.2.1.7*** Verify that each column is logically discrete and independently accessible.

***9.2.1.8*** Verify that the duplicate primary keys are detected and disallowed.

***9.2.1.9*** Verify that any row or column can be updated with the same semantics and syntax.

**9.2.2** Clause 2 – Web Service Interactions

***9.2.2.1*** Verify that all web service interactions are compliant with the security requirements of clause.

***9.2.2.2*** Verify that each Business Session opens a new connection between the RBE and SUT.

***9.2.2.3*** Verify that the SOAP messages for all requests are parsed on the SUT with a commercially available product.

***9.2.2.4*** Verify that the SOAP response messages from the SUT to the RBE are encoded using a commercially available product.

***9.2.2.5*** Verify that all XML parsing on the SUT be done with a commercially available product.

***9.2.2.6*** Verify that connection management is compliant with clauses 2.1.8, 2.1.10 and 2.1.12.

***9.2.2.7*** Verify that a commercially available product is used for message queuing.

***9.2.2.8*** Verify that the RBE generates web service requests that meet all input requirements.

***9.2.2.9*** Verify that the SUT produces responses to web service requests as specified.

***9.2.2.10*** Verify that each EB uses at least one random number generator and that random number generators are not shared between EB's.

***9.2.2.11*** Verify that the random number generation method is the one specified in Appendix A.1.

***9.2.2.12*** Verify that the random number seeds used by the EB's are unique and in compliance with clause 2.1.19.

***9.2.2.13*** Verify that the New Customer web service is implemented as specified.

***9.2.2.14*** Verify that the Change Payment Method web service is implemented as specified.

***9.2.2.15*** Verify that the Create Order web service is implemented as specified.

***9.2.2.16*** Verify that the Shipping process is implemented as specified.

***9.2.2.17*** Verify that the Stock Management process is implemented as specified.

***9.2.2.18*** Verify that the Order Status web service is implemented as specified.

***9.2.2.19*** Verify that the New Products web service is implemented as specified.

***9.2.2.20*** Verify that the Product Detail web service is implemented as specified.

***9.2.2.21*** Verify that the Change Item web service is implemented as specified.

***9.2.2.22*** Verify that business sessions are not shared between multiple EBs.

***9.2.2.23*** Verify that the Web Server Access Log meets the collection and formatting requirements.

**9.2.3** Clause 3 – Database Transaction and System Properties

***9.2.3.1*** Verify that all mechanisms needed to enforce the ACID properties required by the benchmark be enabled.

***9.2.3.2*** Verify that all atomicity tests were performed and completed successfully.

***9.2.3.3*** Verify that the database is compliant with all of the consistency requirements.

***9.2.3.4*** Verify that all isolation tests were performed and completed successfully.

***9.2.3.5*** Verify that the durability test was performed and completed successfully for each component of the SUT required in clause 3.1.5.3.

***9.2.3.6*** Verify that the durability test for the message queues and distributed transaction manager completed successfully.

**9.2.4** Clause 4 – Scaling and Database Population

***9.2.4.1*** Validate that the cardinality of the database tables is compliant.

***9.2.4.2*** Verify that the database population is compliant.

***9.2.4.3*** Verify that the database load utility uses the required random number generator.

***9.2.4.4*** Validate the 60-day space calculations for database growth.

***9.2.4.5*** Validate that the 8-hour space calculations for the web server access log(s), stock management log(s) and database log is compliant.

**9.2.5** Clause 5 – Performance Metrics and Response Time

***9.2.5.1*** Validate that all EB's are active and submitting requests during the test run.

***9.2.5.2*** Verify the log of retries contains no more than 20 entries for any web service interaction identifier (See clause 5.4.1.2).

***9.2.5.3*** Verify that the mix of web service requests satisfies the required mix during the measurement interval.

***9.2.5.4*** Verify that the SIRT's reported during the measurement interval meet the response time constraints.

***9.2.5.5*** Verify the validity of the method used to accurately measure and report the Service Interaction Response Time (SIRT) at the RBE.

