-- --

IEEE P1003.0 Draft 13 – September 1991

4.10 Transaction Processing Services

Responsibility: Bob Gambrel

Based on input created by Carl Hall of P1003.11, with contributions from Jeremy Denny, John Penny, Patrick Prange, and Jeff DeMent Revised in Draft 13 with input from Dorm Fulton of P1003.11.

4.10.1 Overview and Rationale

The database management clause (see 4.4) described some transaction processing service requirements (specific to databases). This clause describes the complete set of transaction processing services from the application software point of view. Note that transaction processing services have long been been regarded, variously, to be within the domain of databases or within the domain of operating systems, and more recently within the domain of interconnect. These services are more broadly applicable and so are treated here as a separate clause.

Transactions (''units of work'') have boundaries (start points and end points) that are determined by the action of the transaction application program. The transaction application program can request to either commit or rollback the work done in the transaction when it identifies the end point. The system will complete a commit operation only if all operations performed during the transaction can complete successfully. Otherwise the system will abort the transaction (rollback the work done by it) and notify the transaction application program of this action.

The following is quoted with a few editorial changes from ISO/IEC DP 10026-1 {*ref*}, the ISO Distributed Transaction Processing standard draft:

> A transaction is characterized by four properties: atomicity, consistency, isolation, and durability. These are the *ACID* properties.
>
> Atomicity implies that the operations of a unit of work are either all performed, or none of them are performed.
>
> Consistency implies that the operations of a unit of work, if performed at all, are performed accurately, correctly, and with validity, with respect to application semantics.
>
> Isolation implies that the partial results of a unit of work are not accessible, except by operations which are part of the unit of work. Isolation also implies that units of work which share bound data are serializable.
>
> Durability implies that all the effects of a completed unit of work are not altered by any sort of failure.

4.10.2 Scope

This clause deals with the transaction processing services needed for a large number of styles of transaction processing including the following:

- Transactional access to a single database manager on a single machine
- Transaction access to nondatabase ‘‘resource managers’’ (such as the software managing the cash in an automatic teller machine
- Distributed Databases—databases spanning multiple machines, but accessed by application programs as if just a single database
- Online Transaction Processing (OLTP)—the scheduling of ‘‘transaction programs’’ based on terminal input with consolidated recovery of the database updates and the terminal messages
- Distributed Transaction Processing (DTP)—different machines running multiple application programs with multiple databases, using a client/server or conversational application-to-application communications paradigm

Note that Transaction Processing Services are used in all of the above situations, and others too.

Finally, it should be noted that ‘‘transactions’’ are not really ‘‘messages,’’ but rather ‘‘units of work’’ that may encompass multiple messages. Furthermore, while traditionally, ‘‘transaction processing’’ has usually been synonymous with ‘‘OLTP’’ where so-called ‘‘immediate transactions’’ are the norm, other types of transactions are also covered: ‘‘batch transactions’’ (where the work is done in the ‘‘background’’) and ‘‘deferred transactions’’ where there may be a time dependence on the transaction, such as a fixed start time.

This clause addresses the current work in progress in groups such as ISO and others.

4.10.3 Reference Model

This subclause addresses the conventional Transaction Processing Reference Model, the POSIX OSE Reference Model (incorporating transaction processing considerations), and other important real world considerations introduced by Transaction Processing.

4.10.3.1 Conventional Transaction Processing Reference Model

A model for transaction processing is developed here to complement the POSIX system model. Current work in progress by the POSIX.11 Transaction Processing Working Group and other groups such as ISO/IEC JTC 1/SC21 Open Systems Interconnection—Distributed Transaction Processing may result in a Transaction Processing Reference Model more suitable than the one developed here. At that time, such a model will be referenced and incorporated into the POSIX OSE reference model. Until that time, the current model will be used as a convenient means for describing the services needed in this domain.

While transaction processing services have usually been thought of as applying to databases, the applicability goes further. Nonetheless, the description given here of the transaction processing model shows how the transaction processing program can view the transaction services as an extension of the the Database view of the POSIX OSE reference model as shown in Figure 4-11. From the transaction application program point of view, a transaction processing system has additional capabilities that go beyond those provided by database systems. These services to the transaction application program are provided at an API that is called the *Transaction Manager API.* (See Figure 4-20.) For convenience in discussing the model, the provider of those services is called the *Transaction Manager* (TM).

