

TPC Procedures
Version 0.3 DRAFT

January 2020

Transaction Processing Performance Council
781 Beach St, Suite 302
San Francisco, CA 94109
Phone: (415) 561-6272
FAX: (415) 561-6120

Email: info@tpc.org
<http://www.tpc.org>

© 2020 Transaction Processing Performance Council
All Rights Reserved

Legal Notice

The TPC reserves all right, title, and interest to this document and associated source code as provided under U.S. and international laws, including without limitation all patent and trademark rights therein.

Permission to copy without fee all or part of this document is granted provided that the TPC copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the Transaction Processing Performance Council. To copy otherwise requires specific permission.

Trademarks

TPC, TPC Benchmark, TPC-C, TPC-E, TPC-H, TPC-DS and TPC-VMS are trademarks of the Transaction Processing Performance Council.

Product names, logos, brands, and other trademarks featured or referred to within this Specification are the property of their respective trademark holders.

TPC Membership

A list of the current TPC Member companies can be found at
http://www.tpc.org/tpc_documents_current_versions/pdf/tpc_membership.pdf

Document Revision History

Date	Version	Description
August 2019	0.1	Initial Draft
January 2020	0.2	Second Draft
January 2020	0.3	Third Draft

Typographic Conventions

The following typographic conventions are used in this specification:

Convention	Description
Bold	Bold type is used to highlight terms that are defined in this document
<i>Italics</i>	Italics is used to highlight text that should be used in TPC documents verbatim
UPPERCASE	N/A

Table of Contents

Section 0: Terms, Notation, and Policy Modification	6
0.1 Notation	6
0.2 Defined Terms	6
0.2.1 Policies. The current published Policies of the TPC.....	6
0.2.2 Procedures. The current Procedures of the TPC, i.e., this document.	6
Section 1: TPC Software Development	7
1.1 Problem Reports	7
1.2 Development.....	7
1.3 Testing.....	7
1.4 Release	8
Section 2: TPC GitHub Usage.....	9
2.1 General.....	9
2.2 Repositories	9
2.3 Repository Naming.....	9
2.4 Creation of Repositories	10
2.5 Access Control.....	10
2.5.1 “Source Code” Repositories	10
2.5.2 “Non-Code” Repositories (except for Standing Committees).....	10
2.5.3 “Non-Code” Repositories (Standing Committees).....	10

Section 0: **Terms, Notation, and Policy Modification**

0.1 **Notation**

- 0.1.1 A reference to a specific clause in the **Bylaws, Policies** or **Procedures** is written as “**Bylaws** § x.y.z”, “**Policies** § x.y.z” or “**Procedures** § x.y.z”, respectively, where x.y.z is the clause number.
- 0.1.2 Throughout the body of this document, defined terms (see **Procedures** § 0.2) are formatted in the same style as used in the term definition to indicate that the term has a precise meaning. For example, “**Members**” specifically refers to voting members of the **TPC**, whereas “members” does not have any special meaning.

0.2 **Defined Terms**

Comment: Any defined term that is not listed in this section shall be found in the **Policies**.

- 0.2.1 **Policies.** The current published **Policies** of the **TPC**.
- 0.2.2 **Procedures.** The current **Procedures** of the **TPC**, i.e., this document.

Section 1: TPC Software Development

1.1 Problem Reports

To facilitate problem reporting, the **TPC** will provide a problem reporting tool via a web-based interface (or through other mechanisms as defined by the **Council**).

1.1.1 **Members** are encouraged to report problems to the **TPC** in a timely fashion.

1.1.2 Problem reports will be classified as one of the following:

- **Bug / Error:** A problem that prevents the proper operation of the **TPC-Provided Software**. This includes any problem that arises out of a change in the version of the **TPC-Provided Software** (e.g. v1.0.0 works fine, but v1.0.1 fails to operate properly).
- **New Feature / Enhancement:** A request for new (or enhanced) functionality.
- **Portability:** A problem that prevents the operation of **TPC-Provided Software** with a specific combination of hardware, operating system, compiler and/or data manager. This includes issues of the following nature:
 - a) Enhancement requests to add support for a new version of a platform (e.g. Add support for MyDBMS v2.0.0)
 - b) Error reports indicating that **TPC-Provided Software** no longer works correctly as a result of the platform change (e.g. v1.0.0 compiles fine on OS v1.x but fails on OS v2.x).