***9.2.5.6*** Verify that the reported throughput rating is computed correctly.

***9.2.5.7*** Verify that the test run meets all of the requirements in clause 5.4.

***9.2.5.8*** Verify that the measurement interval meets all of the requirements in clause 5.5.

**9.2.6** Clause 6 – SUT, Driver and Communications

***9.2.6.1*** Verify the RBE is performing as specified.

***9.2.6.2*** Verify that the required communication protocol and encryption level is used between the RBE, SUT, and external emulators (POV, PGE, ICE).

***9.2.6.3*** Verify that each EB opens its own socket connection to the SUT instead of sharing a socket with another EB.

***9.2.6.4*** Verify that the restrictions placed on the activity of the SUT are met (see clause 6.4.3).

***9.2.6.5*** Verify that the SUT meets the resource growth requirements defined in clause 6.10.

***9.2.6.6*** Verify that the SUT meets the non-disruptive maintenance requirements defined in clause 6.10. If the auditor feels it is sufficient, documentation can be used to meet this requirement.

***9.2.6.7*** Verify that the network configuration documented in the FDR matches the tested network configuration.

***9.2.6.8*** Verify that the RBE drives the SUT in a compliant manner.

***9.2.6.9*** Verify that the RBE generates compliant business session lengths.

***9.2.6.10*** Verify that the SUT components are compliant with clause 6.4.1.

***9.2.6.11*** Verify the functions of the SUT comply with clause 6.4.2

***9.2.6.12*** Verify the SUT complies with the restrictions defined in clause 6.4.3.

***9.2.6.13*** Verify that the POV is compliant with clause 6.5

***9.2.6.14*** Verify that the PGE is compliant with clause 6.6.

***9.2.6.15*** Verify that the ICE is compliant with clause 6.7.

***9.2.6.16*** Verify that the application protocols used are compliant with the WS-I basic profile 1.0.

**9.2.7** Clause 7 – Pricing

***9.2.7.1*** Verify that the pricing model includes all the tested hardware and software components. This includes but is not limited to sufficient storage for database and all other logs. It also includes any and all components required to develop and deploy the services.

***9.2.7.2*** Verify the pricing spreadsheet correctly computes the 3-year cost of ownership.

***9.2.7.3*** Verify that any customer spareable components used are priced in sufficient quantity to meet the requirements.

***9.2.7.4*** Verify the compliance of any discount, warranty or other maintenance used to meet the pricing requirements.

***9.2.7.5*** Verify the compliance of all price quotes for components from third party sources.

***9.2.7.6*** The auditor is not required to verify the pricing of components supplied by the benchmark sponsor(s).

***9.2.7.7*** Verify that the price performance ratio is computed according to clause 7.4.

**9.2.8** Clause 8 – FDR

Verify that the Full Disclosure Report is complete and accurate. Auditors can limit their review of the FDR to the following sections:

- Executive Summary
- System Configuration
- Database population and space calculations
- Sections that include the disclosure of measurement data.
- Any other section that the auditor feels is appropriate.

**Comment:** The intent is to limit the review to something less than a line by line check while still providing for a reasonable assurance that the data is accurate.

# Appendix A - RANDOM NUMBER GENERATOR

## A.1 Marsaglia Algorithm

The generation of pseudo-random numbers in this specification requires the use of the following algorithm by George Marsaglia:

```
#define znew ((z=A*(z&65535)+(z>>16))<<16)
#define wnew ((w=B*(w&65535)+(w>>16))&65535)
#define IUNI (znew+wnew)
#define UNI (znew+wnew)*2.32830643708e-10
```

where z and w are static 32 bit unsigned integers that are initialized to unique non-zero values across all Ebs. The term “start seeds” refers to this unique combination of z and w.