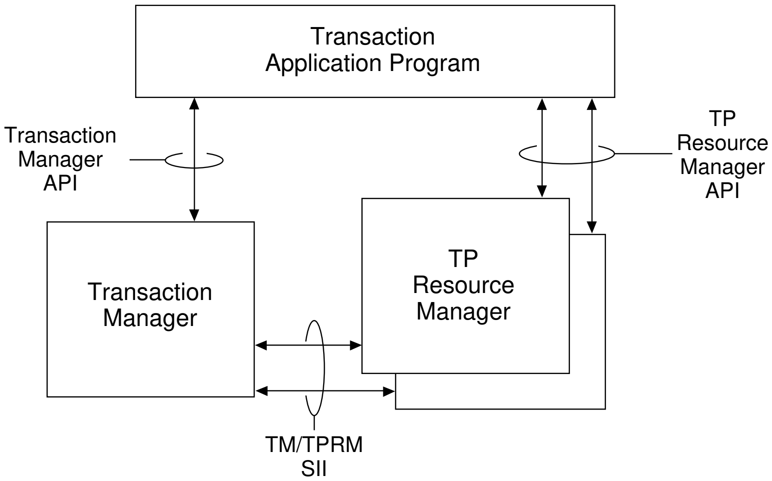

Figure 4-20 – The Conventional Transaction Processing Model

The transaction application program requests services provided by the *TP resource manager* (e.g., a database manager) via the *TP resource manager API.* The transaction manager API and the TP resource manager API are called the *transaction services API* and provide all the services needed by transaction application programs.

The ACID properties are maintained for each managed resource by a *TP Resource Manager (TPRM),* coordinated by a *Transaction Manager.* The interface between the TP Resource Manager and the Transaction Manager will be called the *Transaction Manager/ TP Resource Manager SII* (*TM / TPRM SII*).

The ACID properties can be applied not only to resources such as databases, but also to other resources that might not be obvious. For instance, a transaction that dispenses cash may wait until the cash dispensing machine has signaled completion before considering the transaction complete and updating involved accounts. This illustration also shows the limits of transaction processing resource management. The machine could signal completion, but a mechanical problem could prevent the cash from being dispensed correctly, undetected by the system.

Besides database TPRMs and miscellaneous nondatabase TPRMs, a third class of of TPRMs exist: Communications TPRMs (cTPRM). Services provided by cTPRMs are used when two cooperating transaction application programs need to communicate with each other in the context of the same transaction. At least two communications paradigms have been identified as beneficial to cooperating transaction applications programs: client/server (RPC, single request/response) and conversational (peer-to-peer, dialog).

4.10.3.2 POSIX OSE Reference Model (with Transaction Processing)

The conventional transaction processing model is shown integrated into the POSIX OSE Reference Model in Figure 4-21. Because the POSIX OSE Reference Model does not address System Integration Interfaces (SIIs) per se, they are not shown in the integrated model. What is shown are the transaction processing services APIs and EEIs.

4.10.3.3 Real World Considerations

The POSIX OSE Reference Model does not provide for a way to expose the details of the Application Platform. In the Transaction Processing world, as shown in the conventional Transaction Processing Reference Model (see 4.10.3.1), the existence of Transaction Managers and multiple TP Resource Managers connected by the TM/TPRM SII is important. One way to think about the real world implications of this is that TP Resource Managers and the Transaction Managers could both be implemented in the Application Platform as separate entities, connected to each other by the TM/TPRM SII. This does not, however, imply that the two must be implemented as separate entities, though there are advantages to the user if they are separate.

4.10.4 Service Requirements

Services provided via the Transaction Processing Services API are described in 4.10.4.1. Services to enable the distribution of transaction processing are described in 4.10.4.2. General services, mostly performing administrative functions, are described in 4.10.4.4. Services provided across the TM/TPRM SII are briefly described in 4.10.4.5.

4.10.4.1 Application Program Interface Services

This subclause describes the major categories of services available at the POSIX API.