1.2 Development

Changes to **TPC-Provided Software** by a subcommittee will follow the process outlined in the following clauses.

1.2.1 A document describing the requirements for a code change is produced. The change must be linked to one or more problem reports entered in the problem reporting system (See **Procedures** § 1.1).

1.2.2 The subcommittee must vote to accept the documented requirements before considering any code changes. (See **Policies** § 3.5.4.2) The subcommittee may modify the requirements during the acceptance process. The subcommittee is encouraged to develop a test case for any proposed changes.

1.2.3 The code change and any test case(s) will be made available for evaluation and a notification sent to the subcommittee. Code changes and test case(s) will only be accepted if a signed **CLA** is on file with the **TPC Administrator**.

1.2.4 Code changes must be accepted by a vote of the subcommittee. (See **Policies** § 3.5.4.2)

1.3 Testing

All **TPC-Provided Software** must be tested on a representative set of platforms with the assistance of member companies. The type and amount of testing performed on each platform must be sufficient to ensure proper operation of the **TPC-Provided Software**. The following categories provide guidelines for the types of testing which is expected.

1.3.1 **Platform testing of source code:** Simple tests that validate the quality of the source code and compliance with coding best practices. This includes verifying that the source code compiles without warnings on a representative set of platforms, as well as testing with third-party code analysis tools used to validate the code for best practices (e.g. memory leaks, exception handling, etc.).

- 1.3.2 **Platform testing of executable code:** Tests that validate the required functionality of code. Tests also verify that exception handling is correct and check for memory leaks and other unintended side effects.
- 1.3.3 **Unit testing:** Tests for specific functionality, on a routine or method basis. Test cases are generally simple (input X produces output Y). Examples include random number and date/time generation.
- 1.3.4 **Functional testing:** Tests designed to exercise specific functionality on a subsystem basis. Test cases are more complicated and may require specialized code to simulate the operation of the benchmark and/or validate the results of the simulation. Examples include input generation and mix control.
- 1.3.5 **End-to End testing:** Tests the entire operation of the benchmark, performed by **Members** in their environment(s). Test cases are designed to validate the data generated by a revision of the code are comparable to previous versions and verify no functional differences have been introduced.

1.4 Release

To create a revision of **TPC-Provided Software**, the subcommittee must:

- 1.4.1 Collect all approved code changes to include in the revision (**Procedures § 1.2**), as appropriate for the type of revision level (**Policies § Error! Reference source not found.**).
- 1.4.2 Produce a “beta” revision of the code with sufficient lead time to allow member companies to integrate the code into their environment for verification of the proposed changes. The “beta” revision could be identified with a fourth-level identifier.
- 1.4.3 Perform appropriate testing (**Procedures § 1.3**) to ensure the proposed changes (**Procedures § 1.2**) properly address the associated problem reports (**Procedures § 1.1**) and ensure that no new problems are introduced.
- 1.4.4 Ensure that a least one member company tests the proposed changes in an end-to-end environment (**Procedures § 1.3.5**) and report back in a timely manner.
- 1.4.5 The details of the testing performed must be documented in subcommittee meeting minutes.
- 1.4.6 Resolve any reported issues with the proposed changes to the satisfaction of the subcommittee.
- 1.4.7 If any previously approved code changes cannot be included in the release for any reason, exclusion requires a subcommittee vote. (See **Policies § 3.5.4.2**)
- 1.4.8 The subcommittee must vote to release an official revision of the **TPC-Provided Software** for approval by the **Directors**. (See **Policies § 3.5.4.2**)
- 1.4.9 **Comment:** The voting requirements to approve the type of revision level for the changes to **TPC-Provided Software** are specified in the **Benchmark Class** requirements. In the case of **TPC-Provided Software** that is independent of a **Benchmark Standard**, the voting requirements are listed in **Policies § Error! Reference source not found.**

Section 2: TPC GitHub Usage

2.1 General

All source code, documentation and issue tracking for **TPC** Benchmarks and **TPC-Provided Software** is to be maintained in GitHub repositories.