The two constants A and B must be chosen as any pair of distinct constants from this list:

```
18000  18030  18273  18513  18879  19074  19098  19164  19215  19584
19599  19950  20088  20508  20544  20664  20814  20970  21153  21243
21423  21723  21954  22125  22188  22293  22860  22938  22965  22974
23109  23124  23163  23208  23508  23520  23553  23658  23865  24114
24219  24660  24699  24864  24948  25023  25308  25443  26004  26088
26154  26550  26679  26838  27183  27258  27753  27795  27810  27834
27960  28320  28380  28689  28710  28794  28854  28959  28980  29013
29379  29889  30135  30345  30459  30714  30903  30963  31059  31083
```

Using the UNI definition will yield a 32-bit floating point result in the range from [0..1]

Using the IUNI definition will yield a 32-bit integer result.

# Appendix B - STANDARD REFERENCES

## B.1 Internet RFCs

The following Internet Requests for Comments are referenced by this benchmark:

Hypertext Transfer Protocol 1.1 (HTTP 1.1), as defined in RFC 2616

http://www.ietf.org/rfc/rfc2616.txt

Internet Protocol (IPv4), as defined in RFC 791

http://www.ietf.org/rfc/rfc0791.txt

Transmission Control Protocol (TCP), as defined in RFC 793

http://www.ietf.org/rfc/rfc0793.txt

Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework as in RFC 2527

http://www.ietf.org/rfc/rfc2527.txt

PKCS#1 RSA Cryptography Specification Version 2.0 as defined in RFC 2437

http://www.ietf.org/rfc/rfc2437.txt

SSL V3 is a de-facto standard and does not have an RFC number. The specifications can be found at

http://home.netscape.com/eng/ssl3/ssl-toc.html

Transport Layer Security. The specifications can be found at

http://www. ietf.org/rfc/rfc2246.txt

WS-I BP1.0 Spec can be found at

http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.htm

RC4 is a trademark of the RSA Corporation

## B.2 Common Log Format

The Common Log Format is a standard of the W3C, Worldwide Web Consortium. W3C is an international industry consortium, jointly hosted by the Massachusetts Institute of Technology Laboratory for Computer Science [MIT/LCS] in the United States; the Institut National de Recherche en Informatique et en Automatique [INRIA] in Europe; and the Keio University Shonan Fujisawa Campus in Japan. Services provided by the Consortium include: a repository of information about the World Wide Web for developers and users; reference code implementations to embody and promote standards; and various prototype and sample applications to demonstrate use of new technology. Initially, the W3C was established in collaboration with CERN, where the Web originated, with support from DARPA and the European Commission

A description of Common Log Format can be found at:

http://www.w3.org/Daemon/User/Config/Logging.html

http://delinfo.cern.ch/WebOffice/Services/WWWlogfiles/CommonLogFormat.html

# Appendix C - **Cumulative Distribution Function**

The following table prescribes the CDF that must be used by the EB to randomly generate a BSL. The range of the distribution is [1, 120] and its mean is approximately 40.5. The CDF table values were generated using Microsoft® Excel 2000's BETADIST function. That is, for BSL = B, where 1 ? B ? 120, the table entry Cumulative Probability = `BETADIST( B, 1.5, 3, 0, 120).`

A tab delimited file with the below values can be obtained on the TPC website at:

http://www.tpc.org/tpcw/

***(Note to committee: The details and derivation of the CDF is TBD. Any CDF with any range and mean can be substituted here. Any table-defined CDF can be input to the code to be listed in Appendix D - .)***