The Transaction Services API provides various services to the application programmer:

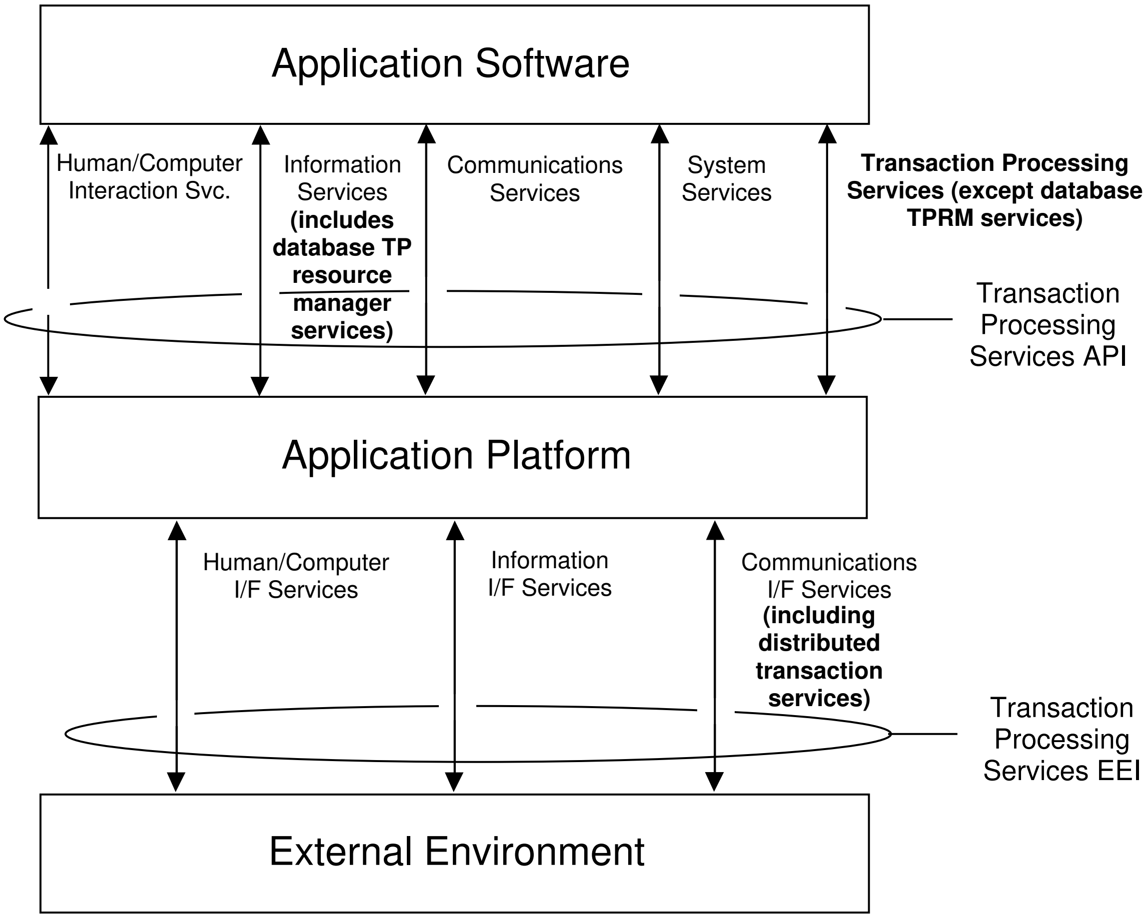

Figure 4-21 – The POSIX OSE Transaction Processing Reference Model

Transaction Demarcation

Ability to indicate the start of a transaction.

Ability to indicate a transaction has ended successfully (commit) or unsuccessfully (rollback).

Ability to indicate the beginning and ending of nested ‘‘subtransactions’’ whose commitment is independent of the ‘‘parent transaction’’. (Nested within a parent transaction can be multiple subtransactions. Subtransactions are independent of each other, and whether subtransactions commit or not does not affect the commitment of the parent.)

The ability to suspend and resume transaction mode (to do work which is not be committed or rolled back when the transaction is completed). This can be thought of as nesting nontransaction work within a transaction.

Ability to explicitly include or exclude items from a transaction. *RJG : PO—this was yours. Isn't this really the above nesting item?*

Communications Between Transaction Application Programs

Ability to call another transaction application program (possibly remote) within the context of a transaction.

Ability to open a dialog and send and receive ‘‘messages’’ to and from another transaction application program (possibly remote) within the context of a transaction.