2.2 Repositories

With GitHub, the central unit is the repository. A repository is a place for source code, documentation, or other materials that are to be version controlled (henceforth "materials") and comes with an issue tracker and a wiki.

For large enterprise projects, there is some debate about whether to keep materials and issues in the same repository. In large projects, since the group of people creating issues are different than the group of people maintaining the materials, it is prudent to use separate repositories for materials and issues, so access can be managed more granularly.

However, for **TPC** purposes, the vast majority of issues will come from developers themselves, and thus it should be fine to track issues in the same repository as the materials. In addition, within a single repository, a user can have read-only access to materials and read-write access to issues and the wiki, so this allows for a suitable level of access control given our rules around code access and the **CLA**.

Resolution #1: We will keep materials and issues in the same repository.

The **TPC** also produces two main types of material -- **TPC** Specifications and **TPC-Provided Software**. Specifications can range from a user-guide type of documentation (for Express benchmarks), to very detailed implementation documents (for Enterprise benchmarks). **TPC-Provided Software** can range from simple data generators (for Enterprise benchmarks) to full end-to-end kits (for Express benchmarks).

Since we have **CLA** requirements for code, but no such requirement for documentation (specifications), it would make sense to keep Specifications and Software in separate repositories.

While this does mean two repositories (and sets of issues) to track, there are various tools for GitHub that can provide unified views of outstanding issues to make it simpler to do issue triage / project planning.

Resolution #2: We will keep code and non-code materials in separate repositories.

2.3 Repository Naming

So how should we name the repositories? We know that there are code and non-code repositories, so that will need to be part of the name. In general code and non-code materials are associated with a single benchmark (since we are not the best at sharing and reusing), but we also need to consider the potential that a single committee may maintain multiple specifications or software packages, so the software package or specification name should be part of the repository name as well. A third token should be allowed for differentiation (eg, major versions that entail complete rewrites, archived code from other repositories, and so on.)

Resolution #3: The following three-part naming convention will be used.

```
<repo_name> := <scope_name> '-' <other_name> '-' <type_name>
```

```
<scope_name> := <package_name> | <benchmark_name> | <committee_name>
```

```
<package_name> := <alnum>
```

```
<benchmark_name> := <alnum>
```

```
<committee_name> := <alnum>
```

```
<other_name> := 'software' | 'kit' | 'benchmark' | 'general'  
<type_name> := 'code' | 'noncode'  
<alnum> := 'a' through 'z', 'A' through 'Z', '0' through '9'
```

Examples:

```
EGen-software-code           # for EGen code  
TPCE-benchmark-noncode      # for TPC-E specification  
  
TPCxBB-kit-code             # for TPCx-BB kit  
TPCxBB-benchmark-noncode    # for TPCx-BB specification  
  
Pricing-general-noncode     # for TPC-Pricing specification  
  
OSS-general-noncode         # for OSS general non-code materials  
HammerDB-software-code      # for HammerDB  
OpenDataGen-software-code   # for an open-source data generator
```

2.4 Creation of Repositories

In what situations would we need to create new repositories? In general we need a repository for any sizable piece of work that we intend to make public (such as a specification or software), or possibly to support the operation of a **Standing Committee** or a technical subcommittee.

Resolution #4: The following actions may drive the creation of a new repository:

- a new benchmark specification
- a new software package or kit
- a major revision of a benchmark or software package (that are substantial, such as a re-architecture)
- license changes to a software package or kit

2.5 Access Control

Under **Policies** § 1.1, the contents of source code and/or document repositories are considered **TPC Confidential**. The contents of these repositories are intended to be open to all **Members**. However, given the legal requirements of our **CLA**, access controls must be maintained on repositories containing **TPC** source code.

2.5.1 “Source Code” Repositories

Any 'code' repository: **TPC**

- read-write access: specific users with business need and signed **CLA** on file
- read-only access: all **Members**

2.5.2 “Non-Code” Repositories (except for Standing Committees)

Any 'non-code' repository except for standing committees:

- read-write access: all **TPC Members**
- read-only access: <none>

2.5.3 “Non-Code” Repositories (Standing Committees)

Any 'non-code' repository for **Standing Committees**:

- read-write access: **Standing Committee** members only

- read-only access: <none>