| BSL | Cumulative probability |
|---|---|
| 1 | 0.003294992562922 |
| 2 | 0.009226351820093 |
| 3 | 0.016779527018572 |
| 4 | 0.025573064780547 |
| 5 | 0.035377310219299 |
| 6 | 0.046031555631884 |
| 7 | 0.057413851386530 |
| 8 | 0.069426812578452 |
| 9 | 0.081989894755458 |
| 10 | 0.095034761109571 |
| 11 | 0.108502302596382 |
| 12 | 0.122340616981276 |
| 13 | 0.136503579980181 |
| 14 | 0.150949800480539 |
| 15 | 0.165641835099602 |
| 16 | 0.180545583775404 |
| 17 | 0.195629815327623 |
| 18 | 0.210865788597611 |
| 19 | 0.226226945354508 |
| 20 | 0.241688658076690 |
| 21 | 0.257228020372218 |
| 22 | 0.272823671006604 |
| 23 | 0.288455644759757 |
| 24 | 0.304105244948701 |
| 25 | 0.319754933629398 |
| 26 | 0.335388236361781 |
| **BSL** | **Cumulative probability** |
| 27 | 0.350989659075434 |
| 28 | 0.366544615069809 |
| 29 | 0.382039360564471 |
| 30 | 0.397460937511409 |
| 31 | 0.412797122614074 |
| 32 | 0.428036381682025 |
| 33 | 0.443167828597196 |
| 34 | 0.458181188286159 |
| 35 | 0.473066763188783 |
| 36 | 0.487815402792041 |
| 37 | 0.502418475862118 |
| 38 | 0.516867845061182 |
| 39 | 0.531155843679473 |
| 40 | 0.545275254250299 |
| 41 | 0.559219288846645 |
| 42 | 0.572981570884307 |
| 43 | 0.586556118278708 |
| 44 | 0.599937327821512 |
| 45 | 0.613119960659312 |
| 46 | 0.626099128770628 |
| 47 | 0.638870281206082 |
| 48 | 0.651429197180008 |
| 49 | 0.663771967156494 |
| 50 | 0.675894988195849 |
| 51 | 0.687794952665143 |
| 52 | 0.699468839097459 |
| **BSL** | **Cumulative probability** |
| 53 | 0.710913903582423 |
| 54 | 0.722127671646222 |
| 55 | 0.733107928255413 |
| 56 | 0.743852720466196 |
| 57 | 0.754360334682216 |
| 58 | 0.764629301552815 |
| 59 | 0.774658386694024 |
| 60 | 0.784446584827467 |
| 61 | 0.793993114207044 |
| 62 | 0.803297411314016 |
| 63 | 0.812359125802747 |
| 64 | 0.821178115680811 |
| 65 | 0.829754442708460 |
| 66 | 0.838088368003678 |
| 67 | 0.846180347840096 |
| 68 | 0.854031029626028 |
| 69 | 0.861641248053797 |
| 70 | 0.869012021409305 |
| 71 | 0.876144548032555 |
| 72 | 0.883040202920511 |
| 73 | 0.889700534464284 |
| 74 | 0.896127261313213 |
| 75 | 0.902322269358916 |
| 76 | 0.908287608580333 |
| 77 | 0.914025491326588 |
| 78 | 0.919538287888426 |
| 79 | 0.924828525154045 |
| 80 | 0.929898883761605 |
| 81 | 0.934752195682031 |
| 82 | 0.939391441884641 |
| 83 | 0.943819750081570 |
| 84 | 0.948040392547227 |
| 85 | 0.952056784009222 |
| 86 | 0.955872479607450 |
| 87 | 0.959491172918172 |
| 88 | 0.962916694040157 |
| 89 | 0.966153007740084 |
| 90 | 0.969204211654587 |
| 91 | 0.972074534546459 |
| 92 | 0.974768334612681 |
| 93 | 0.977290097842055 |
| 94 | 0.979644436420346 |
| 95 | 0.981836087180971 |
| 96 | 0.983869910099336 |
| 97 | 0.985750886829062 |
| 98 | 0.987484119278408 |
| 99 | 0.989074828225297 |
| 100 | 0.990528351969418 |
| 101 | 0.991850145019981 |
| 102 | 0.993045776817743 |
| 103 | 0.994120930489998 |
| 104 | 0.995081401636415 |
| 105 | 0.995933097150345 |
| 106 | 0.996682034058694 |
| 107 | 0.997334338402302 |
| 108 | 0.997896244136765 |
| 109 | 0.998374092061239 |
| 110 | 0.998774328772679 |
| 111 | 0.999103505644643 |
| 112 | 0.999368277829786 |
| 113 | 0.999575403285268 |
| 114 | 0.999731741820280 |
| 115 | 0.999844254164961 |
| 116 | 0.999920001059985 |
| 117 | 0.999966142366158 |
| 118 | 0.999989936193367 |
| 119 | 0.999998738048247 |
| 120 | 1.000000000000000 |