NOTE: The above services provide ‘‘Distributed Transaction Processing.’’ The above services often go by shorthand names such as ‘‘Remote Procedure Call (RPC),’’ ‘‘Single Request/Response,’’ or ‘‘Client/Server,’’ in the first case, and ‘‘Dialogs,’’ ‘‘Conversations,’’ or ‘‘Peer-to-Peer’’ in the second case.

Terminal Communications

Ability to send and receive messages to and from terminals within the context of a transaction (i.e., messages sent to terminals are not to be actually delivered unless the transaction commits.

Transaction Program Scheduling

Ability to cause to be started another transaction application program outside of the context of this transaction. Involved here are two transactions: one starts the other. The actual scheduling of the second transaction can be dependent on the completion or not of the original transaction.

Transaction Message Queuing

Ability to define a ‘‘message’’ (based, possibly, on screen input from the end user) that, from the application point of view, precisely defines a unit of work to be done by this transaction or another transaction.

Ability to ‘‘send’’ a message to another transaction application program.

Ability to retrieve the next message (and then act upon it)

Ability to prioritize and associate start times with messages

NOTE: The actual handling of messages can be dependent on the completion or not of the original transaction.

NOTE: Several of the above services are similar to but semantically different from similar sounding services in other clauses of this section. They are listed here because they are ‘‘transactional’’; i.e., the concept of a transaction and the ACID properties are provided by these services.

TP Resource Managers provide services usable by the transaction application program and are made visible by the TP Resource Manager API. An example of this is the Database API covered earlier in this document.

NOTE: TP Resource Managers, in general, ‘‘protect’’ a critical resource. Databases are good examples of TP resource managers where the resource actually being protected is the data, for example, of an enterprise. Often the data of an enterprise reflects the amount of a real resource such as cash holdings. In this case a tangible resource is indirectly protected by a TP resource manager. The importance to the enterprise in insuring that the data (quantifying money) is accurate, should be obvious. Other TP resource managers, on the other hand, could protect an actual, tangible resource. An example of such a TP resource manager is the program that controls the cash drawer of an automated teller machine. The resource protected is the cash in the drawer. The actual application program interface of the TP resource manager protecting that resource could include the ability to reduce the amount of money in the drawer (by pushing it out of the machine). A transaction application program using two TP resource managers (a conventional database manager that keeps track of the balance in accounts, and the teller machine's cash drawer TP resource manager) would want to insure that the two TP resource managers decrement both the cash and the balance of the correct account in the context of a single transaction (i.e., with the ACID properties.)

The TP Resource Manager API, then, generally provides the following services:

(1) Ability to increment or decrement a valuable resource by a certain amount.

(2) Ability to determine the amount of a valuable resource that remains.

Specific capabilities for the very wide variety of specific TP resource managers, cannot, of course, be documented here.

4.10.4.2 External Environment Interface Services

When two or more machines are involved in the same transaction, the following service is required:

— The ability for two application platforms to interoperate with each other (pass along global transaction identifiers, participate with each other in commitment process, participate with each other in recovery).

4.10.4.3 Interapplication Software Entity Services

4.10.4.4 OLTP Resource Management Services

The services listed in this subclause are not provided by application program interfaces or external environment interfaces.

— Management Services — Ability to control the operation of the transaction processing services, including the ability to assign dispatching priorities to individual transaction application programs.

— Monitoring Services — Ability to collect data on resource utilization for purposes such as performance analysis and accounting (data on utilization of the transaction processing services resources: processes, connection pools, . . .).

— Modeling Services — Ability to predict the system resources needed to process a given transaction processing workload.

— Directory/Namespace Services — Ability to map names to addresses.

— Recovery/Restart Services — Ability to recover and restart transactions involving one or more transaction application programs using one or more TP Resource Managers.

— Test Services — Ability to automatically generate tests for workload simulation, etc.

— System Configuration Services — Ability to replace or add transaction application programs without the need to shut down the execution environment.

4.10.4.5 Services Provided at the TM/TPRM SII

HLJ : RJG asked for this to be an ''informative subclause.'' The only way to do that is to make it a NOTE:

NOTE: For application portability it is not required that the application platform actually consist of Transaction Managers and TP Resource Managers, but in the new age of Open Systems, there are clear advantages in doing so. Two advantages seem obvious: the ability to ''mix and match'' Transaction Managers and TP Resource Managers from different vendors; and the ability of a user to construct his/her own TP Resource Manager to manage particular critical resources. A market has already developed for ''plug compatible'' TMs and TPRMs. All TPRMs at this printing are Database type TPRMs. It is expected that a market will also develop for Communications type TPRMs. It is not at all clear that the industry will develop other types of widely applicable TPRMs, thus forcing users to develop their own. Users could use the interface described here to do such development work.

This NOTE very briefly describes the services that should be provided at such an interface.

The TM/TPRM interface must provide the ability of TMs and TPRMs to: register with each other; obtain recovery status information; pass along transaction identifier information; rollback, prepare to commit, and commit the transaction. The interface must provide for the needs of the full range of transaction processing including distributed transaction processing with multiple TPRMs.

Finally it should be noted that the industry recognizes the need for standardization of components as well as the standardization of portability and interoperability. At least one industry group is standardizing and several vendors are implementing products utilizing an interface as described here.

4.10.5 Standards, Specifications, and Gaps

There are currently three transaction processing standards development activities, either completed or in the draft stage. Table 4-11 summarizes the service requirements provided by the various standards.

Table 4-12 summarizes the applicability of the various standards to the various programming languages supported by the POSIX Open System Environment.

4.10.5.1 Current Standards in the POSIX OSE

4.10.5.1.1 International Standards

None.

4.10.5.1.2 Regional Standards

None.

Table 4-11 – Transaction Processing Standards Activities

Service	Specification	Subclause	
Transaction Demarcation	Emerging (P1003.11 {*ref*})	4.10.5.?	D
Communications	Gap	4.10.5.?	D
Terminal Communications	Gap	4.10.5.?	D
Program Scheduling	Gap	4.10.5.?	D
Message Queuing	Gap	4.10.5.?	D
EEI Services	Emerging (ISO/IEC DIS 10026-1,2,3 {*ref*})	4.10.5.?	
Management	Gap	4.10.5.?	D
Monitoring	Gap	4.10.5.?	D
Modeling	Gap	4.10.5.?	D
Directory/Namespace	Gap	4.10.5.?	D
Recovery/Restart	Gap	4.10.5.?	D
Test	Gap	4.10.5.?	D
System Configuration	Gap	4.10.5.?	D
TM/TPRM SI	Emerging (P1003.11 {*ref*})	4.10.5.?	D

Table 4-12 – Transaction Processing Standards Language Bindings D

Standard	N/A	LIS	Ada	APL	BASIC	C	C++
ISO/IEC DIS 10026-1 {*ref*}	•						
ISO/IEC DIS 10026-2 {*ref*}	•						
ISO/IEC DIS 10026-3 {*ref*}	•						
POSIX. 11 {*ref*}		•				•	

Standard	COBOL	C-LISP	Fortran	Pascal	PL/1	Prolog	
ISO/IEC DIS 10026-1 {*ref*}							D
ISO/IEC DIS 10026-2 {*ref*}							D
ISO/IEC DIS 10026-3 {*ref*}							D
POSIX. 11 {*ref*}							D

NOTE: N/A — Language bindings are not appropriate for this standard.

LIS — Language-independent specification is available.

Ada, APL, BASIC, — Language-dependent specifications exist.

4.10.5.1.3 National Standards

None. D

4.10.5.2 Emerging Standards in the POSIX OSE

OSI Distributed Transaction Processing (DTP)

ISO/IEC DIS 10026-1 {*ref*}
ISO/IEC DIS 10026-2 {*ref*}
ISO/IEC DIS 10026-3 {*ref*}

These standards, developed by ISO/IEC JTC 1/SC21/WG5, deal expressly with the OSI services and protocols for transaction mode communications in an OSI environment.

These standards provide for some of the communications services described in 4.10.4.1.

POSIX.11 POSIX Transaction Processing

POSIX.11 {*ref*}

The POSIX.11 working group, formed in 1989, is chartered to work on a profile for Transaction Processing within the POSIX OSE. In the process of developing that profile, it has identified a number of gaps in the standards coverage and is in the process of proposing base standardization activities to address those gaps. Specifically, P1003.11 is currently working on the following services identified earlier:

— Transaction Manager (TM) Services provided at the Transaction API.