# Appendix D - **BSL Variate Sample Code**

The following is sample code for generating the BSL variates from the BSL cumulative distribution function (CDF).

The following code may be used to implement how EB's generate BSL's. It is not required that this code be used; it is being provided as a convenience. What is required is that the EB's generation of BSL's conforms to the BSL CDF given in Appendix C - .

***(Note to committee: How to test this conformance?)***

C:
<source listing to be inserted here>

C#:
<source listing to be inserted here>

Java:
<source listing to be inserted here>

# Appendix E - **WSDL Samples**

The following are sample WSDL documents for the web service SOAP requests and responses used by the specification. These WSDL documents are provided for reference purposes and the actual implementation may vary.

## E.1 New Customer WSDL

```
<source listing to be inserted here>
```

## E.2 Change Payment Method WSDL

```
<source listing to be inserted here>
```

## E.3 Create Order WSDL

```
<source listing to be inserted here>
```

## E.4 Order Status WSDL

```
<source listing to be inserted here>
```

## E.5 New Products WSDL

```
<source listing to be inserted here>
```

## E.6 Product Detail WSDL

```
<source listing to be inserted here>
```

## E.7 Change Item WSDL

```
<source listing to be inserted here>
```

# Appendix F - **EXECUTIVE SUMMARY**

This appendix includes a sample Executive Summary, which includes the following 3 pages:

Overview Page

Pricing Page

Numerical Quantities Summary Page.

Clause 7 - and Clause 8.2 give a detailed description of the required format and content of the Executive Summary. This sample is provided only as an illustration of the requirements set forth in this specification. In the event of a conflict between this example and the specification, the specification shall prevail.

The required format of the TPC-W Executive Summary may change from time to time. The latest version of the required format is available upon request from the TPW administrator (see cover page).