— Services provided at the Transaction Manager/TP Resource Manager (TM/TPRM) SII. A typical TPRM is a database manager (e.g., SQL).

RJG: The above list should be adjusted later to reflect the then current state of .11's activities (in terms of approved PARs.) If no PAR is approved by the beginning of formal ballot, I'll fix this prose and move it to the Gap section.

POSIX.11 is working to assure that POSIX Transaction Processing work is consistent with the emerging work of OSI DTP (cited above), certain ongoing work of X/Open TP, several related POSIX activities (POSIX.1 {2}, POSIX.4 {*ref*}, POSIX.8 {*ref*}) and the work of ANSI X3T5.5 (RPC).

4.10.5.3 Gaps in Available Standards

4.10.5.3.1 Standards/Specifications Outside the POSIX OSE

Existing standards and emerging standards do not adequately address all the requirements identified earlier. While POSIX.11 is addressing some of the gaps, there are many other gaps still not being addressed by formal standards committees. Most notable is the work of X/Open TP. While not formally a standards making body, it is addressing most of the gaps, and its output will be potentially useful as the basis of a formal standard.

X/Open TP

This group published an ''Online Transaction Processing Reference Model'' in 1987 and in 1990 published ''Preliminary Specification—Distributed Transaction Processing: The XA Specification.'' The group is studying the use of OSI DTP, two-phase commit, and global transaction identifiers. The group is also actively exploring APIs in support of peer-to-peer distributed transactions.

The work of this group addresses several of the services addressed in this clause, including transaction demarcation and conversation services.

Consideration is also being given to allowing alternative interoperability standards including proprietary mechanisms. Additional APIs are being defined by X/Open TP to facilitate this.

4.10.5.3.2 Unsatisfied Service Requirements

Other than the work of X/Open TP, the following areas are not currently being addressed by standardization activities: communications, terminal communications, program scheduling, message queueing, management, monitoring, modeling, directory/namespace, recovery/restart, test, and system configuration.

4.10.6 OSE Cross-Category Services

Not applicable.

4.10.7 Related Standards

CCR

The following standards relating to commitment are related to the ISO/IEC DIS 10026 series and are referenced in DIS 10026:

ISO/IEC DIS 9804-3 {*ref*}
ISO/IEC DIS 9805-3 {*ref*}

See 4.3 for more information.

Remote Procedure Call

ECMA-127 {*ref*}

See 4.3 for more information.

SQL Standard Database Language

The following standards for SQL also provide transaction demarcation services for relational database access:

ANSI X3.135.1 (ISO 9075, FIPS 127)
ANSI X3.168

See 4.4 D

4.10.8 Open Issues

None.

4.11 Software Development Environments

Responsibility: Don Folland

4.11.1 Overview and Rationale

Software Development Environments are dealt with as a particular application area needing special attention for the following reasons:

— The domain of Software Development Environments is one of prime importance. Software development is a major area of expenditure for government and large commercial organizations.

— The need for standardization is being driven not only by the SDE vendors and users, but also by the Independent tool developers who want to get their tool products on as many vendor platforms as possible.

— The SDE domain calls not only for portability, but also for particular integration and interoperability requirements.

— The domain is primarily of interest to that user community that has large complex software development requirements, but it is also of interest to all application areas as software development is an enabling technology for all applications.

Software engineers seek more powerful assistance to improve productivity and quality in the software development process. Considered opinion currently favors Project Support Environments (PSE) underpinned in such a way that the facilities are capable of being implemented on different machines. A PSE needs a base holding information such as specifications, designs, code, schedules, configuration plans, tests, etc., to support the developers. The interface between the base and the tools must ensure portability of the tools. Again, these tools will be supported by relevant language standards.

Certain design methodologies themselves have been modeled formally to establish their degree of rigor and self-consistency. Function Point Analysis is one method of measuring software systems and computing productivity that is increasing in use. It measures inputs, outputs, and entities accessed to determine transaction size; it gauges technical complexity by reference to 19 characteristics. These are combined to give a measure of systems size. Productivity is the ratio of system size in function points to the effort required to produce or maintain the system.

Generally, software support for the development process is in its infancy and effective metrics have not yet been developed.