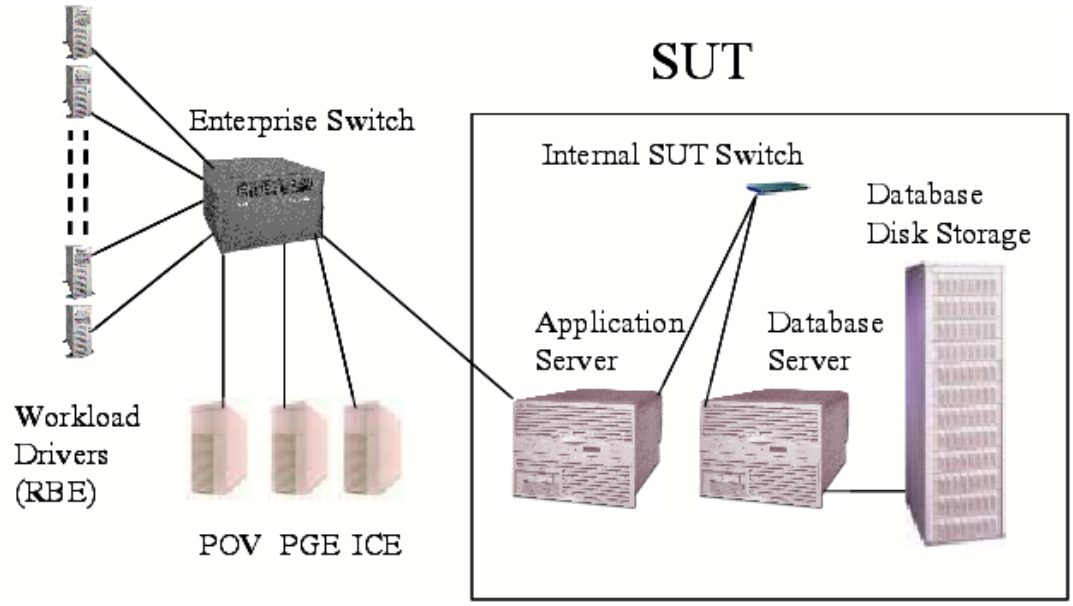
## F.1 Implementation Overview Page

The following page demonstrates how the Overview page should appear.

| Some Logo | Mothra 2000 | TPC-W Rev. 2.0<br>Report Date<br>7/4/2003 |
|---|---|---|

| Price Performance | Performance | Availability Date |
|---|---|---|
| **$89.88 per SIPS** | **10,439.6 SIPS** | **July 10, 2003** |
| **Total Cost** | **Number of Active Users** | **Average Response Time** |
| **$938,347** | **500** | **2.17** |
| **Number of Systems** | **Application Server** | **Managed Runtime** |
| **2** | **Excalibur Appmaster XII** | **Merlin 3.0** |
| **Distributed Transaction Mgr** | **Database Mgr** | **Message Queueing Product** |
| **Pivotman** | **Joe's DBMS** | **Message Master VII** |

**Other Software:** Joe's Visual RPG Developer Studio




| Function | System | Num. of Systems | Operating System | Processors | Memory |
|---|---|---|---|---|---|
| Application Server | Mothra 2000 | 1 | Exec 8 | 16 Thunderbolt IV 900 Mhz/2MB L2 | 8 GB |
| Database Server | Mothra 2000 | 1 | Exec 8 | 16 Thunderbolt IV 900 Mhz/2MB L2 | 32 GB |

## F.2 Pricing Page

| Some Logo | Mothra 2000 | TPC-W Rev. 2.0<br>Report Date<br>7/4/2003 |
|---|---|---|

| Price Performance | Performance | Availability Date |
|---|---|---|
| $89.88 per SIPS | 10,439.6 SIPS | July 10, 2003 |

| Description | Part No. | Third Party Brand | Third Party Price | Unit Price | Qty. | Extended Price | 3 yr.Maint. Price |
|---|---|---|---|---|---|---|---|
| **Application & Database Server Hardware** | | | | | | | |
| SYS:Mothra 2000, Dual Pwr, 0CPU, 0MB | MO716-DDN | | | $200,000 | 1 | $200,000 | $24,000 |
| CPU:4x 700MHz Thunderbolt IV, 2MB L2 | MO747-2MB | | | $22,000 | 4 | $88,000 | |
| MEM: 16MB L3 Cache | MO714-SPD | | | $15,000 | 4 | $60,000 | |
| MEM: 4GB Memory, SDRAM | RAM512-4G | | | $24,000 | 2 | $48,000 | |
| ACC: High Availability Package | MO7001-HAP | | | $25,000 | 1 | $25,000 | |
| ACC: PCI Module, 3.3v | PCI3-MOD | | | $400 | 4 | $1,600 | |
| ETHERNET: 10/100Mbit/sec, PCI 32-bit | E1010052-PCI | | | $135 | 2 | $270 | |
| ETHERNET: 10/100Mbit/sec, Dual Port | E1010053-PCI | | | $300 | 2 | $600 | |
| ETHERNET: 1000Mbit/sec, PCI 64-bit | E100051-P64 | | | $850 | 2 | $1,700 | |
| CTRL: Fibre Channel HBA, 64-bit PCI | FCH201-P64 | | | $1,700 | 2 | $3,400 | |
| CTRL: SCSI, 2 Channel LVD, 64-bit PCI | PCI50233-P64 | | | $765 | 2 | $1,530 | |
| COMM: 56K LAN Modem | MDM56 | | | $275 | 1 | $275 | |
| DISK: 18GB Drive, 10Krpm, SCSI LVD | H8110 | | | $600 | 2 | $1,200 | |
| PWR: AC Entry Module, 3 PH | APA1000 | | | $2,000 | 2 | $4,000 | |
| SYS MGT: Mothra 2000 Value Add Software | MO700011-N | | | $5,000 | 1 | $5,000 | $324 |
| SYS MGT: Mothra 2000 SW Update Subscription | S200016-L | | | $16,200 | 3 | $48,600 | |
| MAINT: 3-Yr. Cmprhnsv-Gold Svc Wrnty Upgrd | WAD7 | | | $22,992 | 1 | | $22,992 |
| MONITOR: 15-inch Color | VGA2100-P | | | $300 | 2 | $600 | |
| DAE: Expansion Enclosure w/ 10x36GB Disks | ESM6 | | | $29,622 | 2 | $59,244 | $9,504 |
| DPE: 2 RAID Cntrls, 2x128MB,10x36GB Disks | ESM6-10 | | | $41,695 | 1 | $41,695 | $6,444 |
| SYS MGT: Storage Agent, Monitor, Call Home & ATF | ESM672K | | | $2,219 | 1 | $2,219 | $108 |
| CBL: Fibre Channel, DB9->DB9 Conn's, 5m | CBL135-5 | | | $310 | 4 | $1,240 | |
| PWR: Distribution Unit, 1U, 220V | RM69-PDU | | | $596 | 2 | $1,192 | |
| CAB: 36U x 19" x 34" Cabinet, Open | RM691934-OFE | | | $1,100 | 1 | $1,100 | |
| DOOR: 36U x 19", Rear | RM6919-RDR | | | $375 | 1 | $375 | |
| PNL: 36U x 34" Side Skins, L&R | RM6934-SDS | | | $300 | 1 | $300 | |
| **Software** | | | | | | | |
| O/S: Exec 8 | | 1 | | Inc. above | 1 | Inc. above | |
| APP: Joe's DBMS | | 1 | 1 | $17,010 | 16 | $272,160 | $6,285 |
| ACC: Huey's RPG Compiler | | 1 | 1 | $549 | 1 | $549 | |
| | | | | **Subtotal** | | **$869,849** | **$69,657** |
| Service Pre-Pay Discount | | | | | | | ($1,159) |

| | | | |
|---|---|---|---|
| 3rd Party Brand: | 1 = Huey's Software Emporium | **3 Year Cost of Ownership** | **$938,347** |
| 3rd Party Pricing: | 1 = Huey's Software Emporium | | |

| Independently Audited by: | William Wallace | Highland Performance |
|---|---|---|

## F.3 Numerical Quantities Summary Page

| Some Logo | Mothra 2000 | TPC-W Rev. 2.0<br>Report Date<br>7/4/2003 |
|---|---|---|

| Price Performance | Performance | Availability Date |
|---|---|---|
| $89.88 per SIPS | 10,439.6 SIPS | July 10, 2003 |

### Numerical Quantities Summary

**SIPS Interaction Summary**

| Web Service Interaction | Mix Percentage | Min Response Time | Max Response Time | 90th Percentile Response Time | Average Response Time |
|---|---|---|---|---|---|
| New Customer | 1.010% | 1.34 | 2.65 | 1.60 | 1.55 |
| Change Payment Method | 4.990% | 1.50 | 2.99 | 1.93 | 1.88 |
| Create Order | 50.020% | 2.03 | 2.66 | 2.60 | 2.50 |
| Order Status | 19.980% | 0.33 | 0.43 | 0.50 | 0.40 |
| New Products List | 10.010% | 0.42 | 0.66 | 0.63 | 0.55 |
| Product Detail | 11.905% | 0.78 | 0.97 | 0.83 | 0.85 |
| Change Item | 2.085% | 0.56 | 0.78 | 0.75 | 0.66 |